A computational biology framework

A data analysis tool to support biomedical engineers in their research

T.P.A. Beishuizen (0791613)
Email: t.p.a.beishuizen@student.tue.nl
1 October 2017 - 23 October 2018
Eindhoven University of Technology

Biomedical Engineering - Computational Biology (8ZM104)
Exam committee:
Dr. Dragan Bošnački (d.bosnacki@tue.nl)
Dr. Ir. Veronika Cheplygina (v.cheplygina@tue.nl)
Prof. Dr. Peter Hilbers (p.a.j.hilbers@tue.nl)

Data Science and Engineering - Data Mining (2IMC00)
Exam committee:
Dr. Dragan Bošnački (d.bosnacki@tue.nl)
Dr. George Fletcher (g.h.l.fletcher@tue.nl)
Dr. Ir. Joaquin Vanschoren (j.vanschoren@tue.nl)

October 16, 2018

Chapter 1

Abstract

The analysis of biomedical datasets is no straightforward task, every dataset has its own challenges and needs different approaches to overcome those. The goal of this thesis is to develop a framework that guides biomedical engineers in their initial steps of data analysis. Four representative dataset types are used for application and evaluation of the framework: microarray, mass spectrometry, clinical and survey datasets.

Firstly, feature selection methods are developed for microarray and mass spectrometry datasets. Experiments with tuning parameters of filter methods show that none of the datasets needed more than 200 features to accurately explain the output, as additional features did not increase the quality significantly. Unexpectedly, results show as well that wrapper methods outperform both filter and embedded methods in feature selection in both quality and feature preservation. Furthermore, the automated machine learning tool *TPOT* is significantly improved for datasets with a high number of features after adding a bias to feature selection methods as well as adding additional feature selection methods.

Secondly, methods to handle missing values are investigated for clinical and survey datasets. The results show that the missing values of features with more than 15% missing values are very hard to cope with for missing value handling algorithms. Those feature are better to be removed. The distribution of features with missing values are better preserved for imputation methods than for list deletion methods and of the imputation methods nearest neighbour imputation performed best. After investigating the quality of the datasets with list deletion or imputation, however, a simple mean imputation method outperformed other imputation methods, showing that more complex imputation methods are not needed for missing value handling.

Thirdly, the *metalearn* package is analysed for its exploration capabilities. This package is capable after several extensions to provide insight into several dataset issues and how to cope with them, therefore showing to be an accurate addition to the framework.

Based on the outcome of above listed research lines a computational biology framework is developed. This framework comprises of recommendations on critical issues for the investigated datasets, exploration of a set of potential issues, multiple preprocessing methods to cope with these issues and finally an automated machine learning tool that helps the user find the best machine learning algorithm for the dataset.

Contents

Chapter 2

Introduction

2.1 Introduction

Biomedical datasets are created with the goal of expanding biomedical knowledge and improvement of healthcare. Biomedical data is a generalizing term that describes different dataset types [1]. Examples of biomedical data are microarray data [2] and mass spectrometry data [3, 4], but also clinically derived data [5, 6] and survey data [7]. From a bioinformatics perspective these dataset types vary significantly [1] and therefore extracting information out of biomedical data is not a trivial task. A framework for data analysis can help guide biomedical engineers in the process of information extraction from their datasets. The framework can provide different options in processing the data, taking into account common dataset issues [8, 9, 10] and approaches to reach a certain goal [11, 12]. Such frameworks are proposed and discussed, however mainly are made for the integration of databases [13, 14], are made specifically for one research area [15, 16, 17] or are limited to one specific type of analysis [18]. Creating it for all types of biomedical data and all types of analyses becomes too broad, as there are major differences within datasets and analyses. Therefore the framework that is investigated and proposed in this research is based on data that can be put into a matrix and machine learning analyses. Matrix form data is one of the most common ways of data processing and machine learning is a well-known approach to analyse matrix form data. Framework topics are usually investigated separately in the biomedical world, for example feature selection [19, 20] and missing value handling [21, 22, 23]. This report is structured per topic, therefore related work for topics specifically can be found in these topic chapters.

2.2 Problem Statement

The main goal of this work is to create a framework for more efficient data analysis. A main research question is formulated and further elaborated in three smaller research projects. The main research question is:

What aspects are of importance to be included in a framework for Biomedical Engineers for more efficient data analysis?

After doing two case studies on different datasets (Appendices C and D) and discussions with biomedical engineers, the focus was put on three different topics within data analysis:

RQ1 *What feature selection methods show the best performance and as such should be added to the framework?* (Chapter 4)

RQ2 *What missing value handling methods show the best performance and as such should be added to the framework?* (Chapter 5)

RQ3 *Which initial analyses help find suitable preprocessing and data analysis algorithms?* (Chapter 6)

Hypotheses for these research questions are formulated in the three separate chapters. These hypotheses are also discussed in the same chapters to formulate an answer to these questions.

2.3 Approach

This research project underwent several phases to come to a final result. These phases are discussed to give an overview of the project.

1. An elaborate literature study was done that discussed several approaches on statistical analysis and machine learning (Appendix B) and with that study *Python* with an *Anaconda* environment was chosen for research (Section 3.2). Also a literature study on biomedical datasets was done and several example datasets were gathered (Section 3.1) In this literature study also the focus was made on matrix datasets, since most datasets are structured as a matrix or can easily be structured that way.
2. Two case studies were performed to define important sub-questions to the main research question (Appendices C and D). The first case study was done by analysing a previous graduation report on bariatric comorbidities [24] and showed that especially missing value handling is an important challenge in data analysis. The second case study was done by analysing public datasets on skin diseases [25, 26, 27, 28, 29, 30, 31, 32] and showed that feature selection becomes very important for datasets with a high number of features. In addition to these two case studies, an interview was conducted for additional insights for the framework. These insights included that an initial exploration of the dataset at the start would greatly increase the understanding of the dataset.
3. An elaborate analysis on feature selection was done to answer research question RQ1 (Chapter 4), as it was one of the more important topics coming out of the skin disease case study (Appendix D). In this analysis two experiments were conducted to test out several feature selection methods. The first experiment focused on the minimum number of features to be useful and the second experiment focused on the quality of a broad selection of feature selection methods. On top of that *TPOT* (Section 3.2), an automated machine learning tool, was used and improved for biomedical datasets with big number of features.
4. An elaborate analysis on missing value handling was done to answer research question RQ2. It was an important topic on the bariatric comorbidity case study (Appendix C) and a discussed and researched topic in literature [33, 34]. The distributions of features with missing values were compared between list deletion and imputation methods. Also an experiment was done to test the quality of the missing value handling, to find differences in predictive power of the dataset after missing value handling.
5. The dataset exploration tool *metalearn* (Section 3.2) was analysed for research question RQ3. It was compared with biomedical dataset issues and improved where necessary (Chapter 6). The addition of such a tool to the framework is intuitively very useful and is conducted due to the previously discussed interview.
6. A framework is proposed that combines all of these topics (Chapter 7). This framework is based on indicating potential issues per dataset type, showing more detailed issues after dataset exploration and recommending solutions to the given issues. At last, also a tool to compute the best machine learning algorithm for creating a model is given.

Chapter 3

Preliminaries

Three separate lines of research are conducted in this report. In these lines of research datasets and software are used, as well as several concepts and abbreviations. In order to prevent explaining these multiple times, all of them are mentioned in this chapter.

3.1 Datasets

The choice of a data analysis approach heavily depends on the nature of the data. The amount of data in the biomedical world is growing at an enormous rate, faster than biomedical engineers can analyse. Several additional challenges came up with this uncontrollable growth. These challenges are in many cases focused on data volume, dimensionality, complexity, heterogeneity and quality [35, 36].

Scientists tend to collect abundant data, which makes data sets bigger than needed. Both in number of instances and features, data sets are harder to understand or analyse when more instances and features are available [35]. This volume problem usually is tackled by taking sub-populations of the complete set. Sub-sets can either be focused around a part of the population (e.g. gender, age, race) or taken at random to still represent all of it. Due to the efficiency of analysis techniques and the rise in computational speed of servers [37], volume on its own becomes less of an issue. Volume does however become a challenge when it is combined with heterogeneity and quality [38, 39].

Not all data sets have a high number of instances that causes a big data volume. Sometimes there are relatively few instances, while the number of features is disproportionally high [40]. This is also called high dimensionality (high number of features and low number of samples). Usually many of those features are not relevant enough for research, however are still used for testing. Trying to remove features that are not important significantly helps finding relations between the other features and create more knowledge about the research topic. Lowering the number of features also makes the data volume decrease, therefore making analysis easier. Essentially an optimal features set should be selected to obtain the best results [41].

A high dimensionality challenge can also be solved by creating more samples. If data is gathered for only a couple of patients, results will hardly ever be consistent. Most analyses require a high number of samples and give biased results when not enough samples are present. There are two ways of dealing with this low number of samples, the first one generating more samples [42, 43] and the second one using the samples very efficiently [44, 45]. Most likely a combination of both gives the best results.

Biomedical data can also be very complex. Useful information may be present in the data, but very hard to obtain. Examples of complex data are images, several biomedical signalsit was improved where necessary and temporal data. Useful information present in images is for example sometimes very hard to find, temporal data can vary significantly over time and biomedical signals can be hard to combine with static biomarkers [34]. This aspect benefits from exchanging

knowledge with other research areas that specialize in mining those complex data sets [38, 46].

The biggest challenge consists of aligning different data sets. Dataset standards are present, but not widely adopted and therefore data sets sometimes differ much from each other. Data is weakly structured or even unstructured [39] and variables are processed differently due to different protocols or the collectors' preference of representation [47]. Also the variety of data is hard to combine when sources are fundamentally different. When the data partially consists of images, another part is a table from the laboratory and a third part is textual remarks of the doctor, creating a merge of those three is much harder than merging three clinical sets. Those merges are also very prone to errors, as imprecisions can be significantly different between those data sets. No tool works directly with these raw data sets and preprocessing almost definitely has to occur beforehand [33, 38].

A last challenge is about data quality. Data are usually gathered by doctors, scientists or laboratory workers. Since the data are manually gathered by humans, the data have a relatively high error rate. The data can be quite noisy, values can be inconsistent, wrongly entered or even missing [33]. Not only human errors cause the data quality to drop, but the heterogeneity as well. Two hospitals can have different protocols for the same treatment and sample different biomarkers for that protocol. Due to that difference, biomarkers can be missing for some of the entries. The time of data gathering is also a big factor as some biomarkers change significantly over time. The databases are usually also built for financial purposes and not for research, which can diminish the quality [34].

These challenges within the data are often discussed [46]. Many proposals to tackle them are made, however none is actually widely adopted, yet, as a global standard for databases. Also, with the uncontrolled growth in biomedical data, it will become hard to have such a standard recognised all over the world [47, 48, 49, 50, 51, 52].

3.1.1 Micro-array Datasets

Micro-array data is a fairly new type of biomedical data. Datasets of this kind have information on expression levels of a high number of genes [2]. These expression levels can be used for gene level research, such as genome mapping, transcription factor activity and pathogen identifications. The data is gathered by fixing DNA molecules on a glass slide on specific spots. The nucleotide base pairs of those fixed DNA molecules can form hydrogen bonds with RNA corresponding to genes, testing the expression of those genes by how many bonds are made [53] (Figure 3.1).

Micro-array data is known to present challenges in data quality and needs normalisation for the data to be useful [53]. The thousands of features that are based in the data also indicate that feature selection is very important, so the irrelevant genes can be removed. Aside from that, size may also be an issue. Due to the sheer size of the data not all analyses will perform optimally in both time and quality. In this research two microarray datasets are used for research:

- *Psoriasis microarray dataset*
  This dataset is comprised of five different data sets [25, 26, 27, 29]. These five different datasets consist of 54675 features, all corresponding to gene expression. Samples were collected from three different test subject groups: affected skin from test subjects suffering from psoriasis (214 samples), unaffected skin from test subjects suffering from psoriasis (209 samples) and skin from healthy test subjects (85 samples). Combining these three sample types gives 508 samples. Since the data comes from five different experiments, the data is normalized for every experiment.

- *Arcene: Cancer microarray dataset*
  This dataset, called Arcene dataset, is used in a challenge focussing on classification problems with a low number of samples, but a high number of features [55]. It consists of the same number of features as the Psoriasis data set, 54675 features corresponding to gene expression. It also has 383 samples corresponding to nine different test subject groups. The challenge did not provide labels for the test subject groups. Also these groups differ in size, one group corresponding to 150 samples and the others varying from 16 to 47 samples.

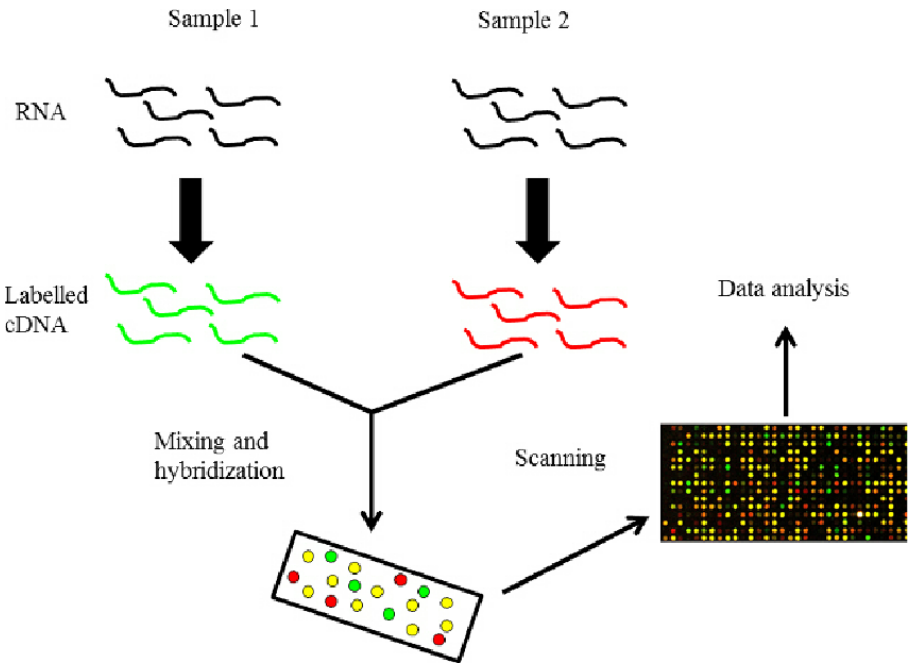


Figure 3.1: A picture showing the creation of microarray datasets [54].

3.1.2 Mass Spectrometry Datasets

Mass spectrometry data contains information on proteins and peptides [3, 4]. Analysis of this information is used in studies about proteins, also known as proteomics [56]. Mass spectrometry is a technique that can be used to find how much of a certain protein is present. This is done by measuring the mass of gas-phase ions originating from molecules of that protein, showing that protein is present. On top of that it can also measure the quantity of the mass, therefore showing how much of a protein is present [57].

Challenges in the analysis of mass spectrometry datasets are mainly found in the way this data is produced. The expression is given for several proteins and this expression can differ significantly between proteins. Therefore some type of normalisation is useful. Aside from that, in this technique usually a high number of proteins is tested at the same time. Therefore feature selection can be helpful to remove irrelevant genes. Two mass spectrometry datasets were used in this research:

- *RSCTC: Cancer mass spectrometry dataset*
  This dataset, called RSCTC dataset, was created as a classification problem to distinguish cancer patterns from normal patterns [59]. It is created for the 'Neural Information Processing Systems' conference by merging three mass spectrometry datasets. It consists of 10000 features corresponding to either spectra of the mass spectrometry or probe variables without any predictive power. Samples from two groups are taken, from patients with ovarian or prostate cancer and from control patients. No labels are given to the groups, however it is known that one of the groups has 88 samples and the other 112 samples, combined in a total of 200 samples.

- *Micro organisms mass spectrometry dataset*
  This dataset is created to back up a proposed method for routinely performing direct mass spectrometry based bacterial species identification [60]. It consists of 1300 features corresponding to different spectra of the mass spectrometry data and 20 test subject groups corresponding to Gram positive and negative bacterial species. Gram classification is a result

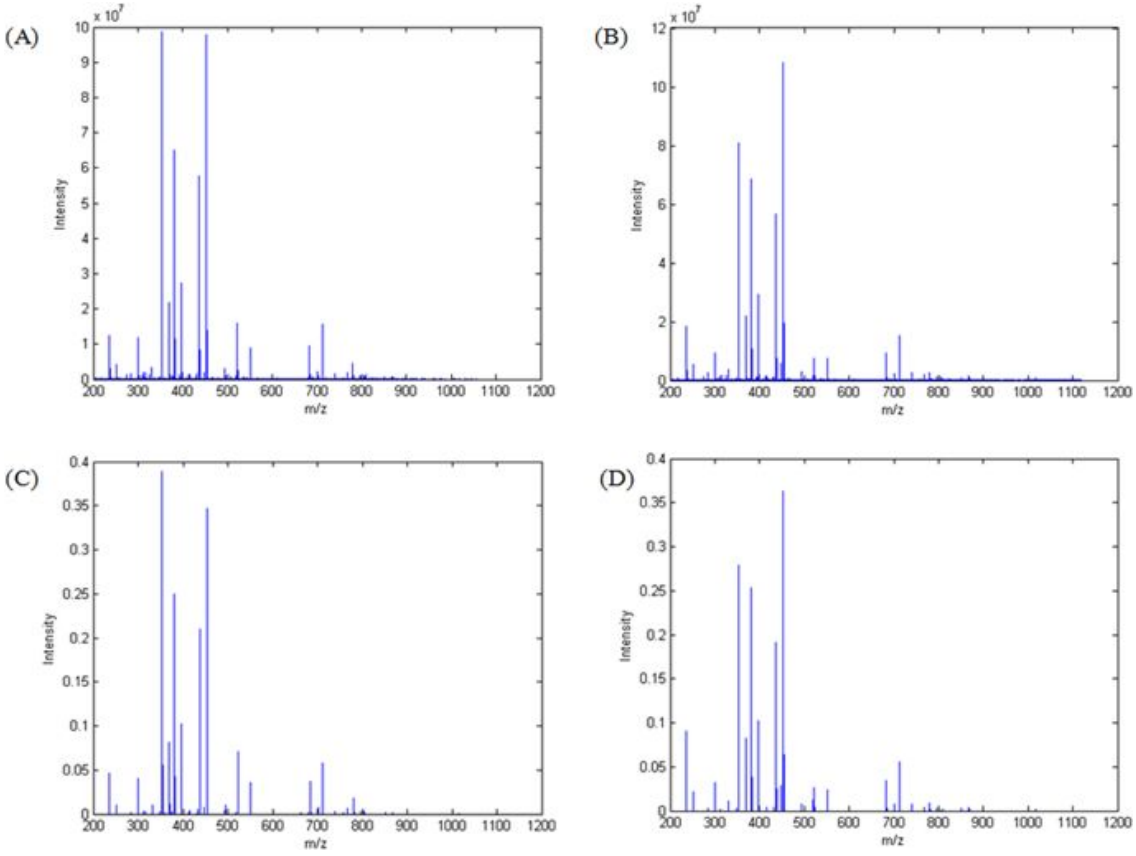


Figure 3.2: Four example figures of the data created by mass spectrometry. The measured molecule masses correspond to the peaks in the figures [58].

of a Gram stain test [61]. The groups differ in size varying from 11 to 60 samples, making a total of 571 samples.

3.1.3 Clinical Datasets

Clinical datasets originate from clinical research [5, 6]. In clinical research data is created to determine the effectiveness of treatments in the hospital, for example for a drug [62]. The values in clinical datasets can originate from blood tests [63], but can also be data derived from other fluids or tissues of the body.

The clinical datasets that are analysed by data analysts usually are created by doctors and laboratory researchers. This sometimes causes mismatches between the data and the goals of the biomedical engineer working with the data. Main challenges are the bias in the data due to difference in focus of the doctors, therefore having misrepresentation of certain patients. Often values are missing in the data and errors are present due to humans entering those values [62]. At last, the number of samples that can be gathered is usually limited, due to the clinical trials being limited to the number of patients present. Three clinical datasets are used in this research:

- *Heart Attack Echo-cardiogram dataset*
  For this dataset patients were selected, who at one point in their lives suffered a heart attack and survived [64]. A prediction model can be made that models the months of survival after the heart attack as the prediction variable. There are 9 features for 108 samples present, with a total of 97 missing values. Every feature but two have at least one missing value, but by far the most missing values are located in the feature "alive-at-1" (still alive after 1 year) with 58 missing values and 54% of total number of samples. This high number of missing values can be partially explained because of lack of documentation, partially because no year had passed, yet, after data distribution.

- *Hepatitis dataset*
  The mortality rate of hepatitis was tested using 19 features over 155 samples [65, 66]. Two class types are distinguished as "died" and "lived" with thirteen boolean and six numeric attributes. The total number of missing values is 167 spread over 15 classes, with the feature labelled "histology" having the majority of 67 missing values. Two previous articles were written, using this dataset as an example for their methods [65, 66].

- *Primary Biliary Cirrhosis dataset*
  Patients were classified in three groups in this dataset: alive, transplanted and dead [67]. 18 features are present in the dataset: one date-specific, two categorical, five boolean and ten numeric attributes. The dataset consists of 1945 instances and 1133 missing values. The missing values are distributed over six features, the feature *serum_cholesterol* having 821 and the other five around 60 missing values. The missing values are noted as most likely not missing completely at random.

3.1.4 Survey Datasets

Whereas opinions are divided whether biomedical surveys can be seen as clinical data [68], in this report they are mentioned separately. Survey data simply consists of answers to biomedical questions. Patients answer questions related to some kind of biomedical subject for analysis. Due to the simple way of conducting such a survey, many patients can fill out the questionnaire and therefore many samples are created.

Challenges in surveys mainly consists of errors in filling out the answers. The patients' answers can be very subjective, possibly creating bias. Also human errors are present when filling out surveys, as well as accidentally or purposely refusing to fill out an answer. One survey dataset was used in this research:

- *Cervical Cancer dataset*
  This dataset is consisting of 858 samples and is based on the increased risk of subjects

Table 3.1: An overview of the symbols used to explain the data

Value	Description
m	The number of features
F	A list of (m) features
n	The number of samples
S	A list of (n) samples
X	A matrix of $(n \times m)$ dataset values
y	A vector of (n) output labels
W	A list of feature weights

becoming cervical cancer patients [69]. Since a risk can be fairly subjective, four aspects were used to determine whether an increased risk is present or not: two doctors, a cytology examination and a biopsy examination. For analysis a new increased risk factor was made by giving each subject a value in $[0, 4]$, implying the number of times this subject has been classified as an increased risk subject. The dataset consists of 31 features, 24 boolean and numeric features originated from a survey and 7 boolean and numeric features originated from the hospital. A total of 3622 values are missing of which 787 times the hospital features "STDs: Time since first diagnosis" and "STDs: Time since last diagnosis" caused by no sexually transmitted disease (STD) ever being present. All other missing values originate from subjects not answering survey questions, the "Age" feature being the only survey question that was always answered.

3.1.5 Data Symbols

Often in this research project, equations and pseudo-algorithms are discussed. In these equations and pseudo-algorithms, several symbols were used to explain the data. A quick overview of these symbols is given (Table 3.1) as well as a more detailed description:

- F - A list that contains all the names of the features. It is a list of size m with m being the number of features. A subset x of F is written as F_x.
- S - A list of all samples in a dataset. It is a list of size n with n being the number of samples. A subset x of S is named S_x and a single sample is usually called s.
- X - A matrix that contains all sample values for every feature. It is an n by m matrix with n being the number of samples and m being the number of features. If the values of a specific feature f or a subset of features F_x are used this is written as X_f and X_{F_x} respectively, hence the column f or columns F_x are selected. Similarly for X_s and X_{S_x} the row s or rows S_x respectively are collected from X.
- y - A vector that contains all class labels for every sample. It is a vector of size n with n being the number of samples.
- W - Weights given to a feature after training a feature selection technique. The weight of feature f or set of features F_x is called W_f and W_{F_x} respectively.

3.2 Software

For this research one programming language is used, *Python* version 3.6. This language was chosen, due to the numerous possibilities in preprocessing and analysis in the open source language [70] (Appendix B). *Python* makes use of packages containing additional classes and methods to be used in experiments. To easily import these packages the *Anaconda* version 3 environment is used [71]. Packages available by *Anaconda* and often used in the analyses are:

- *NumPy* [72]
  NumPy provides data structures efficient in use for data analysis. On top of that it also provides multiple methods to alter and preprocess these data structures, improving data analysis efficiency significantly.

- *SciPy* [73]
  SciPy provides analysis methods, mainly focused on statistics. These methods are compatible with the *NumPy* data structures and provide a way to simply perform statistical analyses.

- *Scikit-learn* [74]
  Scikit-learn, also known and shortened as *sklearn*, is focused around machine learning. It provides numerous simple machine learning algorithms as well as several other algorithms and classes that are often used in analysis, such as preprocessing methods and environment testing creation.

- *Pandas* [75]
  Pandas is also used in parts of this research. *Pandas* creates a very intuitive way of distributing data, presenting datasets that are easy to understand for users and in analysis. It is an overlay of the *NumPy* data structure and often used in *Python*.

Two other packages were used in this research that are not part of *Anaconda*. These packages have to be downloaded separately:

- *TPOT* [76]
  TPOT is an abbreviation for "Tree-based Pipeline Optimization Tool" and uses automated machine learning to find the best machine learning pipeline. It makes use of several other packages, such as the *NumPy*, *SciPy* and *Scikit-learn*, all of these being available in *Anaconda*.

- *metalearn*
  metalearn is a package that computes meta-features (Appendix G.1) for datasets. These meta-features can be used for further analysis or can give a direction for further analysis. *metalearn* uses packages available in *Anaconda*, as well, such as *NumPy*, *SciPy*, *Scikit-learn* and *Pandas*.

3.3 Concepts

Several concepts occur in more than one of the three research chapters. These concepts are explained in this section to avoid overlap:

- *Cardinality*
  The cardinality of features is the number of unique options available. For categorical and ordinal data it is the total number of categories and for numeric data it is every unique numeric value.

- *Correlation*
  Correlation computes the expressibility of two variables in regard to each other. A variable is relevant if it can predict the outcome of the another variable. One way of computing this relevance for a variable x_i of feature i with mean $\bar{x}_i$ and target variable y_i with mean $\bar{y}_i$ is by using the linear correlation coefficient r, also known as Pearson' correlation [77] (Equation 3.1). The correlation filter methods mainly works for continuous data, but can also be used for discrete data [78].

$$r(x,y) = \frac{\sum_i (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sum_i (x_i - \bar{x}_i) \sum_i (y_i - \bar{y}_i)} \tag{3.1}$$

- *Cross validation*
  Cross validation is a model validation technique. A dataset is split in two groups: training and validation data. The training data is used for training the algorithm and the validation data is used for validating it. Splitting the data can only be done once, but also more times to create multiple folds. Multiple types of cross validation are known, examples are k-fold cross validation, bootstrapping and leave-one-out [79].

- *Leave-one-out*
  Leave-one-out is a type of cross validation. For a dataset of size n, n times the data is split into a training set of $n-1$ samples and a validation set of the other 1 sample. All samples are in the validation set exactly onece. The algorithm is then trained and validated for the n splits and the validation scores are averaged. This way the validation score is found while using as much training data as possible [80].

- *Data types*
  Feature values can be of multiple types, for example: numbers, text, dates and other possibilities. The usefulness of these types can differ between analysis, showing that not all techniques are suitable for a dataset. Usually all of these types are put in one of these three groups:

  - *Categorical data*: This type of data splits the features into different categories, giving a finite number of options for the feature values. These categories are usually text-based and lack special relations between them, such as an ordering [81].
  - *Numeric data*: Features that have numbers as values that originate from a predefined range are of the numeric type. Numeric values can be ordered. Also the cardinality (number of distinct feature values) usually is much higher for numeric data [82].
  - *Ordinal data*: Ordinal data are in between categorical and numeric data. Ordinal features have multiple predefined categories that have some kind of ordering. The number of categories usually is very low, giving the outcome a low cardinality [83]. Ordinal data are harder to define as a data type, due to the lack of ordering in text based values and the lack of possibilities in defining only a finite number of categories in numerical data. Therefore usually the most suitable of categorical data and numerical data is chosen and ordinal data are transformed to that type.

- *Dimensionality*
  The dimensionality of a dataset in literature is the number of features of that dataset[77, 87, 89]. A high dimensionality therefore indicates a high number of features. The package *metalearn* calls dimensionality the number of features divided by the number of instances. If this definition of dimensionality is used it is specifically stated. This definition is used to show possible imbalance between the number of features and instances and normally the dimensionality should be very low, at least < 1.

- *Experiment set-up*
  When doing machine learning experiments, a set-up is made that takes into account training the data, validating the result of the training and testing the data set with separate data. The tests are exectured with a to be trained algorithm and a scoring function that shows the quality of the trained algorithm. Three phases are present during the experiment:

  - *Splitting the data*
    The data is split into a training set and test set at the start, usually more than 75% of the data is chosen to be in training set. The training set is split in training data and validation data. This usually is done by cross validation or leave-one-out.
  - *Training and validation*
    The algorithm is trained by giving it the training data. Afterwards with the validation data a score is computed for how well the algorithm can predict the validation data.

Since with cross validation and leave-one-out multiple combinations of training and validation data are made, the algorithm is separately trained and validated multiple times. The separate validation scores are combined to one validation score, usually by averaging all of them. This validation score then shows how well the algorithm trained by the training set would perform on the training data.

- *Testing the algorithm*
  Since also new data should be tested on the algorithm, at last the initially created test set is used. A score is computed on how well the algorithm, trained on the complete training set, can correctly predict the test data. This data that has no direct relation with the training data shows how well the algorithm performs when new data is present.

- *Hot encoding*:
  One-hot encoding is the actual name, but in the report it is called hot encoing. Hot encoding is used when numerical and categorical values are combined in the dataset. Whereas numerical features have numbers as values, categorical features usually correspond to text with no clear order. Hot encoding is used to give numerical values to categorical features. This is done by creating a new separate feature for every category and differ the values between presence of this category (e.g. the feature value becomes 1) and absence of this category (e.g. the feature value becomes 0). An example would be for a patient either undergoing treatment 'A', 'B' or 'C'. The categorical feature 'treatment' then has the options 'A', 'B', 'C'. After hot encoding the 'treatment' feature is replaced by features 'treatment A', 'treatment B' and 'treatment C', and the values of these features are either 1 if that treatment is performed or 0 if that treatment is not performed. This way no ordering is present for the categorical data.

- *Mutual information*
  Mutual information can be used on two different variables. The relevance of using one variable to predict the other variable is used in the equation to compute mutual information (Equation 3.2). In this equation the probability density function p is used to find the mutual information (MI) between variables x and y [84]. Mutual information can be used for both classification and regression. It was initially meant for use in communication channels, however its statistical decision making capabilities made it a good filter method [85].

$$MI(x, y) = \sum_{x} \sum_{y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \tag{3.2}$$

- *Normalisation*:
  The difference in value size between numerical features can differ greatly. For example, if both the length of a patient's body and finger are measured, the value of body length is much bigger than the value of finger length. On top of that the variance in the features body length and tongue length differ in size, too. To prevent this potential bias in data analysis, all features are normalized on both mean and variance. This normalisation is usually done in two steps for a feature with mean μ_0 and standard deviation σ_0: Step one is changing to a mean value μ_1 with value $\mu_1 = 0$ by subtracting all values by the mean μ_0. Step two is to divide all features by the standard deviation σ_0, so the new standard deviation σ_1 becomes $\sigma_1 = 1$.

- *Scoring functions*
  This quality of an algorithm in machine learning is measured with a scoring function. Three of these scoring functions are explained. They are defined in terms of true and false positives (TP and FP respectively) and true and false negatives (TN and FN) and in these discussions n is the size of the test set.

  - *Accuracy*
    The simplest and most used approach to measure quality is by using accuracy. This is

usually the choice when no imbalance is present and there is no specific focus in results. Accuracy is simply the ratio of correctly classified samples and all classified samples (Equation 3.3). So for a test set of 30 samples, if 6 samples are incorrectly classified the accuracy would be $24/30 = 0.8$.

$$\text{accuracy} = (TP + TN)/(TP + FP + TN + FN) \tag{3.3}$$

- *F1 score*
  The accuracy assumes there is no difference in output categories. If the output is unbalanced and majority and minority classes exist, another approach can be used called the F1 score. The F1 score treats classes equally by computing the precision and recall and combining those as a harmonic mean. Precision shows the portion correctly classified out of all positively classified (Equation 3.4) and recall shows the portion correctly positively classified out of all positive values (Equation 3.5). After that both are balanced in the F1 score by computing a harmonic mean of precision and recall (Equation 3.6). Therefore F1 evaluates both FP and FN evenly. Precision and recall are also sometimes used if one of the false classifications FP and FN is more important than the other.

$$\text{precision} = TP/(TP + FP) \tag{3.4}$$

$$\text{recall} = TP/(TP + FN) \tag{3.5}$$

$$\text{F1} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.6}$$

- *Cohen's kappa*
  Cohen's kappa κ is a statistical approach for a quality label. κ shows the agreement between the result p_0 (for example the aforementioned accuracy) and compares this with the expected agreement by chance p_e (Equation 3.7) It is mainly used if the output can be two classes [86] and does not take into account random classification. Therefore it gives a better indication of the effectiveness.

$$\kappa = \frac{p_0 - p_e}{1 - p_e} = \frac{2(TP \cdot TN - FP \cdot FN)}{(TP + FP) \cdot (TN + FN) + (TP + TN) \cdot (FP + FN))} \tag{3.7}$$

- *Standardisation*
  Standardisation is an alternative to normalisation, a technique to use when value sizes differ greatly between numerical features. Normalisation is based on changing values to a $[0, 1]$ range. The minimum value is changed to 0, the maximum value to 1 and all values in the middle are changed accordingly to fit the range $[0, 1]$.

Chapter 4

Feature Selection

4.1 Introduction

For a biomedical engineering framework dimensionality reduction is important. Some biomedical datasets contain a high number of features, whereas only a selection of those features are interesting. A direct link of features to the output is desired, therefore especially dimensionality reduction based on feature selection is important. Multiple projects attempted to reduce the number of features [19, 20] which resulted in multiple algorithm proposals [87, 88, 89, 90] and tests on their performance [91, 92]. Basic feature selection algorithms should be present in the framework and are therefore tested on the available data sets. The research goal is *to evaluate the performance of feature selection methods and make a choice on which methods should be added to the framework.* In this document several of those algorithms are presented and test their quality.

Table 4.1: A schematic overview of the four datasets.

Dataset focus	Data type	Features	Samples	Classes	Remarks
Psoriasis	Micro-array	54675	580	3	- Derived from five different datasets [25, 26, 27, 29] –
Cancer	Micro-array	54675	383	9	- Used in a data mining challenge [55]
Cancer	Mass Spectrometry	10000	200	2	- Created for the NIPS conference [59] - Several probe features are present –
Micro-organisms	Mass Spectrometry	1300	571	20	- Originates from a micro organisms study [60]

4.2 Background

Before testing several feature selection methods, first the background of the study is explained. Firstly, the datasets and their characteristics are explained. Secondly, feature selection methods are discussed. Thirdly the concept automated machine learning is discussed as a possible technique to add to the framework, combined with a tool that implements automated machine learning.

4.2.1 Datasets

Four datasets were used as a case study for the feature selection algorithms (Subsection 3.1). Two sets are microarray datasets that are used for research on psoriasis [25, 26, 27, 29] and cancer [55]. Two other sets are mass spectrometry data sets, used for research on cancer [59] and micro organisms [60] (Subsection 3.1). These four datasets all have a high number of features varying from 1300 to 54675 features with a number of samples varying from 200 to 580 samples (Table 4.1). All of these datasets are based on classification as tests are done for different test subject groups, therefore the focus on feature selection algorithms will also be based on classification. Many features are expected to be irrelevant in this classification and therefore could be removed with feature selection.

4.2.2 Feature Selection Methods

Feature selection is a way to perform dimensionality reduction. In feature selection a subset of features is chosen to represent the complete sample space [93]. Several techniques are available to choose a representation subset and the effectiveness of these techniques has been tested multiple times [94, 95, **?**]. These techniques can be grouped accordingly in three different categories: *filter methods*, *wrapper methods* and *embedded methods* [96]. Each of these methods is explained in what follows. For better understanding of the different feature selection methods, pseudo-algorithms are created for each of them.

Before actually performing feature selection, usually some kind of preprocessing is done. This preparation consists of making the data adequate for feature selection to be possible as well as preprocessing, such that all features are processed evenly. Therefore normalisation of numerical values is done and possible hot encoding if categorical features are present.

Filter Methods

Filter methods are based on giving relative ranks to the features in a feature space. All features are given a value based on their performance and are ranked by those values [97, 96]. Several methods are used to rank the features. These methods are usually based on splitting data matrix X twice.

First they are split by feature ($X_{f_1}, X_{f_2}...X_{f_n}$) for computation. Secondly the columns are split by output classes $c_1, c_2...c_n$ ($X_f^{c_1}, X_f^{c_2}...X_f^{c_p}$) to distinguish the predictive power for features between classes. A ranking method would quantify the significance according to these values. This results in ranking method $R(X_f, y)$ in which sample label y is used to split the data per class. For illustration a ranking method collection is given:

- *T-test and ANOVA*
  Statistics can be used to compare groups with each other. In statistics these groups are seen as separate distributions, which may or may not be seen as independent distributions. If there are two groups a t-test can be done to find the chance for these groups to originate from the same distribution [98] (Appendix B). For sample classifications the groups would consist of samples with the same label ($X_f^{c_1}$ and $X_f^{c_2}$). For example, the t-test can be done between group with class c_1 and group with class c_2. If the values of a certain feature from these groups are highly unlikely to follow the same distribution ($R(X_f^{c_1}, X_f^{c_2})$), the p-value will be very low and therefore the rank will be higher.

  If there are more than two groups that must be checked whether they are from the same distribution ($R(X_f^{c_1}, X_f^{c_2}...X_f^{c_p})$, a t-test cannot be used. In this case the option is available for a multiple group testing, also called analysis of variance (ANOVA). ANOVA computes whether not only two, but multiple groups can originate from the same distribution. ANOVA needs equal group sizes, otherwise the results of the tests are less reliable. The less powerful Kruskall Wallis test can be used if this reliability is needed. Another disadvantage is that t-test and ANOVA should only be used if the groups follow a normal distribution, which might not be the case (Appendix B) [98].

- *Mutual information*
  Mutual information is another way of matching features with the results. The relevance of using one variable to predict the other variable is used in the equation to compute mutual information.

- *Correlation*
  Correlation, similarly to mutual information, computes the expressibility of two variables for each other. A variable is relevant if it can predict the outcome of the target variable.

These ranking methods R to rank the features are only the first step in the filter methods. The next step is choose which features to filter out based on R. A quick approach would be to choose a number or fraction n and select the top n ranked features from the complete space (Algorithm 1). Another approach would be to use thresholds derived from literature [99] (Algorithm 2). For the t-test, a p-value of 0.05 is often chosen as the threshold value [100, 101].

Algorithm 1 A basic top n filter algorithm [97]

```
procedure FilterSelection(X, y, F, R, n)
    F_selected ← ∅                          ▷ Start with empty feature set
    Z ← ∅                                   ▷ Start with empty set of ranking values
    for f in F do                           ▷ For all features in F
        Z_f ← R(X_f, y)                     ▷ Compute the ranking value between feature and output
        Z ← Z ∪ {Z_f}                       ▷ Add the ranking value to the set of ranking values
    end for
    Sort F by Z                             ▷ Sort the features by their ranking value
    Fselected ← F_[1,n]                     ▷ Select the top n features from the sorted feature set
    return F_selected
end procedure
```

Algorithm 2 A basic filter algorithm [97]

```
1: procedure FilterSelection(X, y, F, R, α)
2:     F_selected ← ∅                                  ▷ Start with empty feature set
3:     for f in F do                                   ▷ For all features in F
4:         if R(X_f, y) > α then                       ▷ Check if ranking value is higher than threshold α
5:             F_selected ← F_selected ∪ {f}           ▷ Add f to selected features
6:         end if
7:     end for
8:     return F_selected
9: end procedure
```

Using filter methods has its advantages and disadvantages. These filter methods are very computationally efficient. Every feature is given a value for its rank and then a subset is selected based on those ranks. These filter methods however do not take into account dependencies between features. These dependencies could make the final feature subset worse, as maybe some features are related. Other methods are better at handling those dependencies [97, 96].

Wrapper Methods

Filter methods take into account the direct relation between features and the output classes, whereas wrapper methods focus more on the subsets of features and their ability to classify the data. Wrapper methods try to find the best combination of features to classify the given data and take into account the change when adding or removing features from a candidate subset. The choice of subset selection is "wrapped" around the quality computation, therefore this method is called a wrapper method [102]. Since an exhaustive search of trying out all possible subsets would span a computation time of 2^n with n being the number of features [103]. This combinatorial explosion should be avoided and therefore less computationally intensive approximation concepts have been constructed. Three of those concepts are explained, focusing on basic sequential search, extensions of sequential search and stochastic search.

Before explaining the possible wrapper methods, first the evaluation function J should be discussed. To find out which subset of features can classify the data the best way, a function should be used to evaluate the performance of the subset [102]. Evaluation functions can be based on conditional independence [102, 104], showing the difference in performance for a subset with and without a certain feature. Other evaluation functions are based on machine learning techniques [105, 96], rating the ability to classify the outcome. Several interesting approaches for evaluation functions are shown in maximum relevance, minimum redundancy algorithms [106, 107, 108] (MRMR algorithms). In these MRMR algorithms, several different evaluation functions are used based on the ability of features to classify the outcome (relevance) and the presence of correlation between features (redundancy). In this project, the evaluation function is fixed on the machine learning algorithm Naive Bayes, due to its understandability.

- *Sequential search*
  The first concept to be explained is sequential search. This wrapper method tries to improve a candidate subset by evaluating the change of the sequential addition or removal of a specific feature. Therefore, two types of sequential search are possible, known as forward selection and backward selection. Forward selection starts with the empty subset. Every feature is iteratively evaluated and added to the feature subset if the evaluation function shows an increase in prediction. A maximum feature subset selection size l can be defined if needed, as well. The layout of forward selection is shown as a pseudo algorithm (Algorithm 3) for understanding [102].

  Backward selection does the opposite of forward selection. It starts with the complete feature set in which every feature is iteratively evaluated by the evaluation function for its contribution. If the evaluation function shows that its contribution is very small, it is

removed. This way only features with a high impact will remain in the subset. The maximum number features that can be removed r can be defined if needed, as well. be A layout of backward selection is shown as a pseudo algorithm (Algorithm 4) for understanding, as well [102].

In both forward and backward selection the sequence order is important. Different feature subsets will be made for different order of features. The ordering can be changed in a specific way if needed. Examples would be using the ranking concepts used by filter methods. Another possibility would be to randomize the order and run the algorithm multiple times to find the best subset [102].

Algorithm 3 A forward selection sequential search algorithm [102]

procedure SEQUENTIALFORWARDSELECTION(X, y, F, J, α, l)
  $F_{selected} \leftarrow \emptyset$ ▷ Start with empty feature set selection
  for f **in** F **do** ▷ For all features in F
    if $J(X_{F_{selected}}, y, X_f) > \alpha$ **then** ▷ Check if evaluation is higher than α with feature f
      $F_{selected} \leftarrow F_{selected} \cup \{f\}$ ▷ Add f to the feature set selection
    end if
    if length$(F_{selected}) = l$ **then** ▷ Stop if size of feature set selection has been reached
      break
    end if
  end for
  return $F_{selected}$
end procedure

Algorithm 4 A backward selection sequential search algorithm [102]

procedure SEQUENTIALBACKWARDSELECTION(X, y, F, J, α, r)
  $F_{selected} \leftarrow F$ ▷ Start with complete feature set selection
  for f **in** $F_{selected}$ **do** ▷ For all features in $F_{selected}$
    if $J(X_{F_{selected}\backslash\{f\}}, y, f) < \alpha$ **then** ▷ Check if evaluation is higher than α without f
      $F_{selected} \leftarrow F_{selected}\backslash\{f\}$ ▷ Remove f from the feature set selection
    end if
    if length$(F\backslash F_{selected}) = r$ **then** ▷ Stop if feature removal limit has been reached
      break
    end if
  end for
  return $F_{selected}$
end procedure

- *Sequential search extension*
  The two basic sequential search algorithms forward and backward selection can be altered for better use. An intuitive idea would be to combine the two algorithms, creating an algorithm that first selects features with the evaluation function followed by removing them according to the evaluation function or the other way around. Also there is no restriction on only doing one iteration of both forward and backward selection, giving rise to "plus l-take away r" selection (PTA, algorithm 5). PTA(l, r) adds l features with forward selection and removes r features with backward selection per iteration, eventually converging to an optimum. These found optima in both sequential search algorithms and possible extensions can converge to local optima. Beam search is an example that also collects suboptimal branches for possible better optima, instead of only keeping the best possible outcome at all times [102].

Algorithm 5 A plus l-take away r sequential search algorithm [102]

```
1: procedure PTA(X, y, F, J, α, l, r)
2:     F_selected ← ∅                                  ▷ Start with empty feature set selection
3:     i ← 0                                           ▷ Initialize feature index
4:     while i < length(F) do                          ▷ Continue while not all features are evaluated
5:         size ← length(F_selected)                   ▷ Update size of feature set selection
6:         for k in [i, length(F)] do                  ▷ Continue for features in F with index > i
7:             if J(X_{F_selected}, y, X_{F_k}) > α then   ▷ Check if evaluation is higher than α with f
8:                 F_selected ← F_selected ∪ F_k       ▷ Add f to the feature set selection
9:             end if
10:            i ← k                                   ▷ Update i as features have been seen
11:            if length(F_selected) − size = l then   ▷ Stop if feature addition limit l has been
    reached
12:                break                               ▷ Break the forward selection
13:            end if
14:        end for
15:        size ← length(F_selected)                   ▷ Update size of feature set selection
16:        for f in F_selected do                      ▷ For features in F_selected
17:            if J(X_{F_selected\{f}}, y, X_f) < α then ▷ Check if evaluation is higher than α without f
18:                F_selected ← F_selected\{f}         ▷ Remove f from the feature set selection
19:            end if
20:            if size − length(F_selected) = r then   ▷ Stop if feature removal limit r has been
    reached
21:                break                               ▷ Break the backward selection
22:            end if
23:        end for
24:    end while
25:    return F_selected
26: end procedure
```

A last example of a sequential search extension is called floating search. Instead of adding and removing a set number of features as in PTA, floating search continues to add and remove features until the best subset is found (Algorithm 6). Since feature relevance and redundancy changes for every new subset, all features are continuously evaluated to find out if they must be added to or removed from the subset. This way an optimal subset can be found [102].

- *Stochastic search*

Stochastic search uses random mutations in a candidate subset to achieve an optimal subset. One interesting approach of stochastic search is simulated annealing (SA), a search algorithm based on the cooling down of physical matter [109]. This search algorithm tries new feature subsets constantly until the temperature is 'cooled down' or an optimal solution is found (Algorithm 7). This is done by choosing a new feature every time regardless of whether it is in the subset, or not. If adding or removing the new feature improves the performance of the subset, it will be added or removed. If it worsens the subset, it may be added or removed depending on the quality of deterioration and the temperature. The temperature is lowered after a number iterations of adding or removing features, until it is completely 'cooled down' and the final subset had been made [102]. A second example of a stochastic search algorithm is a genetic algorithm, that collects multiple subsets and mutates them in an evolutionary way [110] (Subsection 4.2.3).

Algorithm 6 A floating search algorithm [102]

procedure FLOATINGSEARCHSELECTION(X, y, F, J, α)
  $F_{selected} \leftarrow \emptyset$ ▷ Start with empty feature set
  while $F_{selected}$ **changes do** ▷ Continue while the feature set selection changes
    for f **in** F **do** ▷ For features in F
      if $f \notin F_{selected}$ **and** $J(X_{F_{selected}}, y, X_f) > \alpha$ **then** ▷ Check if evaluation is higher than α with feature f
        $F_{selected} \leftarrow F_{selected} \cup \{f\}$ ▷ Add f to the feature set selection
      end if
    end for
    for f **in** $F_{selected}$ **do** ▷ For features in $F_{selected}$
      if $J(X_{F_{selected}\backslash\{f\}}, y, X_f) < \alpha$ **then** ▷ Check if evaluation is higher than α without feature f
        $F_{selected} \leftarrow F_{selected}\backslash f$ ▷ Remove f from the feature set selection
      end if
    end for
  end while
  return $F_{selected}$
end procedure

Algorithm 7 Simulated Annealing search algorithm [102]

procedure SA($X, y, F, J, T_0, T_1, m, v$)
  $F_{selected} \leftarrow$ randomSubset(F) ▷ Start with random feature subset
  $T \leftarrow T_0$ ▷ Initialize simulation temperature T with T_0
  while $T_0 \geq T_1$ **do** ▷ While temperature is not cold enough (T_1), yet
    $i \leftarrow 0$ ▷ Initialize counter for not improvements
    while $i < m$ **do** ▷ While not improvements is lower than maximum m
      $f \leftarrow$ randomFeature(F) ▷ Select random feature f
      if $f \in F_{selected}$ **then** ▷ Check if f is in feature set selection
        $F_{candidate} \leftarrow F_{selected}\backslash\{f\}$ ▷ create candidate feature subset selection
        $\Delta = J(X_{F_{selected}\backslash\{f\}}, y, X_f)$ ▷ Calculate evaluation difference without feature f
      else ▷ Check if f is not in feature set selection
        $F_{candidate} \leftarrow F_{selected} \cup \{f\}$ ▷ create candidate feature subset selection
        $\Delta = J(X_{F_{selected}}, y, X_f)$ ▷ Calculate evaluation difference with feature f
      end if
      if $\Delta > 0$ **then** ▷ Check if change is positive
        $F_{selected} \leftarrow F_{candidate}$ ▷ Use candidate as feature subset selection
      else
        $r \leftarrow randomReal(0, 1)$ ▷ Get a random value $r \in [0, 1]$
        if $r < \exp(-\Delta/T)$ **then** ▷ Check if a random change should occur
          $F_{selected} \leftarrow F_{candidate}$ ▷ Use candidate as feature subset selection
        end if
        $i \leftarrow i + 1$ ▷ Increment counter for no improvement
      end if
    end while
    $T \leftarrow T \times v$ ▷ Lower the temperature with $v \in [0, 1]$
  end while
  return $F_{selected}$
end procedure

The major advantage of using wrapper methods is that they also take into account possible

dependencies between features. The computation time of wrapper methods usually is higher than of filter methods, but still relatively short as it should always converge to an optimum. These optima can be local however, so the result may not be the optimal, due to its greedy character. The stochastic search algorithms provide a way to find a global optimum at the cost of being more computationally intensive. Also, these methods are dependent on the evaluation function and are known to be prone for over fitting [102, 96].

Embedded Methods

In both filter and wrapper methods, machine learning plays little to no role in selecting features. Filter methods do not use learning at all and a wrapper method can only use machine learning for evaluating feature subsets. Several machine learning algorithms contain attributes that can be used directly to select or eliminate features from the ideal feature set selection, for example the coefficients in linear SVM. Embedded methods combine feature selection with a machine learning algorithm M, in contrast to wrapper methods in which these are separated [111]. Usually this combination involves using the weights given to features by machine learning algorithms [112].

Embedded methods usually solve feature selection by using one of two possible approaches. The first approach is based on contribution relaxation minimization and the second approach is based on convex function minimization. Since the theoretical approach of embedded methods can be very computationally intensive, approximations of the minimization problems are given, showing example embedded methods [111].

- *Contribution relaxation minimization*

  Contribution relaxation is based on giving every feature f a contribution factor $\sigma_f \in [0, 1]$. This contribution factor shows the contribution of a feature to the preferred outcome. The final goal is to select a subset of features, though, and not use all features with a contribution factor. To achieve that a minimization function is used in which all contribution factors become $\sigma_f \in \{0, 1\}$. With contribution factors in $\{0, 1\}$, a subset with all features having $\sigma_f = 1$ can be selected as the ideal subset [111].

  The minimization can be implemented with an embedded method. This method would however be computationally intensive and therefore approximations are used. The most used approximation is similar to the filter methods, ranking each feature and choosing features based on rank (Algorithm 8). This embedded method is also called a forward selection method. A machine learning algorithm M assigns a weight to each feature, which can be used as a rank (Algorithm 2). The difference between a ranking method R of the filter methods and the weights of a machine learning algorithm M is that M also takes into account the dependency between features [111].

Algorithm 8 An embedded forward selection algorithm [111]

```
1: procedure EmbeddedForwardSelection(X, y, F, M, α)
2:     F_selected ← ∅                          ▷ Start with empty feature set
3:     W ← M(X, y)                             ▷ Extract the weights for every feature
4:     for f in F do                           ▷ For all features in F
5:         if W_f > α then                     ▷ Check if weight is higher than threshold α
6:             F_selected ← F_selected ∪ {f}   ▷ Add f to selected features
7:         end if
8:     end for
9:     return F_selected
10: end procedure
```

- *Convex function minimization*

The convex function minimization focuses on the trade-off between prediction quality and feature subset size. Convex function minimization combines a loss function, used for quality measurements, with a penalty term for the number of features used. This results in checking for which features the quality increase is lower than the penalty term, so it should be removed. There are many different loss functions that can be used for this case, all with their own advantages and disadvantages [111].

For approximation of this minimization, an embedded method closely related to the backward selection wrapper method can be used called Recursive Backward Elimination (RBE, Algorithm 9). In RBE a machine learning method gives weights for every feature. The difference with backward selection is that no predefined order selects the next feature that may be removed. The feature with the lowest weight is then compared with a threshold and removed if too low for the threshold. Since weights of a machine learning algorithm change when features are removed, this should be done recursively until no feature can be found any more with a weight lower than the threshold [111].

Algorithm 9 An embedded backward elimination algorithm [111]

procedure RecursiveBackwardElimination(X, y, F, M, α)
  $F_{selected} \leftarrow F$ ▷ Start with complete feature set selection
  while $F_{selected}$ **changes do** ▷ While the feature set selection changes
    $W_{F_{selected}} \leftarrow M(X_{F_{selected}}, y)$ ▷ Extract the weights for every feature f
    $f_{min} \leftarrow f \in F_{selected}$ **with** $W_f = \min(W_{F_{selected}})$ ▷ Find f with the lowest weight
    if $W_{f_{min}} < \alpha$ **then** ▷ Check if weight is lower than threshold α
      $F_{selected} \leftarrow F_{selected} \backslash \{f\}$ ▷ Remove f from selected features
    end if
  end while
  return $F_{selected}$
end procedure

The weights given by machine learning are different for every algorithm. A first and most obvious example of an algorithm giving weights are algorithms based on linear support vector machines (linear SVM, from now on called SVM). The coefficients given by SVM give the features a contribution factor in the outcome [113, 114, 115, 116]. A second example would be using decision trees and random forests. Decision trees use features to reduce the entropy between a set by splitting it in two subsets with a threshold and a feature. These splitting features can be used as a subset to create an effective feature selection outcome [117, 118, 97]. To overcome an over fitting problem, commonly occurring in decision trees, random forests can be used instead and the best splitting features can be used [119].

4.2.3 Automated Machine Learning

Before automated machine learning (autoML) existed, a dataset was mined by hand. First a preprocessing algorithm was chosen and used to prepare the data. Next a (machine learning) algorithm was chosen to mine the desired results out of the data. At last the hyper-parameters of the chosen algorithm were tuned to optimize the desired results. These three steps are vastly different and significant issues arise when combining these. Several ideas arose to combine the steps, called Combined Algorithm Selection and Hyperparameter optimization (CASH) [120]. After some time, when preprocessing was added in the mix as well, the name autoML was being used [121]. The first autoML approach tool was published as `Auto-WEKA` that focused on classification methods, spanning 2 ensemble methods, 10 meta-methods, 27 base classifiers and their hyperparameter settings [120]. An upgrade was published that added regression and parallelism [122].

As explained briefly before, to go from data to results several steps must be taken: *Preprocessing*, *algorithm selection* and *hyperparameter optimization*. This sequencing is called a machine learning pipeline. Such a pipeline can consists of zero, one or multiple preprocessing steps for data preparation, can be one of many different machine learning algorithms which on their part have wide ranges for multiple hyper-parameters. The explosion of possible pipelines makes it hard to choose the right one. Knowing successful combinations is useful, however every data set has different features that ask for different pipelines [121]. AutoML tries to find the best machine learning pipelines to compute which algorithms must be selected, combined with tuning the hyper-parameters and preprocessing.

Genetic programming is an example of an automated machine learning approach. It is based on the evolution principles as described by Darwin. A genetic algorithm starts with a problem and searches for a solution, for example the best machine learning pipeline to predict the output using the available data. First it tries out a set of possible solutions to the problem. Then it evolves these possible solutions to find a better solution. This evolution is done by first evaluating them and selecting the best ones to continue to the next generation. Then both crossovers between and mutations on possible solutions are performed. After that again evaluation and selection, followed by crossovers and mutations, take place a number of times until a certain quality is found, time has run out or another ending condition has been met [123].

Several approaches for algorithm selection are available, such as random search, grid search or Bayesian optimisation. Hyperparameter optimization has challenges on his own to find the right ones and there are many different approaches to tackle preprocessing. The concept of meta-learning is briefly explained in more detail. Aside from that, a tool that makes use of AutoML, *TPOT*, is also discussed.

Meta-Learning

Machine learning algorithms show different behaviour for different tasks. Meta-learning tries to find out how their performance changes between those tasks (Figure 4.1). It tries to link the datasets with a suitable algorithm for high performance and tries to improve this performance by fine-tuning its hyper-parameters. In combination with the machine learning pipelines, meta-learning tries to find the best ones available. Since not only algorithms must be selected, but also hyper-parameters must be optimized and preprocessing must be done, the time and space needed for meta-learning causes a combinatorial explosion. This can be lessened by stopping the investigation of bad pipelines and limiting the range of hyper-parameters, machine learning- and preprocessing algorithms as much as possible.

Features of the meta-learning phenomenon are used to predict the performance. There are three meta-features categories (Appendix G.1). The first category is a combination of simple, statistical and information-theoretic computations of a dataset. They can be a basic feature of the data set, as well as a value after a statistical computation or a specific theoretical value. The second meta-feature category can be called *landmarks*. Landmarks give the performance of algorithms, how well they are doing with the given data set. The last meta-feature category is model-based. Specific characteristics of the used model can be used as meta-features as well [124, 125].

For using those meta-features in picking the best machine learning algorithm meta-learners can be used. Meta-learners are algorithms that choose the best machine learning algorithm, using the knowledge available in meta-features. There are four ways of doing that. The first is plainly choosing the best algorithm in the set, this choice speeds up the process but is prone to being a bad choice. Second a subset of good algorithms can be chosen, which is slower, but has a higher chance to give a good outcome. Thirdly the algorithms can be ranked, which makes the chance of picking a good algorithm quicker starting at the top. Fourth is to use estimations of performance which gives information expectations over all algorithms [126].

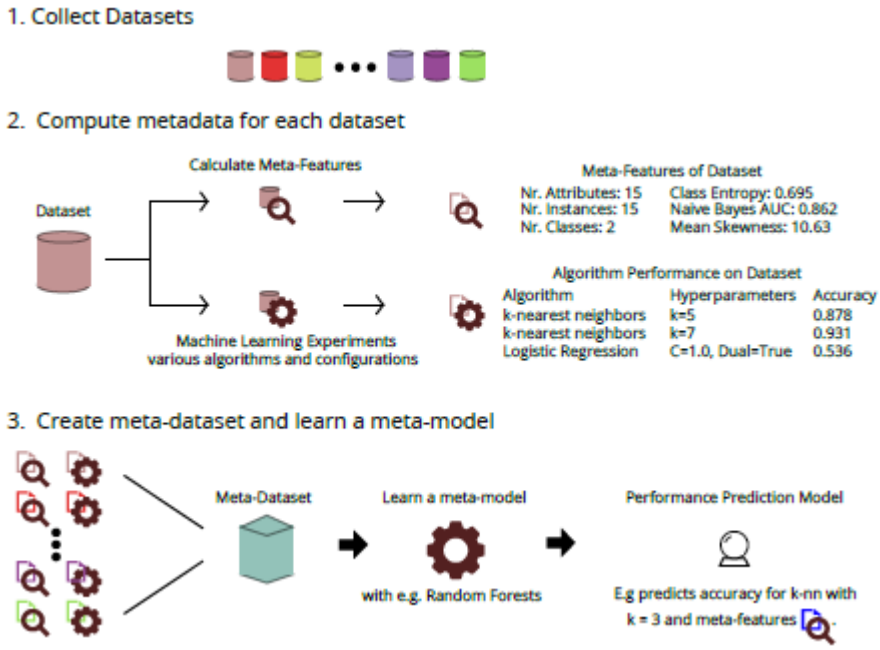


Figure 4.1: A layout of how meta-learning works. 1. Data sets are collected. 2. Meta-data is computed for each dataset. 3. A meta-dataset is created and a meta-model is learned [121].

Tree-based Pipeline Optimization Tool

Tree-based Pipeline Optimization Tool (TPOT) is a tool that implements autoML. It uses machine learning pipelines and genetic programming to find the best solution for every data set. TPOT makes use of this genetic programming with using the machine learning pipelines in a tree (Figure 4.2). *TPOT* consists of preprocessing and machine learning algorithms, that form the backbone of the pipelines (Table 4.2). Their hyper-parameters are the variables. *TPOT* makes mostly use of the machine learning and preprocessing algorithms of *scikit-learn* from *Python* in which it is also written, but also several additional algorithms are present (e.g. a hot encoding algorithm).

TPOT has three different types of mutations within one pipeline. The first one is insertion, inserting a primitive somewhere in the tree. An example would be the insertion of an additional preprocessing algorithm. The second one is replacement, which replaces a random terminal. It can for example change a binary hyper parameter from true to false. The third one is shrinking. A primitive is replaced by a terminal. For example a preprocessing step can be replaced by just raw data. This different mutations can all be seen visually (Figure 4.3)

TPOT also focuses on mutations between two pipelines by means of crossovers. Between two pipelines, sub-trees and primitives can be changed, given that the both pipelines remain valid (Figure 4.4). Every time a crossover is performed, two separate pipelines are used and changed, creating two new ones.

Considering the capabilities from *TPOT* to cope with challenges in biomedical data, several methods are available. It has several different normalisation/standardisation scalers (StandardScaler, RobustScaler, MinMaxScaler) to tackle feature heterogeneity between different data sets and errors. It also has some feature selection operators to tackle errors (VarianceThreshold, SelectKBest, SelectPercentile). Wrapper methods however are not included, which may perform better in some situations. At last, if missing values are present, it imputes the median for those values before starting the evolutionary algorithm. However, it does not have any possibilities to handle non-numeric data. If non-numeric data is present, the user first needs to preprocess himself.

A big variety of algorithms are present for TPOT, as well as numerous hyper parameter values

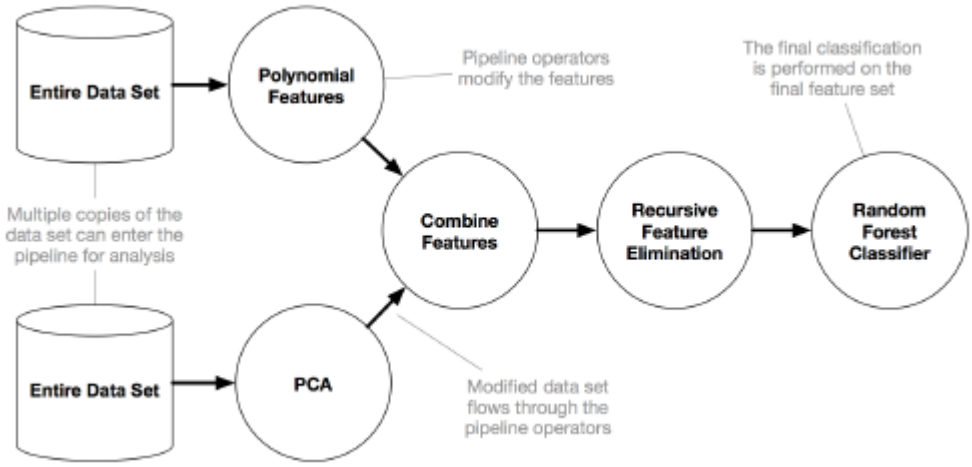


Figure 4.2: An example of a machine learning pipeline in *TPOT*. It only shows the primitive algorithms and not hyper parameter terminals. At the root is the machine learning algorithm [121].

Table 4.2: All *TPOT* Algorithms used for the *TPOT* classifier function, their *scikit-learn* class name is given. Not only the algorithms differ, but also the hyper-parameters within these algorithms.

Algorithm type	**Specification**	**Algorithms**
Classifier	Naive Bayes	GaussianNB, BernoulliNB, MultinomialNB
	Decision Tree	DecisionTree, ExtraTrees, RandomForest, GradientBoosting
	Nearest Neighbour	KNeighbours
	Support Vector Machines	LinearSVC
	LogisticRegression	Logistic Regression
Preprocessors	Scaler	Binarizer, MaxAbsScaler, MinMaxScaler, Normalizer, RobustScaler, StandardScaler
	Feature reduction	PCA, FastICA, RBFSampler, Nystroem, FeatureAgglomeration
	Feature Modifier	Polynomial, OneHotEncoder,ZeroCount
	Feature Selectors	SelectFwe, SelectPercentile, VarianceThreshold, RFE, SelectFromModel

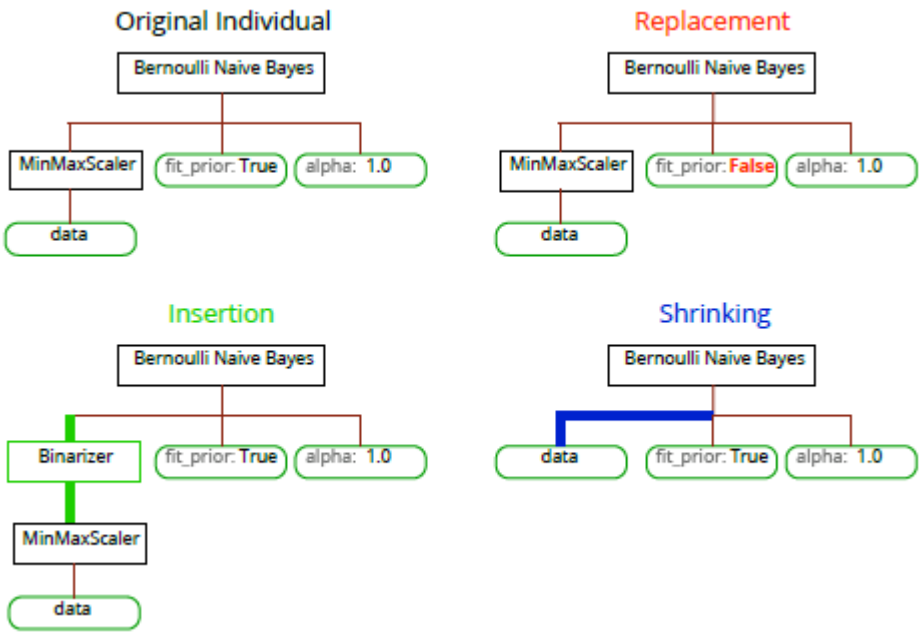


Figure 4.3: Examples of the three mutations in the *TPOT* algorithm: insertion, replacement and shrinking [121].

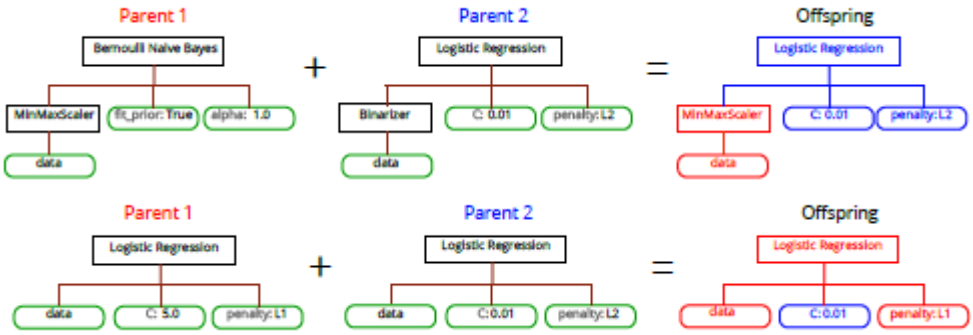


Figure 4.4: An example of a *TPOT* crossover [121].

for those algorithms. This is both an advantage as an disadvantage as probably most algorithms and hyper parameter combinations are not useful as an outcome. Since TPOT randomly chooses mutations, possibly better algorithms could be chosen with less random choices. For example meta-features could be used to improve the algorithm selection.

4.3 Hypotheses

The main goal of this chapter is to find more insight in feature selection and add that insight to the final framework. Several hypotheses are made before designing experiments to test them and they are discussed here:

1. *Optimum in number of selected features*
   An optimum should be found in the selected features. Whereas using too many features may cause issues for a machine learning algorithms to extract the correct information from them, preserving too few features may result in lost information. Previous research scaled down the number of features from 400 to 96 [127], 500 to 13 and for even bigger datasets from 5000-100000 features to between 200 and 900 relevant features [128]. Interestingly, the number of relevant features was not related to the original feature set size, indicating the optimum to be an absolute number. Therefore:

   H1: The value of this absolute optimum is estimated to be lower than 1000, as no study before ever used a higher number of relevant features.

2. *Best feature selection methods*
   Three types of feature selection methods are discussed, all of them having their own advantages. Out of these three, the embedded methods are expected to outperform the others. Whereas filter methods are quick and easy, their efficiency should be lower than the other methods. Wrapper methods are closer to trying to find the optimal solution in an exhaustive search way. Therefore they are expected to perform best in feature selection at the cost of computation efficiency. At last the embedded methods are expected to be in between of the two. The embedded methods use machine learning methods that are fairly quick and easy to understand. Those machine learning methods also use relative contributions between features. This would mean that embedded methods are quicker and easier than wrapper methods and more efficient than filter methods and therefore combining the best of both.

   H2: The embedded methods are expected to outperform filter methods in quality and wrapper methods in computation time efficiency.

3. *TPOT feature selection integration*
   Integration of feature selection in *TPOT* for the framework can be done in different ways. Two ways are thought to be a beneficial addition to the tool. First expanding the number of feature selection methods should improve *TPOT* in finding better pipelines. Secondly having *TPOT* start with an initial pool of pipelines including feature selection methods should give it a head-start and therefore be quicker and better and finding good pipelines.

   H3: Both additions (warm starting with feature selection methods and additional feature selection methods) are expected to improve TPOT with feature selection.

4.4 Methods

For empirical evaluation the collection of feature selection methods experiments are done. The quality of the feature selection is tested by defining a definition of quality first. After that an exploration of feature selection is done with filter methods, followed by comparisons of multiple feature selection methods. The four example datasets (subsection 6.2.1) were used as example datasets for the experiments.

4.4.1 Feature Selection Quality

To find out the quality of the feature selection, multiple machine learning algorithms are used [129]. Classification of the datasets can be done with several machine learning algorithms and validation and tests scores show how well the data can be classified after feature selection. The quality will be described with the accuracy of machine learning algorithms: the number of right classifications divided by the total number of classifications. Five different machine learning classifiers are used from *scikit-learn*: logistic regression, decision trees, nearest neighbour, support vector machines and Naive Bayes. Better feature selection algorithms have relatively higher accuracy, as they are better at preserving the right features. Since over fitting sometimes happens when fitting data in a machine learning algorithm, both a validation and a test score is computed for the accuracy. For every experiment a training set and a test set is created, making the test set 20% of the complete dataset. A validation score is computed by using the "leave one out" technique on the training set and a test score is computed by testing the classification score of the test set. Since the samples are not evenly distributed over the classes the precision, recall and F1 scores are also computed to find potential bias in the result.

In some cases the quality of the machine learning algorithm lacks a factor of feature subset size. The standard score in those cases is not sufficient enough to evaluate the result. To incorporate the quality of feature selection a new modified quality term *FS_score* is created (Equation 4.4.1). In *FS_score* a modified version of the original score, in this thesis being accuracy, is given by first multiplying it by a factor dependent on the number of features. This factor consists of a constant β, a value in range $[0, 1]$ which can be chosen for the influence of the value that represent the number of features. In practice $\beta > 0.95$ so the reduction does not have too much influence on the score. An example plot of how the correction factor changes is given (Figure 4.5).

$$\text{FS\_score} = \text{score} \times \beta^{\#\text{features}}$$

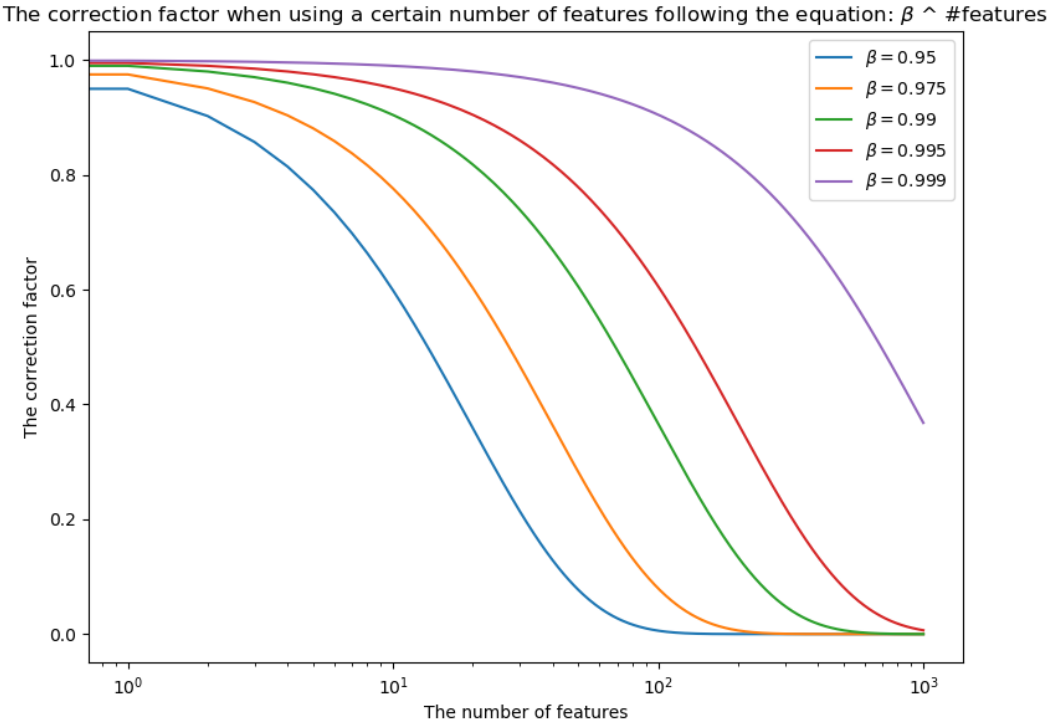


Figure 4.5: The correction factor for the *FS_score* from Equation 4.4.1. Every graph corresponds to a different value of β.

The factor #features is an absolute count, instead of a relative count (for example a percentage of features). The reason for using an absolute count can be found in the goal of the data analysis. In data analysis the number of features that can be relevant are very limited. Useful results need relations between the input and the output, and these relations need to be as simple as possible. If the number of features is relative to the total number of features, the input size can change significantly. Take for example the Micro-organisms and the RSCTC datasets provided for this study (Subsection 6.2.1). 2% of the total features would be 26 and 1100 features for these datasets respectively. Whereas relations between 26 features and the output is possible to understand, relations between 1100 features and the output is much harder. Taking an absolute number of features allows for a more active choice on the input size.

To put the impact of the *FS_score* in perspective, an example figure is made how the outcome is computed (Figure 4.6). This figure shows the impact on the performance of a method when using a certain number of features. After using a correction factor, an optimum is created for which the number of features is important, the old optimal score did not include the number of features. The example (with $\beta = 0.99$) seems to give a good indication of the desired outcome. For every 10 features, the *FS_score* is reduced by about 10% (Figure 4.5), which meant for the example filter method that for 50 features an optimum was reached (Figure 4.6). Because of this empirically found desired trade-off, $\beta = 0.99$ will be chosen in this project when using *FS_score*.

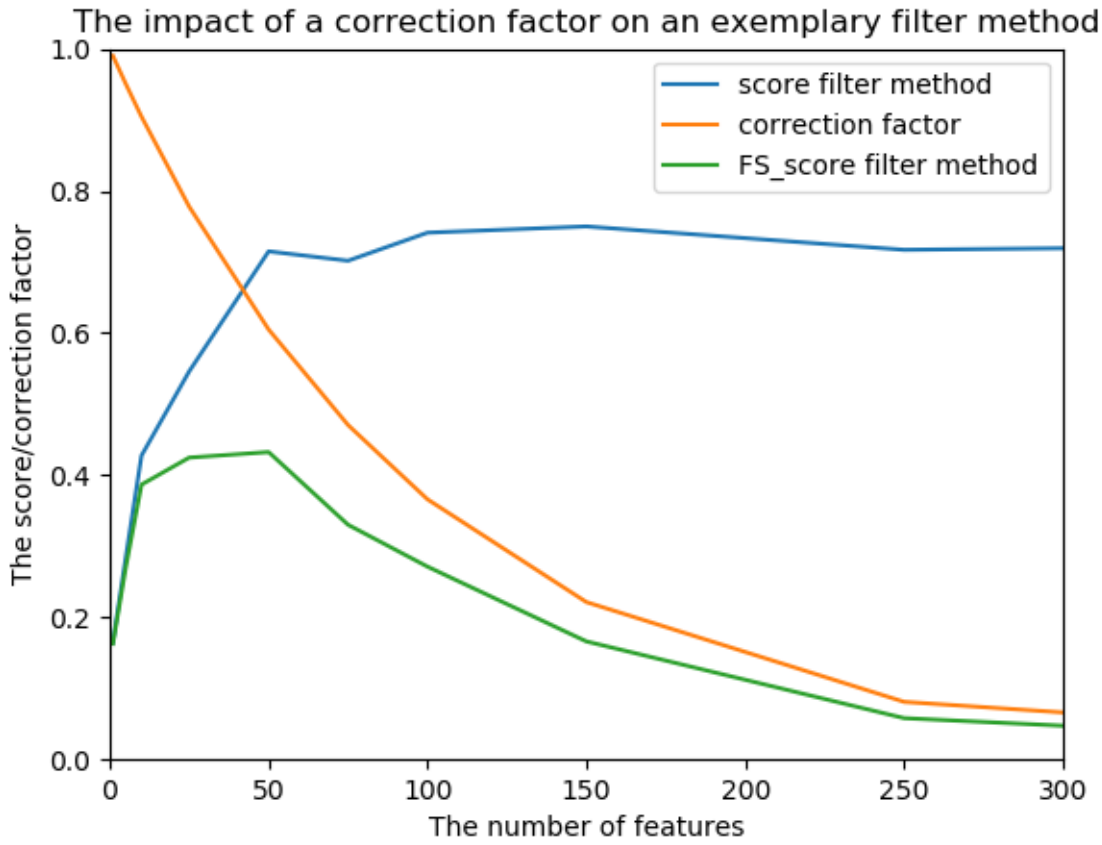


Figure 4.6: An example of the impact of the correction factor on the score, in this case accuracy. The shown correction factor uses $\beta = 0.99$. On the x-axis the number of features is shown and on the y-axis the value for the original accuracy, the correction factor and the *FS_score*.

4.4.2 Feature Selection Exploration

The first set of experiments is used to explore a combination of the basic filter method combined with the quality measurements. The basic filter method algorithm selecting the top n features (Algorithm 1) is used. The changing variables are the dataset, the ranking method, the feature preservation values and the accuracy computation methods (Table 4.3). The range of chosen feature preservation values comes from both its ability to show impact of separate features (more impact from fewer features) and the relevance of keeping that number of features (irrelevant feature selection when having more than 1000 features). All of this together results in a total of $4(\text{datasets}) \times 2(\text{ranking methods}) \times 11(\text{top } n \text{ features}) \times 5(\text{accuracy computation methods}) = 440$ experiments. These experiments are visualized in eight plots, one plot for every combination of data set and ranking method. These plots then show the change in quality for different number of preserved features.

4.4.3 Feature Selection Algorithms Evaluation

The second set of experiments compares the feature selection methods in both feature selection and quality. For this experiment the four datasets are used with the logistic regression quality measurement, since the logistic regression gave the most consistent result of the five machine learning algorithms, showing the smallest variation across the datasets. The average performance for

Table 4.3: The four meta-parameters with their possible values in the first experiment.

Variable	**Description**	**Values**
Dataset	The datasets used (subsection 4.2.1)	Psoriasis RSCTC Arcene Micro-Organisms
Ranking method	The method used for ranking the features (Subsection 4.2.2)	– T-test (*SciPy*) Mutual Information (*scikit-learn*)
Feature preservation values	The fixed size of the feature subset after feature selection	– 1, 5, 10, 25, 50, 75, 100, 150, 250, 500, 1000
Accuracy computation	The machine learning algorithms used to compute accuracy (subsection 4.4.1)	– Naive Bayes Logistic Regression Support Vector Machine Decision Tree Nearest Neighbours

all datasets is also computed for clarification purposes. An overview of feature selection methods is made (Table 4.4) and all separate meta-parameters are explained here.

- *Filter methods*
  The ranking methods T-test and Mutual Information were used as those were both explored in the first experiment. The thresholds of 50, 100 and 150 features were chosen as a compromise between the desired number of features (as few as possible) and the quality of the result (as explored in the first experiment).

- *Wrapper methods*
  The order of the features used for feature selection with wrapper methods was put at random (no special ordering) or with mutual information (special ordering). The evaluation was chosen to be Naive Bayes as it is simple and not much affected by conditional dependency. The aforementioned threshold value for wrapper methods α was chosen to be 0.01 and 0.001 with a maximum number of features in mind. If $\alpha = 0.01$, a feature will only be added when it raises the quality by 0.01. The accuracy can be at most 1, so at most 100 features can be added, even though the final number of features would be much less. $\alpha = 0.001$ was taken for a solution with a higher accuracy in mind. At last, for the PTA method, for l and r the combinations $[l, r] = [5, 2]$ and $[l, r] = [20, 10]$ were chosen. The ratio was picked close to $2:1$ as a compromise between number of features added and removed, 50% being a reasonable ratio to still keep a decent number of features. The size of l and r were different in a factor 4 or 5 to find out if this size difference would make a change.

- *Embedded methods*
  The machine learning algorithms that were tested were Support Vector Machines (SVM) and RandomForests (rf), as both of them have a measurement of feature importance. The number of features preserved was put on 50, 100 and 150 being the same as for the filter methods.

Spectra are made with the results of the experiment that show the performance of all different combinations, showing the accuracy, precision, recall and F1-score. On top of that the computation time is computed and shown per algorithm, as well, in a separate bar chart. At last a combination of the earlier proposed *FS_score* and the computation time is shown as well for $\alpha = 0.99$ to show the relation between computation time on one hand and the *FS_score* on the other hand. Several

Table 4.4: The methods that are evaluated in the second experiment setup.

Type	Method	Parameters
Filter methods	Basic Filter Methods (Algorithm 1)	- Rank: T-test, Mutual Information - Thresholds: 50, 100, 150 features
Wrapper methods	Forward selection (Algorithm 3)	- Order: Random, Mutual Information - Evaluation: Naive Bayes - Alpha: 0.01, 0.001
	PTA (Algorithm 5)	– - Order: Random, Mutual Information - Evaluation: Naive Bayes - [l, r] = [20, 10], [5, 2] - Alpha: 0.01, 0.001
	Floating search (Algorithm 6)	– - Order: Random, Mutual Information - Evaluation: Naive Bayes - Alpha: 0.01, 0.001
Embedded methods	Forward selection (Algorithm 8)	- Machine Learning: SVM, RandomForests - Threshold: 50, 100, 150 features

discussed algorithms are chosen to be excluded from the evaluation. This choice is based on their poor scalability with regards to computation time and therefore unfit for datasets with this many features. These excluded algorithms are the backwards elimination sequential method, the simulated annealing stochastic search method and the embedded backwards elimination method (Subsection 4.2.2).

4.4.4 TPOT Feature selection integration

TPOT is an effective tool to find the best machine learning pipeline for a certain dataset (Subsection 4.2.3). *TPOT* however has three restrictions on finding the correct pipeline when a high number of features is involved:

1. *Input versus output*
   The goal of feature selection is to directly link a smaller number of features to the output. Whereas *TPOT* has possibilities to find pipelines with feature selection, it gives no advantage to pipelines that actively use feature selection if needed.

2. *Memory/Invalid dataset issues*
   Firstly, not all algorithms are suitable for handling more than 10000 features. The feature agglomeration algorithm, one of the algorithms present in *TPOT*, will create memory issues in those cases. This makes this algorithm unfit for usage. Similarly, the *scikit-learn* multinomial Naive Bayes algorithm cannot be used with negative values. Biomedical data does not have that restriction and therefore possibly gives errors because of this. On top of that, several preprocessing algorithms also change values to be negative.

3. *General insert issue*
   After some testing and consultation it was concluded that inserting new algorithms to a pipeline created issues in *TPOT*. These issues resulted in a very low number of insertion mutations, which caused a suboptimal search for the best machine learning pipeline that includes feature selection methods and insertion of those.

These three initial restrictions must first be tackled for proper use of *TPOT* with feature selection. This resulted in three measures, explained here in corresponding order.

1. *Feature modifier addition to accuracy*
   In the case of feature selection, the possibility is given to change the performance measurement to the earlier proposed *FS_score*. Using this *FS_score* instead of normal accuracy creates an incentive for *TPOT* to prefer feature selection algorithms in the resulting pipeline, due to higher performance.

2. *Algorithm removal*
   For this restriction, simply the feature agglomeration algorithm is removed. This type of feature selection is not essential in biomedical research and therefore removal is not expected to hinder removal. The multinomial Naive Bayes algorithm is removed, as well, with the same reasoning.

3. *Change in insert method*
   This insert issue was removed by rewriting the code for the insert function. This insert function originally attempted to insert algorithms for every possible hyper-parameter as well as at the start of the pipeline. This is replaced by only having insertion at the start of the pipeline This is an improvement because no algorithm in *TPOT* actually has a hyper-parameter that can be changed by a different algorithm in *TPOT*.

On top of these three restrictions and the measures taken to tackle them, two additional restrictions hinder optimization regarding feature selection:

1. *Lack of warm start*
   TPOT has a vast array of machine learning and preprocessing algorithms to find the best possible pipeline. Due to the number of possibilities being very high, a lot of time may be wasted due to searching in wrong directions. For feature selection, a pre-defined selection of pipelines (also known as a warm start) would improve efficiency.

2. *Feature selection possibilities*
   Several filter and embedded methods are present (Subsection 4.2.2). All of these select percentages of feature selection, though. Still a high number of features can be present in the result after using percentages. On top of that, no wrapper methods are available, either.

For both of these optimization restrictions, two additions are made to the *TPOT* code structure for increased functionality with regards to feature selection, in the same order as the restrictions:

1. *Focused feature selection addition*
   The possibility is created to always start the original population with a feature selection algorithm in the feature selection pipeline. Due to this start the expected search for a good feature selection method is bypassed immediately, which should result in more optimized final pipeline.

2. *Alternative feature selection algorithm set*
   The possibility is created to start with an alternative feature selection algorithms set. This selection consists of filter, wrapper and embedded methods and the hyper-parameters are predefined to create an upper bound of 200 features.

The outcome of the first initial measures are trivial: the use of *FS_score* makes *TPOT* search for solutions with fewer features. The removal of algorithms that cause errors and memory issues creates better optimization. Furthermore more effective use of the insert evolution, makes the algorithm make better use of that possibility. The outcome of optimization additions however may not necessarily improve the performance of *TPOT* for feature selection. Therefore the changes for these two solutions are tested with an experiment.

The experiment consists of multiple runs of *TPOT* and all of these steps are also shown in an explanatory table (Table 4.5). All four datasets are tested (Subsection 4.2.1) and the accuracy is changed to the feature sensitive *FS_score* with $\beta = 0.99$ (Equation 4.4.1), as previously discussed

Table 4.5: The experiment details for testing the non-trivial changes in *TPOT*. This experiment is rerun 5 times.

Experiment factors	**Detailed values**	**Remarks**
Datasets	Micro-organisms Arcene RSCTC Psoriasis	High number of features (Subsection 4.2.1)
Performance measurement	One type: *FS_score* - $\beta = 0.99$	Addition of correction factor (Equation 4.4.1)
TPOT input parameters	One set of input values: - max. optimization time = 120 min - max. alg. evaluation time = 10 min - pop. size = 12 - train size = 0.9	Explanation of input values: - The time one run should take (2 hours) - The evaluation time of one pipeline - The number of pipelines in one generation - The number of samples used for training
Pipeline selection	Regular selection Feature selection focused	Possible obligatory addition of a feature selection algorithm
Change in feature selection algorithm set	Regular feature selection algorithm set New feature selection algorithm set	A change between several basic feature selection algorithms to feature selection algorithms designed for at most 200 features preservation

(Subsection 4.4.1). For testing *TPOT* an optimization time of 120 minutes (two hours) was chosen as a reasonable time constraint to run each experiment 5 times, an algorithm was not allowed to run for longer than ten minutes, a population size of 12 was chosen to not be to selective at the start and a training set size of 0.90 which is a general training set size when not many samples are present. Furthermore tests were done for pipeline selection both with and without focused feature selection and for both with and without alternative feature selection algorithms set, as these additions must be tested for their quality. This gives a total of 4(datasets) $\times$ 2(pipeline selection) $\times$ 2(feature selection set) $\times$ 5(experiment reruns) = 16 different experiments.

4.5 Results

The different experiments are all explained in their own subsections. First the results of the minimum feature preservation experiment were shown, followed by the feature selection algorithms evaluation.

4.5.1 Feature Selection Exploration Results

The results were plotted for every dataset separately (Appendix E.1). An average of the four datasets was also created to show the difference between validation and test score (Figure 4.7), between machine learning quality measures (Figure 4.8) and between data sets with ranking methods (Figure 4.9).

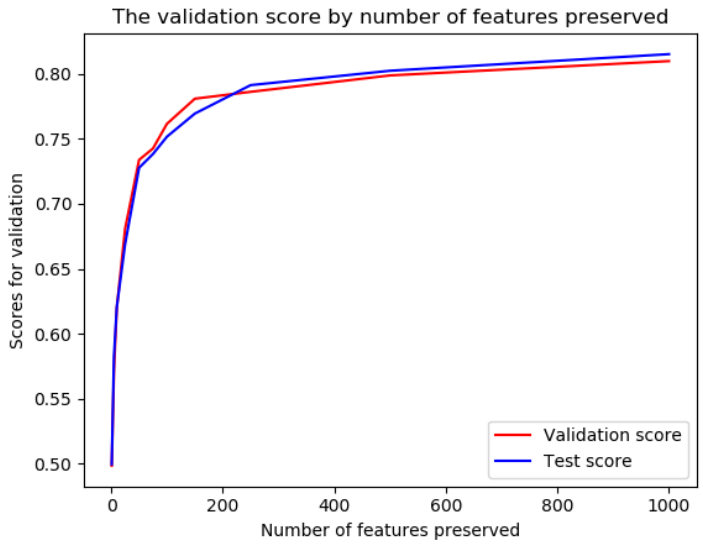


Figure 4.7: The average validation and test scores after averaging the scores for the data sets and ranking methods.

As could be seen (Figure 4.7) the difference between the validation and the test scores was very low. The average almost showed two identical curves which indicates that there was hardly any over fitting present for filter methods. The test error should not be higher than the validation error usually, but in this case the test error had a higher variance. The test error was only one measurement (after splitting the data into training and test set) and the validation error was found with leave-one-out, which should have a much lower variance. Because of the lower variance, the validation score was used in the other two figures (Figures 4.8 and 4.9).

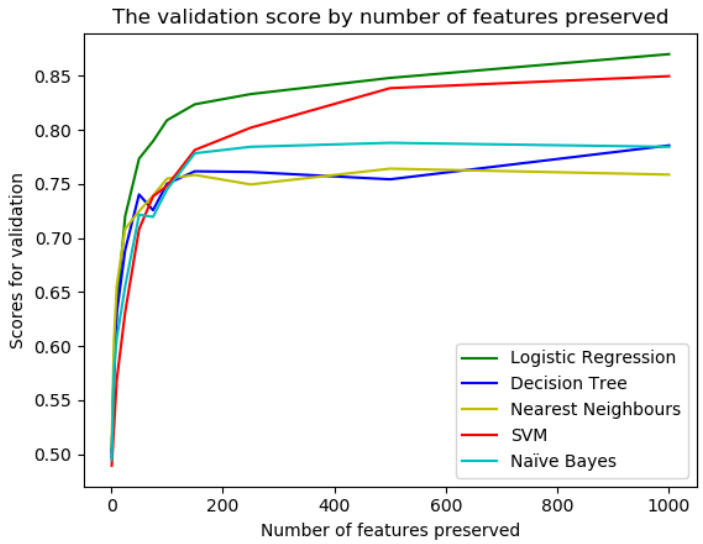


Figure 4.8: The average validation accuracy shown per machine learning quality measurement.

Logistic regression gave the best validation scores taking all data sets combined, followed by SVM. Also, when looking at all eight plots separately (Appendix E.1), logistic regression also showed the most consistent behaviour. All five of the machine learning algorithms showed an 'elbow', an area in the graph for which the score stops growing as much when more features are used. This 'elbow' was located around 100 features with 200 features being the end for almost every 'elbow'.

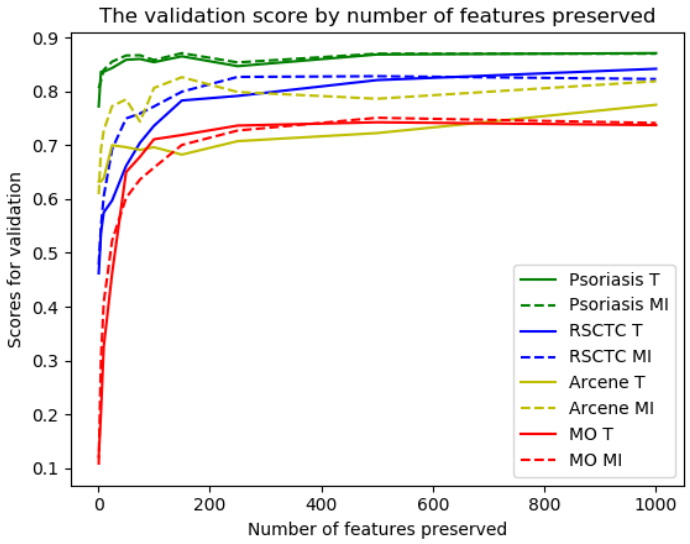


Figure 4.9: The average validation accuracy shown per dataset and rank.

A difference in score quality for the datasets was visible (Figure 4.9). The difference between using Mutual Information and T-test/ANOVA was not, as for only the Arcene dataset there was a significant difference between the two. Therefore it seems that the ranking method type has less influence on the measurement quality. One interesting aspect was that methods using Mutual

Information had a longer computation time than methods using the T-test/ANOVA.

Accuracy was used to test the validation quality of the filter methods. Aside from accuracy the precision, recall and F1-score were also computed to find out whether accuracy is a proper representation of the quality (Figures 4.9 and 4.10 and Appendix E.2). These scores showed that the accuracy is a plausible way to depict quality and it is not needed to use F1-score. Only the Arcene dataset shows some difference between accuracy, precision, recall and the F1-score but these differences would not affect any conclusions.

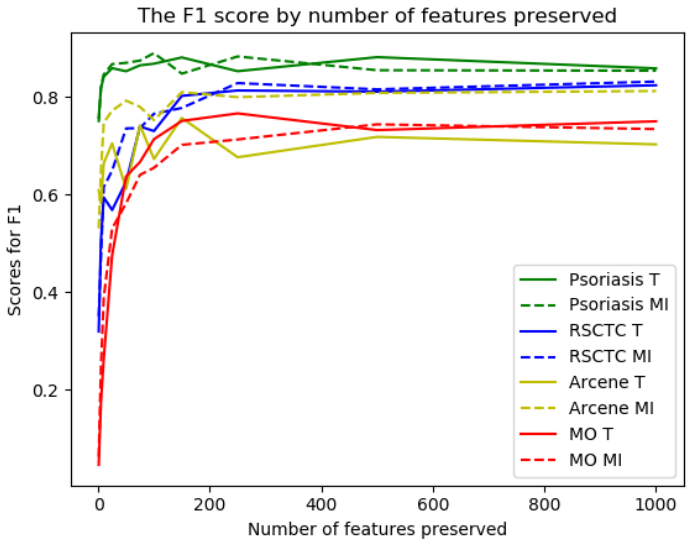


Figure 4.10: The average validation F1-scores shown per dataset and rank.

4.5.2 Feature Selection Algorithms Evaluation Results

Spectra using all basic filter methods and wrapper sequential search methods were created, averaging the accuracy (Figure 4.11), precision (Appendix E.3), recall (Appendix E.3) and F1 score (Figure 4.12) of the four datasets. The figures showed that all wrapper algorithms preserved on average less than 61 features for these settings, whereas the performance seems to average around 75% for all four performance scores. The filter and embedded methods performed worse, with an overall lower performance score than the wrapper methods, even when more features were present. Also, no immediate observations can be made by looking only at the filter and embedded algorithms.

When only looking at the wrapper algorithms, some other observations could be done, as well. Ordering the features before using a wrapper method structurally gave a better result than using a random ordering. Also a threshold of $\alpha = 0.001$ usually resulted in more features and in a higher scores in comparison with a threshold of $\alpha = 0.01$. Comparing the algorithms, the floating search with ordering did best in performance, whereas the other algorithms are performing closer together.

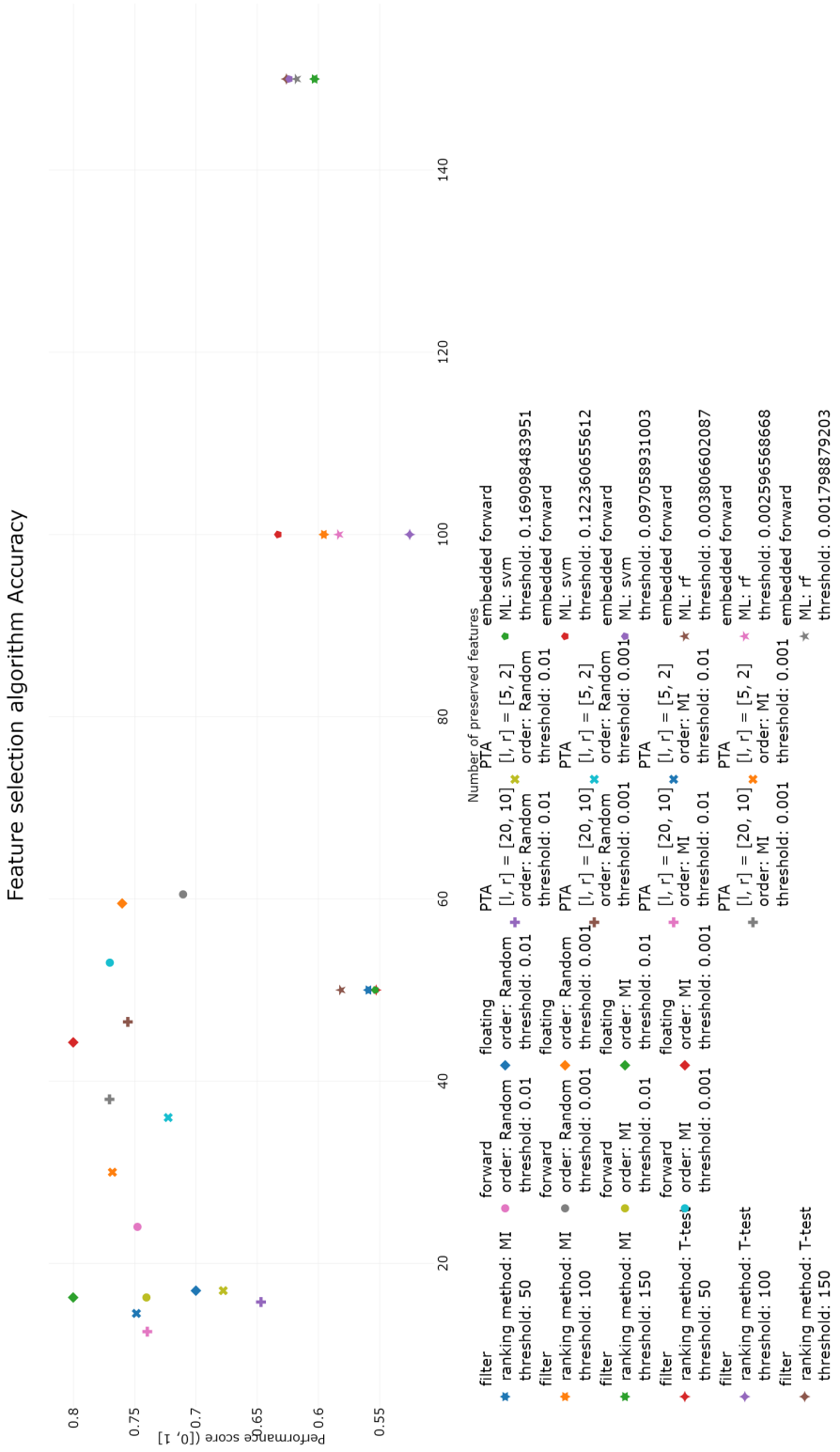


Figure 4.11: The accuracy spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the accuracy of logistic regression. The legend indicates the algorithms (Table 4.4) and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

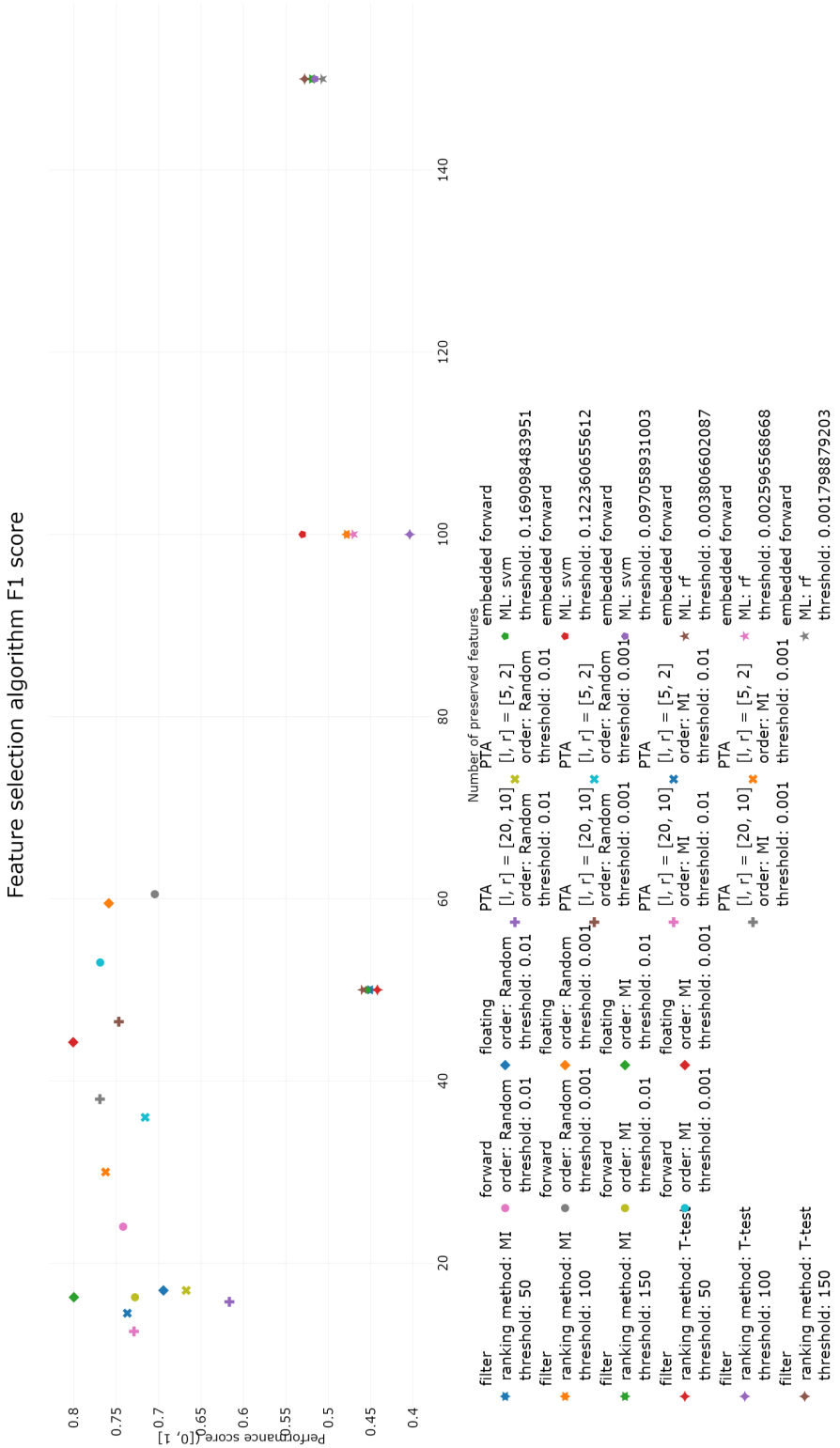


Figure 4.12: The F1 spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the F1 score of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

Aside from the ability to preserve the correct features, also an evaluation of the computation time was made for the different algorithms (Figure 4.13). This chart immediately showed that the wrapper methods took significantly more computation time, with floating search taking the longest time. There was no major difference between forward selection and both PTAs, however there was a difference between filter methods using Mutual Information and T-test. Also for the embedded methods, the SVM based embedded method took significantly less time as well, compared with the random forest embedded method.

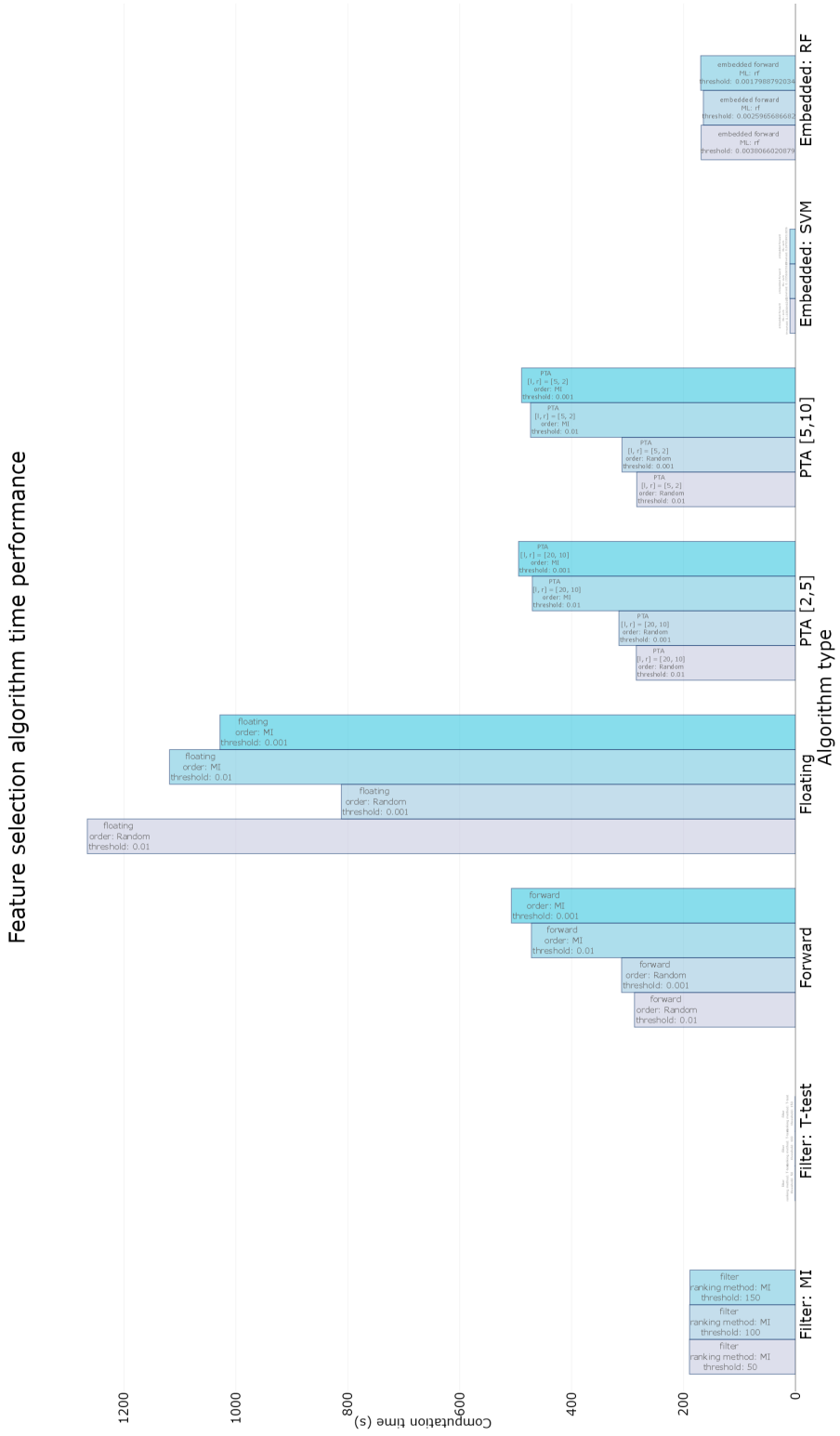


Figure 4.13: A bar chart showing the average computation time of all evaluated algorithms. The x-axis corresponded to the different algorithms, for which the computation time for multiple parameter combinations were given. The y-axis corresponded to the computation time in seconds. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Support Vector Machine (svm), random forest (rf)

At last the relation between the computation time and *FS_score*, the corrected accuracy, are visualized (Figure 4.14). In this chart the trade-of between computation time and quality of a feature selection algorithm is shown by the following statements:

- Filter and embedded methods take the shortest time, but have the worst performance. Floating methods take the longest time, but are not necessarily better than other wrapper methods
- For both filter and embedded methods using T-test and SVM over mutual information and random forests respectively seems to greatly reduce computation time for little to no improvement in *FS_score*.
- The addition of an ordering and the choice of a threshold of $\alpha = 0.01$ over $\alpha = 0.001$ gives better results for all wrapper methods. Using these hyper-parameters however usually add computation time.
- Forward selection and PTA do not show a big difference compared to each other.

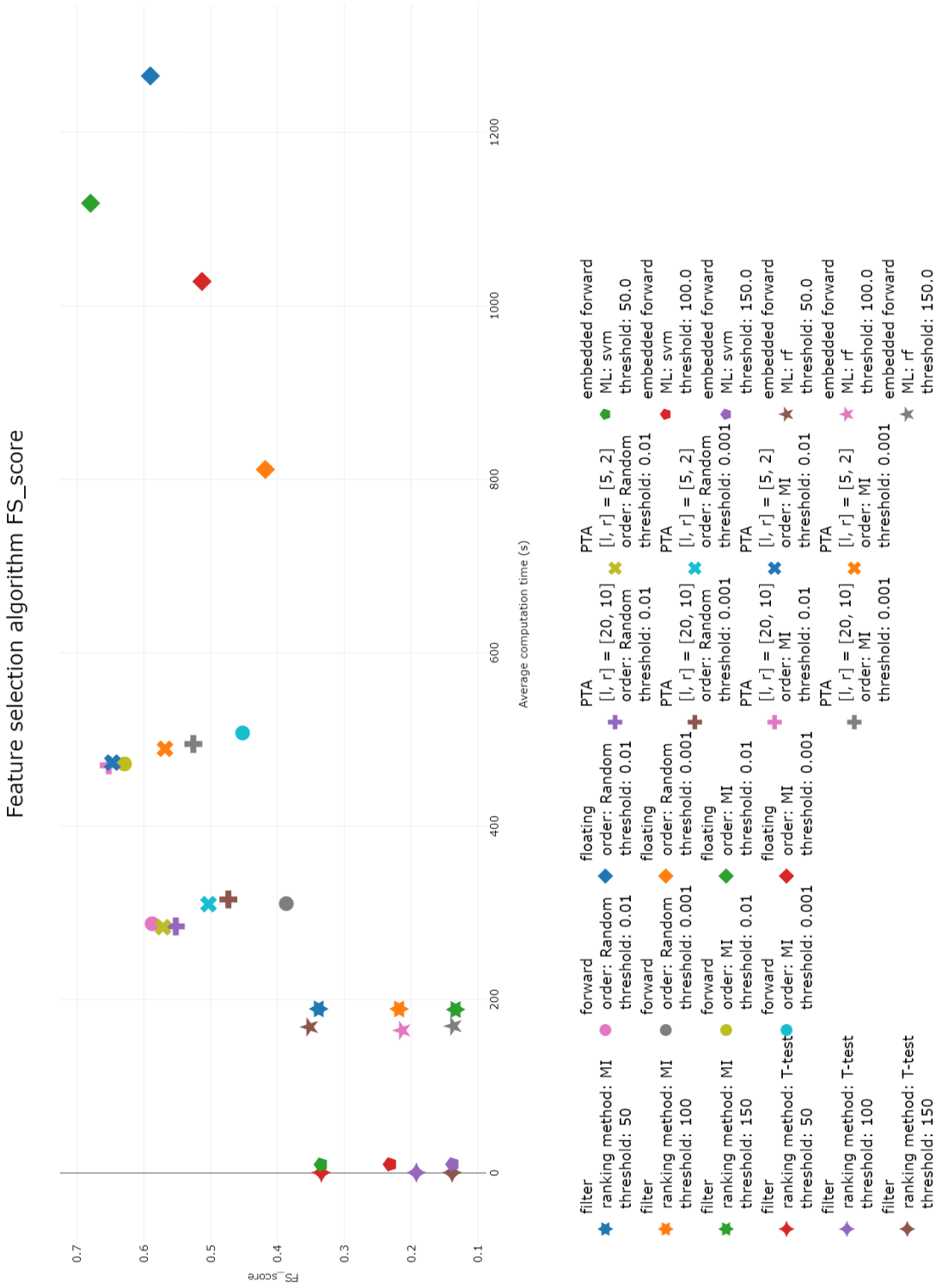


Figure 4.14: The relation between the *FS_score* and the computation time. The x-axis shows the computation time the y-axis shows the *FS_score* score of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

Table 4.6: The performance of the final pipeline for the different types of *TPOT* and the mentioned dataset. The five reruns are averaged into this one result.

Algorithm	selection availability	regular selection		always feature selection	
		regular algorithms	feature selection algorithms	regular algorithms	feature selection algorithms
Datasets	Micro-organisms	0.45	0.71	0.64	0.59
	Arcene	0.46	0.69	0.62	0.69
	RSCTC	0.00	0.22	0.17	0.44
	Psoriasis	0.00	0.24	0.16	0.54

4.5.3 TPOT Feature Selection Integration Results

To visualize the results of the experiment, a table with the final results is made (Table 4.6). The optimization process is also recorded and shown in a plot for the Micro-organisms (Figure 4.15), Arcene (Figure 4.16), RSCTC (Figure 4.17) and Psoriasis (Figure 4.18) dataset. The five experiment reruns are averaged to one complete result.

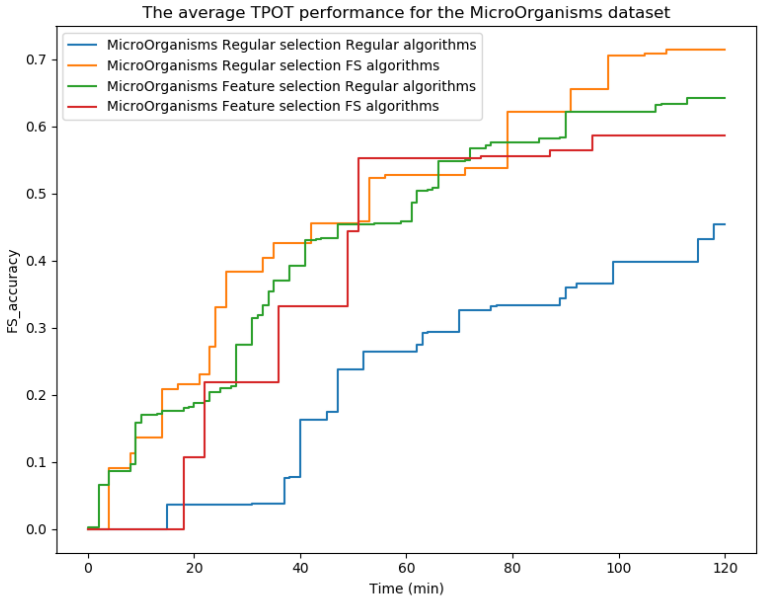


Figure 4.15: The optimization process for the different *TPOT* algorithms for the micro-organisms dataset.

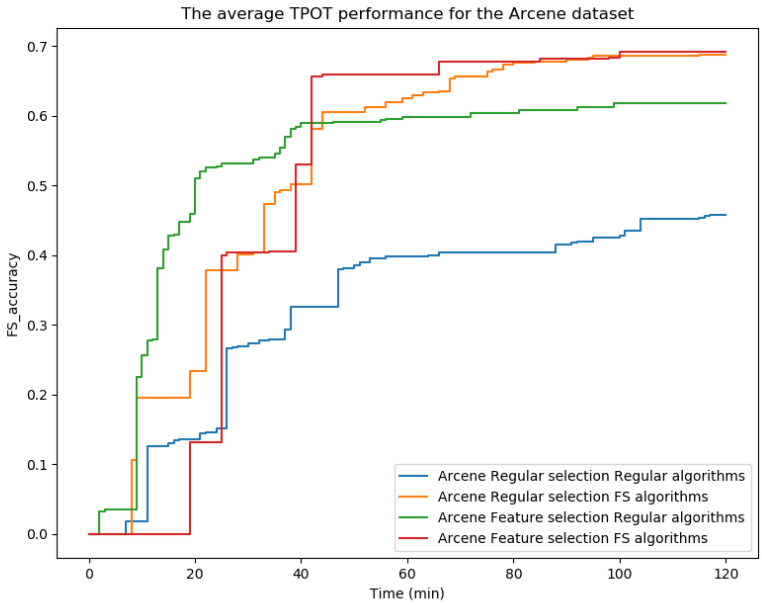


Figure 4.16: The optimization process for the different *TPOT* algorithms for the Arcene dataset.

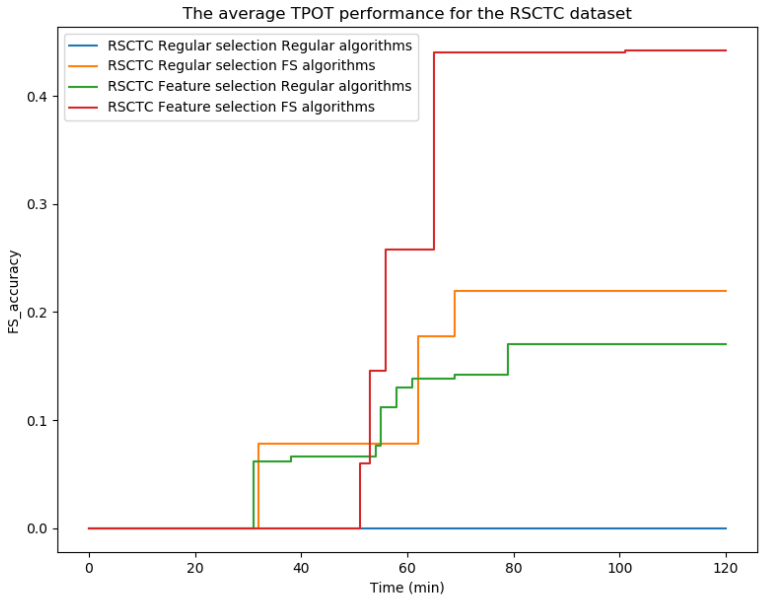


Figure 4.17: The optimization process for the different *TPOT* algorithms for the RSCTC dataset.

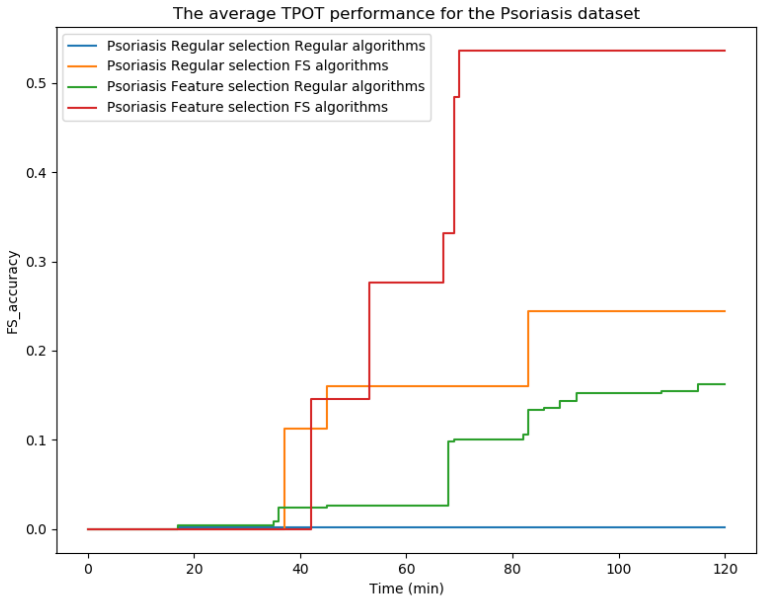


Figure 4.18: The optimization process for the different *TPOT* algorithms for the psoriasis dataset.

The results show that after two hours of running *TPOT*, regular *TPOT* always performs worst. According to the processes the high feature selection algorithm set shows much more big stepwise improvements, accompanied with fewer smaller stepwise improvements. This shows the earlier observed trade-off between computation time and feature selection quality for these specific feature selection algorithms. It takes longer to compute those algorithms, but it also gives a much better result.

The choice for the best *TPOT* algorithm is not conclusive. For high feature selection, all three new possibilities show improved results. The *TPOT* variant that is feature selection focused and has the new feature selection algorithm set performs the best for datasets with a very high number of features (RSCTC and Psoriasis), but the trade-off in computation time seems to hurt the performance for smaller datasets. The algorithm that is only feature selection focused has a steady performance for datasets with a lower number of features, but this performance does not seem to be better than the algorithm with only the new feature selection algorithm set.

4.6 Discussion

There are several works focused on the possibility of feature selection, both with a data analytical [91] as well as a biomedical [19, 20, 92] point of view. These research projects are usually focused on datasets with a low number of features, for which feature selection is less relevant. This research for datasets that have a significantly higher number of at least 1000 features is new, even though newly available datasets become bigger over time due to new and improved techniques of measuring and storing data. With this research a beginning is made on how to approach feature selection on these bigger datasets.

Whereas multiple feature selection methods are discussed, several were not tested due to computation time constraints. The wrapper methods backwards elimination and simulated annealing and the embedded backwards elimination all were too computationally intensive to become relevant for the research. In datasets with fewer features, these methods may be showing better results and could be possible candidates for feature selection.

4.7 Conclusions

After evaluation of the results in the first experiment set-up, it could be concluded that there was not a big difference between using T-test/ANOVA or Mutual Information as a ranking method. Both gave similar results, with the exception of one dataset (Figure 4.9). T-test/ANOVA was computationally faster than Mutual Information as was seen in the second experiment (Figure 4.13), however T-test/ANOVA also worked with the assumption that the classes are normally distributed. This was not always the case. A rule of thumb for choice of ranking method would be to use T-test /ANOVA, except for when the data was not normally distributed.

A second conclusion was drawn from looking at the accuracy with the number of features preserved. After a threshold of 200 features, additional features did not raise the validation score as much as the first 200 features did. This indicated a second rule of thumb, that at least 200 features should be preserved after using a filter method. This rule of thumb was according to the hypothesis H1 notion that at most 1000 features should be preserved to still show an efficient result, however was much lower than the expected upper bound of 1000 features. This absolute number was not intuitive and much lower than expected. More research could be done on these preserved 200 features and why no other features were needed to predict the output. Cluster analyses for example may show insights in this phenomenon.

After evaluation of both filter, wrapper and embedded methods, wrapper methods were significantly better at selecting a smaller fraction of features while preserving a similar test score. Whereas hypothesis H2 stated wrapper methods to be best in efficiency, embedded methods were expected to outperform both filter methods and wrapper methods when looking at both quality and computation time. This was not the case, however, as embedded methods had a near identical quality to filter methods and performed much worse than wrapper methods. The outcome indicated that wrapper methods are significantly different in results and filter and embedded methods being nearly identical.

Since wrapper methods take dependencies between features into account, they kept these dependencies at a minimum. If these dependencies are unwanted, wrapper methods are more useful than filter methods and embedded and should be recommended. A downside of the wrapper methods however was that they took much more computation time than the filter and embedded methods. Therefore if it does not matter when features have dependencies with each other, a filter method should be recommended.

There was a difference in quality within the wrapper methods. The forward selection and PTA all showed promising results and therefore should be considered for the framework. Floating search showed the best results, however took significantly longer in computation time. Within the wrapper methods, an ordering beforehand showed improved results. Backward elimination wrapper algorithms, stochastic search algorithms and embedded backwards elimination algorithms all were significantly worse in computation time than the other wrapper algorithms and therefore

should not be considered in these cases.

A recommendation for the threshold in wrapper methods is not trivial. A higher threshold of $\alpha = 0.01$ gave a smaller feature subset at the cost of a lower classification score. If a smaller subset is desired or more influential features are needed, a bigger threshold should be chosen, whereas it will be smaller when a higher quality of the feature subset is desired.

In this research only separate feature selection methods were tested. Whereas the filter methods used very little computation time, the feature subset size and quality of the wrapper methods were much more interesting for further research. A combination of a filter and a wrapper method might create a better result. If first the majority of the features is filtered out by a filter method and then a wrapper method computes the best combination of feature, the result can be relatively quicker than the wrapper method while still maintaining a high enough quality. This combination is worth investigating in future research.

Both additions to *TPOT* were beneficial according to the experiment. This was also expected, as H3 states. When the initial number of features was very high (50.000), a combination of initial bias towards feature selection and using feature selection algorithms focusing on preservation of only 200 features performed best. For lower number of features only adding one of the improvements was better, as computation time became the limiting factor in case of using both improvements.

Chapter 5

Missing Value Handling

5.1 Introduction

The quality of data should be as high as possible. Data values however could be missing, for example due to patients not showing up or because of mistakes by the medical staff. Missing values hinder the data set quality significantly and for numerous datasets removing or replacing them would significantly help to increase the quality of the research. Several works already focused on the handling of missing values [21, 22, 23]. A biomedical data analysis framework would benefit from the possibility to remove or replace missing values. Therefore the research goal for this project is *to evaluate the performance of missing value handling methods and make a choice on which methods should be added to the framework.* Therefore in this document several of those algorithms are presented and tested on their quality.

5.2 Background

5.2.1 Datasets

A total of four datasets were used for the analysis of missing values. Three of these datasets are clinical data and one of them is survey data (Subsection 3.1). The specifications of the four datasets are summarized together (Table 5.1).

Table 5.1: A schematic overview of the four datasets

Dataset focus	**Features**	**Samples**	**Classes**	**Total Missing values (%)**	**Most missing in single feature (%)**	**Features with missing values**	**Remarks**
Heart attack	9	108	53	8.29%	43.85%	7	- Output in months (Regression preferred) - Values are missing due to time constraints
Hepatitis	19	155	2	5.67%	43.22%	15	— - Used in previous published studies
Cirrhosis	18	1945	3	3.24%	42.21%	6	— - Missing not completely at random
Cervical Cancer	28	858	5	15.07%	91.72%	26	— - Missing values mainly from the survey part - Output comprised of four separate indicators

5.2.2 Missing Value Handling

Numerous reasons can cause entries to be missing from a dataset. A sample can be missing, corrupted or contaminated, a measurement malfunction can occur, a patient can fail to show up at a scheduled meeting or not respond to a survey. These missing entries are different on both origin and possibly also on randomness. This also indicates that techniques differ in effectiveness on missing values caused by separate reasons. Therefore the different types of missing values will be explained, as well as techniques to cope with these missing values.

Since there are multiple steps to be taken for gathering data, a problem during an earlier step can generate more values to be missing. When a patient fails to come to the hospital for a second round of tests all values for the second test are absent, whereas a nurse accidentally skipping to note the weight of a patient will only cause the absence of one value. A difference is therefore made between item and unit non-responses. An item response is a single missing value in a data set and a unit non-response corresponds to a series of missing values. [130]

Another aspect of missing values is its randomness. A missing value can occur completely random without any relation with other values, however it is not uncommon to have some relation with another value. Three different types of randomness are defined to explain this. The first type is *missing completely at random* (MCAR) [21, 22, 23], and means that missing entries have no relation with any part of the data. This means removing samples with missing values should not create any bias in the resulting dataset. An example would be accidentally dropping a blood vial, not being able to report the blood values. The opposite of MCAR is *missing not at random* (MNAR) [21, 23], also known as *non ignorable* (NI) [22]. MNAR means that missing values are linked to the feature they are missing from, which makes it very difficult to find the relation between the missingness and a feature. Examples with MNAR values are patients that are unsuccessfully treated for a disease, and are therefore less likely to return for future tests, creating a bias in the values about disease severity. In between MCAR and MNAR is *missing at random* (MAR) [21, 22, 23]. MAR indicates missing entries that have no relation with the feature they are missing from, but have a relation with other features in the data sets. These entries can be estimated using the relation to better identify a possible value. Examples are specific diagnostic tests that are done for patients of a specific disease for increased health risks, whereas other (healthy) subjects were not tested [23]. Whereas MCAR values are relatively easy and MNAR values are almost impossible to cope with, MAR values can be treated properly with complex approaches so proper values are imputed in those cases. Therefore many imputation techniques are created on basis of the MAR assumption.

The techniques of coping with missing values are based on three approaches. The first principle is list deletion (LD) [21, 23], deleting samples, features or both to create a smaller dataset without missing values. This is the most common way to treat missing values as it is quick and easy. If enough samples are present and there is not too much bias between samples including and excluding missing values, this is the way to go. If the number of samples is limited or a bias is present in the missing value samples, value imputation becomes more interesting. In this case valuable information will not be omitted and the possible bias is taken into account. Imputation can be split into two different types, single imputation (SI) [21, 22, 23] and multiple imputation (MuI) [21, 130, 131]. Thirdly, instead of removing or adding values, the dataset can be directly used in a model, disregarding missing values in the model [132]. Since methods handling missing values can usually be best explained by showing example algorithms, several pseudo-algorithms were created for visual clarity.

List Deletion

List deletion (LD) is a quick and easy manner to perform for coping with missing values. Samples or features are removed when they consist of at least one missing value until no more missing values are present. Three difference approaches are explained: *complete case analysis* (CCA), *available case analysis* (ACA) and *weighted case analysis* (WCA).

- *Complete case analysis*

CCA is the most basic LD approach. CCA removes all samples that have missing values and therefore prepares only complete samples for future analysis (Algorithm 11). The dataset should consist of a reasonable number of complete samples to leave enough data for analysis. Also data should be MCAR, as removing cases that have a dependency with certain features can create a bias. CCA is most used as a way of removing missing values, even though not every time MCAR is available [23, 130, 132, 21]. In those cases CCA is proven to be biased [22].

Algorithm 10 Complete Case Analysis

```
1: procedure CCA(X)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     for s in S do                                ▷ For all samples in S
4:         if missing_values(X_s) is True then      ▷ find out if missing values are present
5:             remove X_s from X                    ▷ Remove the sample with missing values
6:         end if
7:     end for
8:     return X
9: end procedure
```

- *Available case analysis*
  ACA tries to preserve more samples by only using useful features. ACA first makes a selection on relevant features in the dataset. Those features are selected by hand so they will be present in future analysis. A second approach is to remove features for which the percentage of missing values is higher than a threshold α. After selecting those features, samples without missing values in the relevant feature set are kept, all others are removed (Algorithm 11). This is similar to CCA, but will preserve more samples if several features have significantly more missing values. For this LD, MCAR would be desired. However if either the MAR values or the features it has a relation with are removed, there will be no relation between the MAR values and the dataset any more and therefore no bias will be created [23, 21, 130, 132]. Often though, this will not be the case, as the dependency is on either a multitude of features and the dependent feature is important for the research.

Algorithm 11 Available Case Analysis

```
1: procedure ACA(X, α)
2:     F ← range(#columns(X))                       ▷ Create a list of all samples
3:     for f in F do                                ▷ For all features in F
4:         if perc_missing_values(X_f) > α then ▷ perc_missing_values finds out the percentage of
   missing values
5:             remove X_f from X                    ▷ Remove the feature with too many missing values
6:         end if
7:     end for
8:     return CCA(X)
9: end procedure
```

- *Weighted case analysis*
  A third type of LD is WCA. WCA tries to overcome bias from CCA by assigning weights to every value (Algorithm 12). If a sample is removed, the nearest complete neighbour is searched for (Algorithm 12 - row 5). This closest complete sample can be found by using one, multiple or available features. The complete sample is then given a higher weight corresponding to the number of closest omitted samples. A simple approach for giving a higher weight is simply by adding an additional complete case to the existing complete

cases, choosing the one that is closest to the incomplete case. This technique tries to remove the bias as much as possible by maintaining the distribution of the samples. This technique specifically does not assume MCAR values, but MAR values [23, 21]. Also this technique is specifically useful for unit non-responsiveness for its simplicity to cope with multiple missing values in one sample [130].

Algorithm 12 Weighted Case Analysis

```
1: procedure WCA(X)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     for s in S do                                ▷ For all samples in S
4:         if missing_values(X_s) is True then      ▷ find out if missing values are present
5:             s_nn = nearest_complete_neighbour(X, s) ▷ Find the nearest complete neighbour of s
6:             X_s ← X_{s_nn}                       ▷ Change the values of s to the values of s_nn
7:         end if
8:     end for
9:     return X
10: end procedure
```

Whereas List Deletion is used most regularly and is quickest and easiest of all the missing value handling techniques, it is often criticized for its drawbacks. If the data is not MCAR, bias will most often be created, even in ACA and WCA [22]. Also the removal of samples is not always desired when not many samples were available at the start [23, 132, 21]. All things considered LD is a quick fix when needed. For better understanding of the data however, other approaches seem more appropriate.

Single Imputation

An intuitive approach of coping with missing values and not wanting to delete entire samples or features is to impute an appropriate entry in its place. If the imputed entry is a proper representation of all possible entries, the possibly valuable information within the sample or feature will be useful for data analysis. Imputing a replacement value for the missing value is called Single Imputation (SI). Several different approaches for SI are known and used. [23, 21, 132]

- *Missing indicator method*
  The simplest way of SI is missing indicator method. This approach imputes a generic variable for every missing value of a feature, for example imputing 0 for heart rate for every missing measurement. Aside from that, it also creates an additional boolean feature that indicated whether a value was missing or not (Algorithm 13). When using a generic feature no additional information is added to the data and no assumptions are made, which makes this method less dependent on assumptions [133]. Another possibility is to combine the missing indicator method with other imputation methods, for example the mean imputation, so from the dataset missing values can still be traced back. Any calculations or modelling done with the feature has an important dependency to the newly created missing value feature, though. Also if data is MAR and not MCAR, bias may be present due to worse representation [21].

Algorithm 13 Missing Indicator Imputation

```
1: procedure MISSING_INDICATOR_IMPUTATION(X, a)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     F ← range(#columns(X))                       ▷ Create a list of all features
4:     for f in F do                                ▷ For all features in F
5:         if missing_values(X_f) is True then      ▷ find out if missing values are present
6:             append X_f' to X                     ▷ Create a missing indicator feature for f
7:             for s in S do                        ▷ For all samples in S
8:                 if missing_values(X_s,f) is True then   ▷ Find out if missing
9:                     X_s,f ← a                    ▷ Impute generic value for missing vlaue
10:                    X_s,f' ← True                ▷ Assign true for missing indicator value
11:                else
12:                    X_s,f' ← False               ▷ Assign false for missing indicator value
13:                end if
14:            end for
15:        end if
16:    end for
17:    return X
18: end procedure
```

- *Mean/Median/Mode imputation*
  The values of a feature usually follow a certain distribution. In a continuous distribution values are most probable to be the mean of the distribution. Therefore if a value should be imputed, the mean would be the most logical imputation (Algorithm 14). For ordinal features this would be the median and for categorical the mode [23, 132, 22]. With this imputation the distribution centre stays the same, but the standard error is lowered due to addition of new values. This creates a bias for the variance of the feature. Moreover if the data is MAR and not MCAR, additional bias will be created. The original and imputed values combined will then not be a representation of the total number of values [21, 133].

Algorithm 14 Mean Imputation

```
1: procedure MEAN_IMPUTATION(X)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     F ← range(#columns(X))                       ▷ Create a list of all features
4:     for s in S, f in F do                        ▷ For all samples in S and features in F
5:         if missing_values(X_s,f) is True then    ▷ find out if missing values are present
6:             X_s,f ← mean(X_f)                    ▷ Assign the mean of the values for feature f
7:         end if
8:     end for
9:     return X
10: end procedure
```

- *Hot deck imputation*
  Whereas the application of hot deck imputations differs somewhat [132, 22, 23], the main concept is based on randomly picking a value from a set of possibilities. The set can be all known possible values for the feature, known from other samples in the dataset (Algorithm 15). Another approach would be to assume the feature is distributed a specific way (for example normally distributed) and randomly pick a value using the values from that distribution. A third possibility is to combine this with kNN imputation and randomly pick a value from the k nearest neighbours. Randomly picking a value instead of calculating a fixed value will not unjustifiably reduce the variance of the distribution. On the other hand, it also makes the estimation to be more imprecise, possibly creating bias this way [23].

Algorithm 15 Hot Deck Imputation

```
1: procedure HOT_DECK_IMPUTATION(X)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     F ← range(#columns(X))                       ▷ Create a list of all features
4:     for s in S, f in F do                        ▷ For all samples in S and features in F
5:         if missing_values(X_{s,f}) is True then  ▷ find out if missing values are present
6:             X_{s,f} ← random(X_f)                ▷ Assign a random choice of the values for feature f
7:         end if
8:     end for
9:     return X
10: end procedure
```

- *Multi-variate regression imputation*
  Instead of only using information of the feature that has missing values, other features can be used to predict the missing value replacements. Especially when multicollinearity is present, a prediction of the actual missing value can be computed quite precisely. A multivariate regression model can be made by using features without missing values as input and the features with missing values as output. For example, with linear regression a function can be made between the input and output, trained by all available values and used for predicting all missing values (Algorithm 16) [134]. This approach is especially suited for MAR values, in which the absence of feature values is related with other features. The regression imputation should be used with care, as it is unwise to be used if afterwards a similar regression technique is used for further analysis [21].

Algorithm 16 Multivariate Regression Imputation

```
1: procedure REGRESSION_IMPUTATION(X)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     F ← range(#columns(X))                       ▷ Create a list of all features
4:     for f in F do                                ▷ For all features in F
5:         if missing_values(X_f) is True then      ▷ Find out if missing values are present
6:             create regressor                     ▷ Initialize a regressor
7:             fit regressor with X_{F\{f}}, X_f    ▷ Fit the regressor by using other features as input and the feature with missing values as output
8:             for s in S do                        ▷ For all samples in S
9:                 if missing_values(X_{s,f}) is True then   ▷ If a missing value is present
10:                    X_{s,f} ← with regressor predict X_{s,F\{f}}   ▷ Impute a predicted value
11:                end if
12:            end for
13:        end if
14:    end for
15:    return X
16: end procedure
```

- *k-Nearest Neighbour imputation*
  Similarly to the multi-variable regression imputation, the k nearest neighbour (kNN) imputation can be used to predict missing values more precisely. This approach also makes use of features that do not have missing values. Similarly to kNN classification and regression it takes the values of the k samples that are closest to the sample with the missing value and combines those into a replacement for the missing value (Algorithm 17). This is also suited for MAR values, as relations with other features are used for prediction [132, 22]. For kNN imputation, further data analysis steps must also be taken with care as additional nearest neighbour analysis techniques may give biased results [21].

Algorithm 17 k Nearest Neighbour Imputation

```
1: procedure kNN_IMPUTATION(X, k)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     F ← range(#columns(X))                       ▷ Create a list of all features
4:     for f in F do                                ▷ For all features in F
5:         if missing_values(X_f) is True then      ▷ Find out if missing values are present
6:             create kNN-model(k)                  ▷ Initialize a kNN-model
7:             fit kNN-model with X_{F\{f}}, X_f    ▷ Add cases to the kNN-model
8:             for s in S do                        ▷ For all samples in S
9:                 if missing_values(X_{s,f}) is True then   ▷ If a missing value is present
10:                    X_{s,f} ← from kNN-model extract X_{s,F\{f}}   ▷ Extract a value with kNN
11:                end if
12:            end for
13:        end if
14:    end for
15:    return X
16: end procedure
```

- *Worst Case Analysis*
  When samples in a dataset have missing values for an important feature, a safe choice can be made by replacing them with the best or worst possible value (Algorithm 18). An example would be if a feature depicts the severity of a disease. If false positives on severe patients are more justifiable than false negatives, all missing values can be replaced by the highest severity value. An advantage of this technique is that boundaries can be made with the analysis and for the example false negatives would occur less. Off course a disadvantage would be that the outcome will not be as precise, due to the imprecise assumption. Also not all data analysis only need bounds and rather have strict models [133, 23].

Algorithm 18 k Nearest Neighbour Imputation

```
1: procedure WORST_CASE_IMPUTATION(X, worst_cases)
2:     S ← range(#rows(X))                          ▷ Create a list of all samples
3:     F ← range(#columns(X))                       ▷ Create a list of all features
4:     for s in S, f in F do                        ▷ For all samples in S and features in F
5:         if missing_values(X_{s,f}) is True then  ▷ find out if missing values are present
6:             X_{s,f} ← worst_case(X_f)            ▷ Assign the worst case of feature f
7:         end if
8:     end for
9:     return X
10: end procedure
```

Single imputation consists of the quickest and easiest types of approach to replace a missing value. The approaches mostly are very straightforward and protect samples from being removed from the dataset by deletion [23, 22]. Especially for small sampled datasets, this would be a quick outcome to not remove too many samples. Using SI methods should be done with care, since not all are fit to be used with MAR values. Also, except for hot deck imputation, all SI techniques cause the feature to have a lower variance that can cause bias in further research on the dataset [21].

Multiple Imputation

Multiple imputation (MuI) creates additional datasets, for the dataset with missing values. These new datasets will have different values imputed, either from some kind of distribution similar to

hot deck imputation or by using a different algorithm. After doing multiple hot deck imputations, analysis is done on all different datasets and afterwards the results are combined (Figure 5.1). Three aspects are important: The value imputation, the creation of multiple datasets and the analysis before merging.

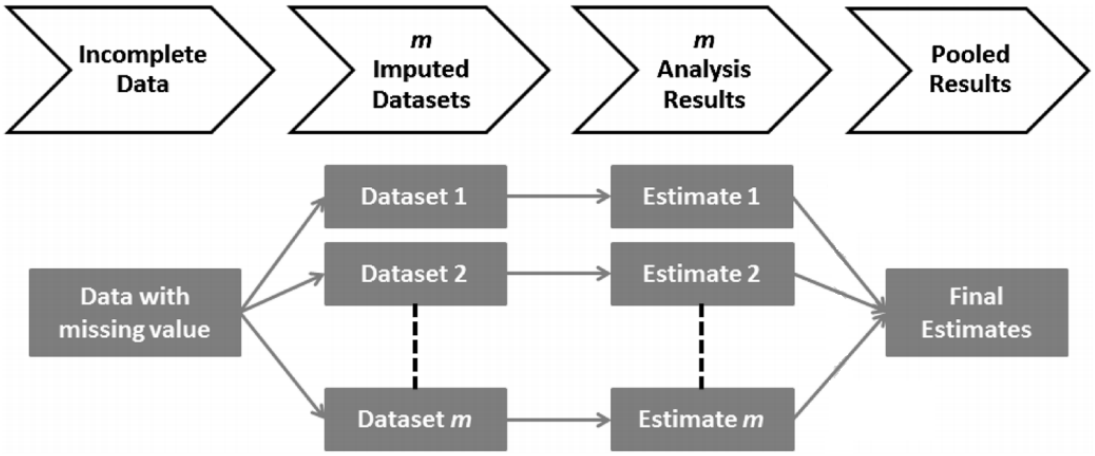


Figure 5.1: An graphical layout of multiple imputation. At the start a dataset with missing values is present. Missing values are imputed m times to create m imputed datasets. The m complete datasets are all analysed for m different sets of estimates. These results are then pooled into one complete set of estimates. [135, 136]

- *Value imputation*
  Different approaches are available for the value imputation per dataset. One is similar to the previously discussed SI hot deck imputation. The quickest imputation would be to randomly choose a value from the distribution of the feature. After analysing all different datasets and combining them, it would seem as if the missing values were following the same distribution. Other techniques are similar to the multivariate regression imputation and the kNN imputation [133, 137]. The variance can be added manually. The variance is then averaged for all variances and in the end using a new averaged variance [21]. Other mainly regression methods use different initial seeds as a way of creating a variance [138].

- *Additional datasets*
  The number of suggested datasets is different per study and usually differs between 3 and 10 [139, 133, 140, 141, 142, 143], however outliers of 15 [144] and even 25 [134] are known. Other researchers even propose the number of datasets to be the same as the percentage of missing values [137]. The number of datasets is proportionally to the quality of the result and more datasets would indicate a better result. However generating and analysing more datasets also takes much more computation time, therefore a trade-off in computation time and quality should be considered.

- *Analysis*
  The analysis of the dataset is project specific. For example, if the outcome should be a linear equation between features and output, linear regression should be used to compute such an equation. Usually the outcome of such an analysis is more precise than it actually should be. By doing the same analysis multiple times with MuI, a measure of uncertainty can be given to the outcome. This is done by combining the different outcomes together to one final outcome, by for example show standard deviations per regression coefficient [21, 141].

The main advantage of multiple imputation is the addition of imputation uncertainty. In single imputation this uncertainty is unfairly low, which can create bias in results. Because of it being an extension on single imputation, it also inherits the advantages of being able to handle MAR values and possibly MNAR, and raises the number of samples to be used for analysis [141, 133]. The disadvantages mainly consist of taking up more computation time, due to multiplication of the analysis for the datasets. Therefore complex analysis techniques that take longer to compute may become unusable. Another aspect is the error given to missing value imputations. Not all analysis techniques perform well with errors which may create worse results even after combining the results [133].

Several MuI implementations are created and also added to mainly statistical analysis programs as SOLAS, SAS, SPSS, S-Plus and Stata [145, 146, 147]. The most used implementation is multiple imputation for chained equations (MICE) [142, 143], also known as sequential regression multiple implementation (SRMI) [138]. MICE stores the location of the missing values and iteratively replaces them with new values, according to a six step system [142].

1. Replace all missing values with the mean of the corresponding feature.
2. Choose a feature f and remove the imputed values on all missing value locations.
3. Create a model with all samples that do not have a missing value in f to predict the value in f by the other features.
4. Impute the values in f using the created model.
5. Repeat steps 2 to 4 for all features that have missing values.
6. Iteratively repeat steps 2 to 5 s cycles for better more precise imputed values to create a final dataset.

The six steps create one dataset. The steps are m times with different orders of choosing f to create m different datasets for analysis. The number of cycles used in the sixth step influences the precision of the outcome, as eventually the imputed missing values are expected to converge to one value. Usually about 10 cycles are done, as it is a good equilibrium between low computation time and high imputed value precision [142, 143].

5.3 Hypotheses

The main goal for this chapter is to find more insight in quality enhancement and add that insight to the final framework. Several hypotheses are made before designing experiments to test them and are discussed here:

1. *Distribution preservation*
   One thing missing value handling algorithms should do is preserve the distribution of a feature, not creating any bias in it. List Deletion methods CCA and ACA are not expected to do that properly in most cases, as most values are not MCAR. WCA might be able to efficiently tackle distribution problems, but is expected to be outperformed by imputation methods. Of the imputation methods MICE is expected to perform best, followed by kNN and regression imputation. Hot deck and mean imputation obviously are also expected to preserve distribution characteristics very well as their imputation method is based on it.

   H1: The advanced imputation methods (MICE, kNN and regression) are expected to preserve the distribution very well, much better than the list deletion methods. Hot deck and mean imputation should be able to preserve the distribution, too, as they use distribution characteristics in their imputation.

2. *Quality improvement*
   Missing values handling algorithms are necessary, however can show different levels of quality improvement. This difference in quality shows the best approaches to cope with missing values. Again, list deletion algorithms are expected to perform worst in quality improvement. Next are the easy single imputation methods, the mean imputation, hot encoding and missing indicator algorithms. These are followed by the more advanced single imputation algorithms the kNN and regression algorithms. The multiple imputation algorithm MICE is expected to outperform all.

   H2: The advanced imputation methods (MICE, kNN and regression) are expected to improve quality the most, followed by the more simple imputation algorithms (means, missing indicator and hot deck). List deletion methods are expected to perform worst in quality improvement.

5.4 Methods

To determine the quality of different missing value handling algorithms, two different approaches are used. The first approach focuses on the distributions of the different features and whether the algorithms create a bias in feature distributions. The second approach focuses on the evaluation quality and which missing value handling algorithm works best on predicting output. The four example datasets (Subsection 5.2.1) were used as example datasets for the experiments.

5.4.1 Bias Evaluation

After manipulating a dataset, the distributions of a feature can change significantly. This change in distribution can create a bias in the results and must be prevented as much as possible. To find out whether this bias is present, feature type specific values are compared between both the old and new feature values and those are compared using several statistical methods.

For future mentions of the two distributions that are compared the names 'old data' (before list deletion or value imputation) and 'new data' (after list deletion or value imputation) are used. The statistical methods that are used to find bias are all based on the same hypothesis Hyp0: the old data and new data originate from the same distribution. This means the rejection hypothesis becomes Hyp1: old and new data do not originate from the same distribution. Since features were numeric, ordinal or categorical, every feature type hypothesis is evaluated differently:

- *Numeric features*
  For numeric features the mean and variance are main aspects to compare numeric distributions. To either accept the hypothesis, these two aspects are tested. The mean comparisons were done by using the known t-test [98]. Normality is not assumed, however the t-test does not show worse performance when the sample size is reasonable [148], which is the case. The equality of variance is tested by using a Levene's test with the Brown and Forsythe adaptation. This variance test was chosen due to not having any major assumptions on the distributions, for example the normality distribution [149].

- *Ordinal features*
  For ordinal features the mean cannot be used, as values are either not numeric or only give an indication with the numeric values. Because of that the median is used instead. The hypothesis is tested by checking whether this median is the same for both the old and new values. To have a better look at the distribution of the ordinal values, also a chi square test is done. This chi square test computes whether the distribution of ordinal values is the same for both the old and new values [150].

Table 5.2: The three different aspects of the bias evaluation. Four different datasets are used. Two tests are done per feature type, to test hypothesis Hyp0 and seven missing value handling algorithms are implemented to be tested.

Datasets	Heart Attack dataset Hepatitis dataset Cirrhosis dataset Cervical cancer dataset	All datasets containing missing values that are evaluated
Test types	Numeric features (t-test, Levene's test) Ordinal features (median, chi-square) Categorical features (mode, chi-square)	Different types of features have different tests for hypothesis Hyp0: The old and new distributions are the same
Methods	CCA WCA Mean/Median/Mode imputation Hot deck imputation kNN imputation Regression imputation MICE	All methods to handle missing values: - Two List Deletion algorithms - Four Single Imputation algorithms - One Multiple Imputation algorithms

- *Categorical features*
  Categorical features do not have a mean or median, due to no ordering being present. Therefore the mode of the old and the new distribution is checked to be the same, to test the hypothesis Hyp0 to be true. On top of that, the chi square test is here used, to find out whether the features from before and after the missing values algorithms can follow the same distribution [150].

All features from the four datasets are tested whether any bias is present or not. The missing values are handled by seven different algorithms: CCA (Algorithm 10), ACA (Algorithm 11), Mean/Median/Mode imputation (Algorithm 14), Hot Deck imputation (Algorithm 15), kNN imputation (Algorithm 17), Regression imputation(Algorithm 16) and at last MICE (Subsection 5.2.2). An overview of the datasets, evaluation tests and the missing value handling algorithms is provided (Table 5.2).

The outcome of all the tests are shown in multiple tables. In those tables tests that show a rejection of Hyp0 are highlighted. For the list deletion algorithms, additionally the type of missing values is evaluated by also testing the change in distribution in values without missing values. If a feature distribution changes for a feature without missing values after removing all missing values, a relation is present and the missing values are not MCAR. Aside from these tables plots of the relation between the percentage of missing values and probability of bias being present are made for every algorithm. This can indicate quality between the different algorithms with regards of the statistical hypothesis. In those plots also a regression line is shown with also the R^2 value, to show the quality of the regression [151]. An equation for R^2 is given for clarification (Equation 5.1).

$$R^2 = \frac{SS_{tot} - SS_{res}}{SS_{tot}}), SS_{tot} = \sum (y_i - \bar{y})^2, SS_{res} = \sum (y_i - f_i)^2 \tag{5.1}$$

R^2 is measured by first measuring the variance sum towards the mean (SS_{tot}, y_i is the value for data point i, $\bar{y}$ is the mean of data points y) and the variance sum towards the regression line (SS_{reg}, f_i is regression value for i). Secondly the difference is measured and this difference is normalized by dividing it by the mean variance sum. R^2 should be a value on interval $[0, 1]$ with 1 being a good fit and 0 being a bad fit.

5.4.2 Quality Evaluation

To determine the quality of the missing value imputation methods, the classification quality of the dataset was tested after handling the missing values. This is done for the Hepatitis, Cirrhosis and Cervical Cancer datasets, as on the Heart Attack dataset regression should be performed. This classification quality was tested by using the basic logistic regression algorithm from *scikit-learn*. This classification algorithm only accepts numerical data, therefore all categorical features are first hot encoded. Previously discussed missing value handling algorithms (Table 5.3) are tested on their accuracy, precision, recall and F1 score with a 10-fold cross validation. For the datasets an additional classification was done after removing the values with more than 15% missing values, to find out if that would give a difference in quality, as well.

Table 5.3: All tested missing value handling methods and possible parameters used during testing

Type	**Method**	**Parameters**
List Deletion	CCA (Algorithm 10)	
	ACA (Algorithm 11)	- threshold: 10% missing values
	WCA (Algorithm 12)	
Single Imputation	Mean/Median/Mode (Algorithm 14)	
	Hot deck (Algorithm 15)	
	Missing indicator (Algorithm 13)	- Imputed value: Mean, 0
	Regression (Algorithm 16)	
	k-Nearest neighbour (Algorithm 17)	- k = 1, 3, 5
Multiple Imputation	MICE (Subsection 5.2.2)	s = 3, 5 m = 3, 5

To show the differences between the datasets, the best outcomes are shown separately, as well as an average over the three datasets. This way interesting differences between the datasets can be located and discussed.

5.5 Results

5.5.1 Bias Evaluation Results

The outcome of the distribution tests for the heart attack dataset are shown (Tables 5.4 and 5.6), as well as the tables for the other datasets (Appendix F.1). The tables are split between list deletion (Table 5.4) and imputation (Table 5.6) results.

Table 5.4: Testing for the heart attack data set whether the type of missing values can be represented by the remaining values. This is done by comparing distributions between the old and new data after CCA and WCA. For numeric values, two tests were used, an independent t-test for equality of mean and a Levene's test with the median in brackets to test equality of variance. For ordinal and categorical features the medians and modes where compared respectively as well as a chi squared test in brackets for equality of distribution. P-values lower than $p < 0.05$ are marked red for failure of representation, p-values higher than $p > 0.05$ are marked green for correctly being represented. If at least one feature is not represented after CCA, the missing values cannot be MCAR. If at least one feature is not represented after WCA, the pseudo-randomness cannot be corrected by only using weights for other values.

Feature name	**still alive**	**age**	**peri-cardial**	**frac-tional**	**epss**	**lvdd**	**wall score**	**wall index**	**alive at 1**
Value type	Cat	Num	Cat	Num	Num	Num	Num	Num	Cat
Missing (%)	0%	3.85%	0%	5.38%	10.77%	7.69%	2.31%	0.77%	43.85%
p-values CCA	True (0.96)	<0.01 (0.39)	True (0.99)	<0.01 (0.83)	0.99 (0.74)	0.99 (0.94)	<0.01 (0.98)	0.09 (0.84)	True (0.94)
p-values WCA	True (1.00)	0.48 (0.82)	True (0.98)	<0.01 (0.45)	0.99 (0.39)	0.99 (0.28)	<0.01 (0.54)	0.99 (0.91)	True (0.92)

The tables show that in the heart attack, cirrhosis and cervical cancer datasets at least one feature shows a difference between the old and new data. Highlights are shown for changes in distribution for all values:

- *Heart attack*
  age, *fractional* and *wall score* all have $p < 0.01$ for having the same mean before and after CCA. This indicates that the missing value type of at least one feature is MAR and for all three possibly even MNAR. WCA does help in creating a representative distribution for the feature *age*, however *fractional* and *wall score* still are differently distributed. Other approaches to handle missing values are most likely more effective

- *Hepatitis*
  For no feature a reason to reject the distributions before and after deleting samples with missing values is present. The feature that is least likely to have the same distribution is *bilirubin* with having a $p = 0.14$ for the same mean and $p = 0.23$ for the same variance. We therefore cannot say that the missing values are MCAR and using only CCA to handle missing values is certainly possible. WCA has worse results for several features and show that WCA might not be the best way to improve the sample size for this dataset.

- *Cirrhosis*
  case number, *status*, *day*, *albumin* and *SGOT* all have $p < 0.05$ if numeric for having the same mean and *False* if ordinal or categorical for having the same median or mode, respectively. These five features do not have missing values. This means that the bias in the distribution is caused by list deletion due to another feature's missing values. Therefore it can be concluded that at least for one feature the missing values are at least MAR. WCA helps fixing the distributions for *case number* and *SGOT*, but also creates additional bias in *presence of ascites*, *presence of hepatocytes* and *presence of edema* and therefore is not a suitable approach to remove possible bias.

- *Cervical cancer*
  STDs, *#STDs*, *condylomatosis*, *vulvoperineal condylomatosis* and *STDs #diagnosis* all have $p < 0.05$ if numeric for having the same mean and *False* if categorical for having the same mode. All but *STDs #diagnosis* also have a $p < 0.05$ for having the same variance if numeric or the same chi square distribution if categorical, which further strengthens hypothesis H1

of the distributions being different. This indicates that the missing values of at least one feature is MCAR.

Table 5.5: A table showing the number of samples that remained after performing CCA.

Dataset	Total samples	Number of samples after CCA
Heart Attack	108	61
Hepatitis	155	80
Cirrhosis	1945	1113
Cervical Cancer	585	59

The heart attack, cirrhosis and cervical cancer datasets all indicate possible bias after deleting samples with missing values. On top of that, after removing all samples with missing values not many samples are left for any of the datasets (Table 5.5). The hepatitis dataset loses almost 50% of its samples after CCA and the cervical cancer dataset drops down to only 10% of its original values. All four datasets would therefore potentially benefit from an imputation approach.

Most imputation methods show an improvement in results. The heart attack dataset has no bias in distributions after any of the five tested imputation methods, the hepatitis dataset only has bias when using mean imputation or MICE, the cirrhosis dataset only shows bias when using mean imputation and at last in the cervical cancer dataset all but the two features with 91.75% missing values contain no bias between distributions. On top of that, the only features that had bias in the newly created distributions are features with more than 40% missing values, indicating that imputation methods have more problems when a higher missing/not missing ratio is present. This gives an indication that removal of features with missing values higher than 40% would greatly help the usefulness of a dataset.

A scatter plot with fitted linear curves is made to show a relation between the probability of the old and new mean (Figure 5.2) and variance (Figure 5.3) originating from the same distribution and the percentage of values missing. Although the R^2 values for some of the regression lines seem plausible, the value distribution does not seem to support this. Most features are within the first 15% of the missing values and the features above 15% missing values seem to deviate significantly from the others.

It seems that it is difficult to perform imputation on features with more than 15% missing values. Therefore an additional approach was used, by first performing ACA for features with more than 15% missing values, followed by the known missing value handling techniques (Figure 5.4 and 5.5).

List Deletion algorithms seem to be better at creating a representative distribution when more missing values are present. This is explainable for these datasets as all of them have a at least one feature with more than 40%. Most samples are removed because of this feature also removing significant portions of other features. This results in features having more missing values will leave them with a higher percentage of their original values, therefore being closer to the original values.

The imputation algorithm seem to show big differences between the results. Hot deck imputation seems to create the best representative distribution, being the best when comparing mean and variance for the old and new distribution. Mean Imputation obviously shows a perfect fit for comparing means, however for variance imputation it is worse than both hot deck, kNN and regression. kNN imputation gave the best result after hot deck imputation. Interestingly kNN with k=1 neighbours shows the p-values for variance whereas k=3 and k=5 show the best p-values for the mean. Regression imputation shows good results as well, but is overall worse than kNN. MICE unexpectedly performs worse than regression and kNN in both mean and variance, even though it was expected to perform better.

The regression lines for all imputations seem reasonable in performance. Either the p-value is almost independent of the percentage missing values (mean, hot deck, regression, kNN) or the $R^2 \approx 0.5$, showing somewhat of a trend in the data. Interestingly the p-value seems to deviate

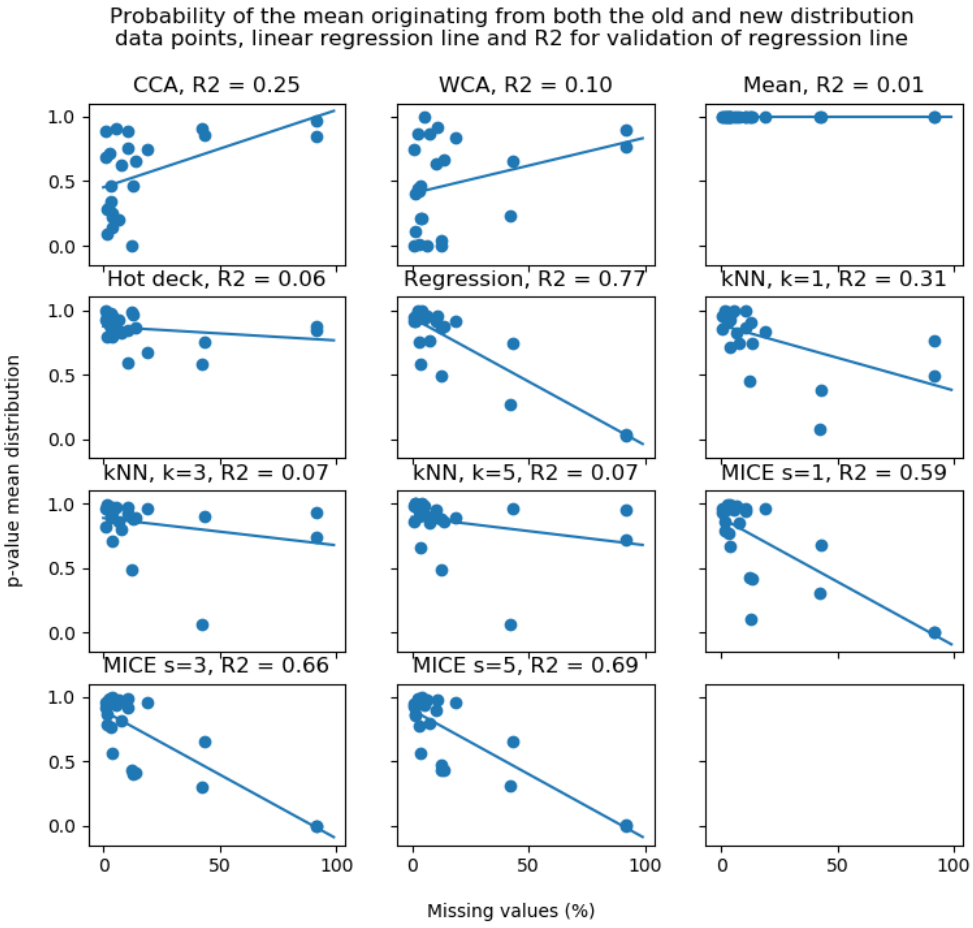


Figure 5.2: Plots showing the probability of bias being present in feature distributions. The mean of the features with the old distribution with missing values and the new distribution after missing value handling are compared and the p-value of them both originating from the same distribution is computed. Every feature of every dataset is a data point. On the x-axis the percentage of missing values is given for a feature and on the y-axis the p-value of the probability. For every missing value algorithm also a linear fit is made to show the trend of the scatter plot.

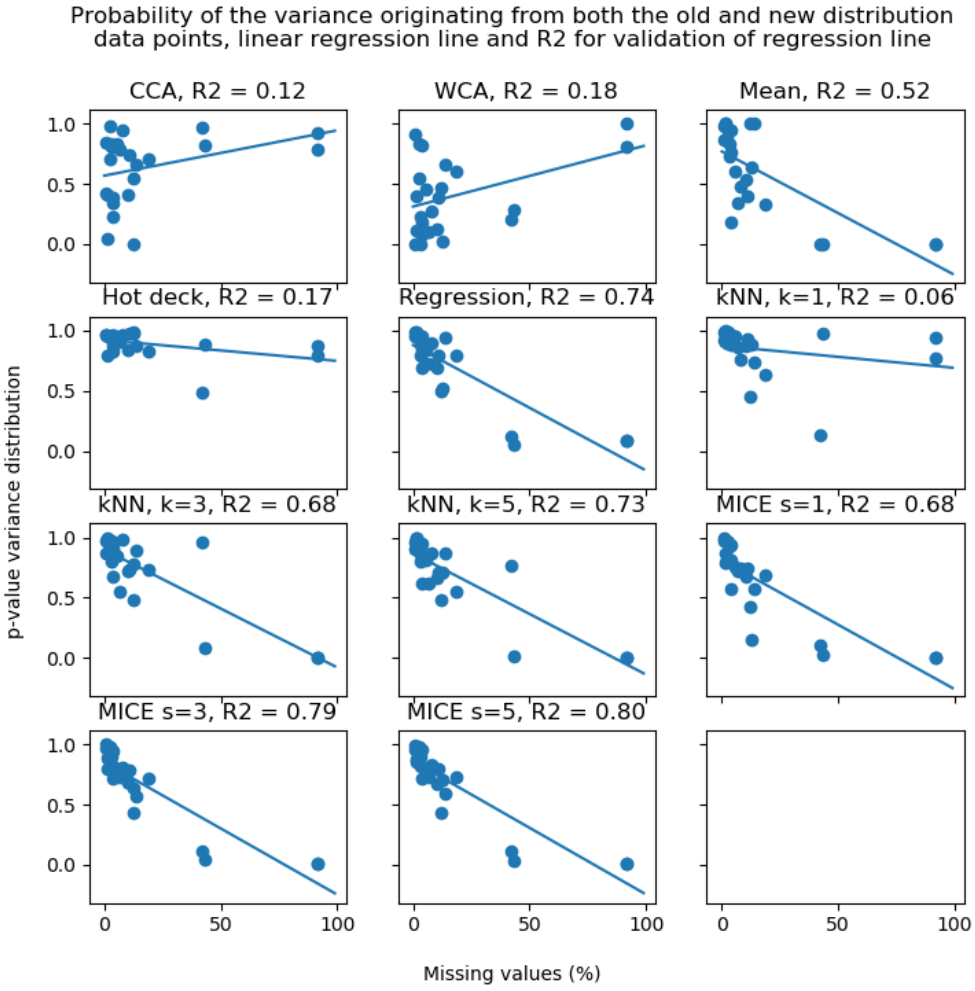


Figure 5.3: Plots showing the probability of bias being present in feature distributions. The variance of the features with the old distribution with missing values and the new distribution after missing value handling are compared and the p-value of them both originating from the same distribution is computed. Every feature of every dataset is a data point. On the x-axis the percentage of missing values is given for a feature and on the y-axis the p-value of the probability. For every missing value algorithm also a linear fit is made to show the trend of the scatter plot.

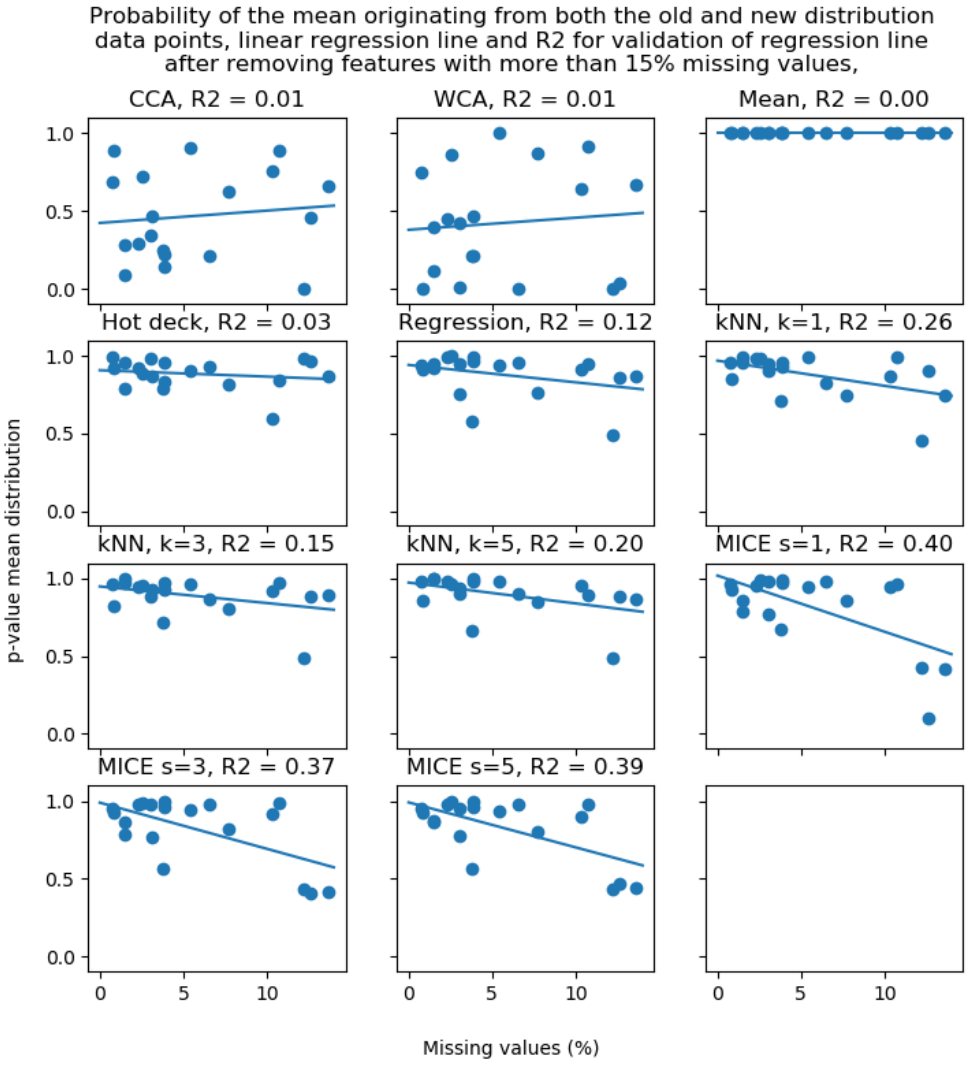


Figure 5.4: Plots showing the probability of bias being present in feature distributions. The mean of the features with the old distribution with missing values and the new distribution after missing value handling are compared and the p-value of them both originating from the same distribution is computed. Every feature of every dataset is a data point, however features with more than 15% missing values are removed. On the x-axis the percentage of missing values is given for a feature and on the y-axis the p-value of the probability. For every missing value algorithm also a linear fit is made to show the trend of the scatter plot.

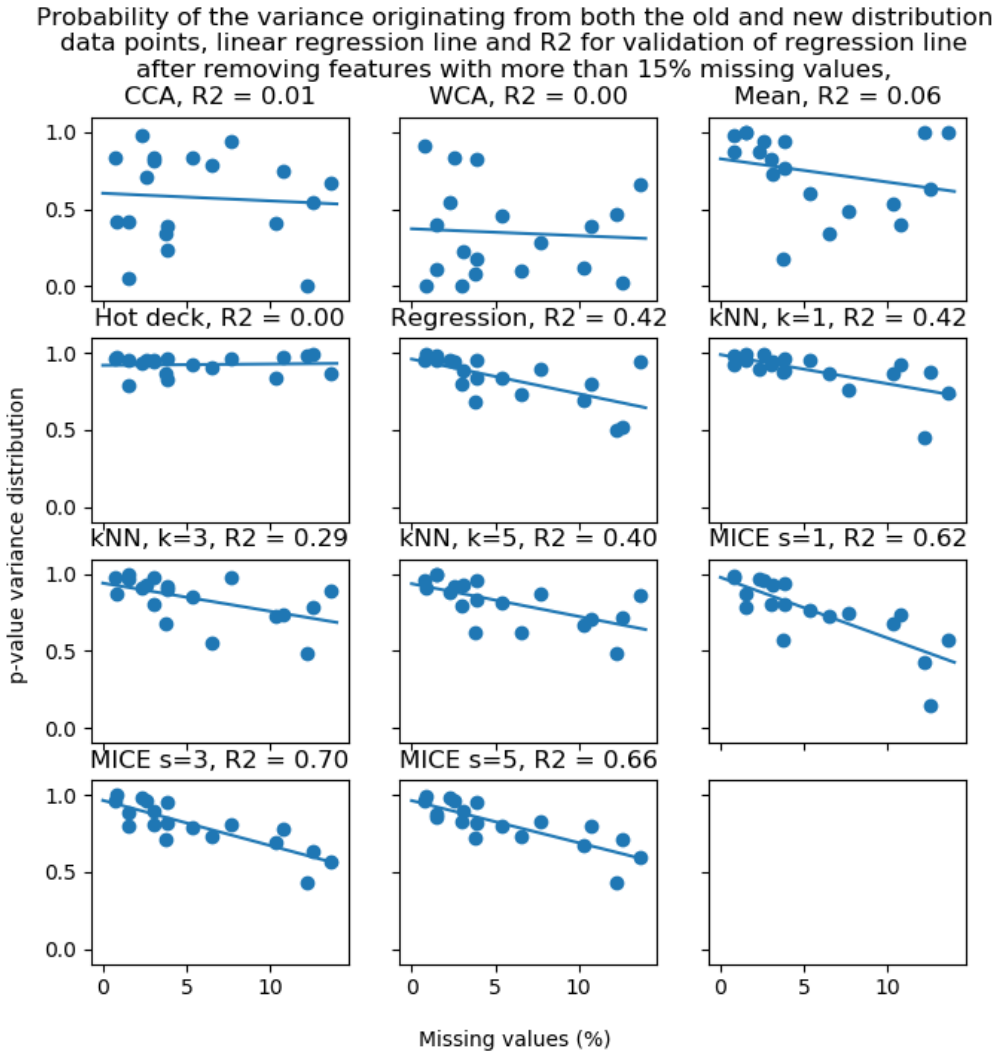


Figure 5.5: Plots showing the probability of bias being present in feature distributions. The variance of the features with the old distribution with missing values and the new distribution after missing value handling are compared and the p-value of them both originating from the same distribution is computed. Every feature of every dataset is a data point, however features with more than 15% missing values are removed. On the x-axis the percentage of missing values is given for a feature and on the y-axis the p-value of the probability. For every missing value algorithm also a linear fit is made to show the trend of the scatter plot.

Table 5.6: Testing for the heart attack data set if certain types of imputation create a vastly different distribution for features with missing values. The imputation values are generated with mean imputation, hot deck imputation, k-Nearest Neighbour imputation ($k = 3$), regression imputation and MICE (number of cycles is $s = 5$). For numeric values, two tests were used, an independent t-test for equality of mean and a Levene's test with the median in brackets to test equality of variance. For ordinal and categorical features the medians and modes where checked respectively to be similar as well as a chi squared test in brackets fro equality of distribution. P-values lower than $p < 0.05$ are marked red for failure of representation, p-values close to $p > 0.05$ are marked green for correctly being represented.

Feature name	**age**	**fractional**	**epss**	**lvdd**	**wall score**	**wall index**	**alive at 1**
Value type	Num	Num	Num	Num	Num	Num	Cat
Missing	3.85%	5.38%	10.77%	7.69%	2.31%	0.77%	43.85%
Mean Imputation	1.00 (0.76)	0.99 (0.60)	0.99 (0.39)	1.00 (0.48)	1.00 (0.88)	1.00 (0.98)	True (0.77)
Hot Deck Imputation	0.97 (0.98)	0.88 (0.88)	0.94 (0.98)	0.82 (0.91)	0.94 (0.86)	0.96 (0.96)	True (0.96)
kNN Imputation	0.86 (0.99)	0.81 (0.88)	0.69 (0.62)	0.94 (0.68)	0.96 (0.97)	0.96 (0.98)	True (0.91)
Regression Imputation	0.84 (1.00)	0.71 (0.95)	0.34 (0.57)	0.91 (0.76)	0.88 (1.00)	0.93 (0.98)	True (0.92)
MICE (5 cycles)	0.97 (0.81)	0.94 (0.80)	0.98 (0.90)	0.80 (0.84)	0.98 (0.98)	0.96 (0.97)	True (0.93)

much more for higher percentages, something that is to be expected since the distribution can also change more when more values are missing.

The high quality for both mean and hot deck imputation can be explained by the evaluation methods. A mean comparison for mean imputation is obviously perfect, and the variance will always become smaller when more means are imputed. Hot deck imputation adds values according to the distribution of the features. This distribution follows the mean and variance of the original values and therefore the mean and variance will always be close to the original distribution. When looking at other aspects, for example the ability to predict the output with the dataset, hot deck imputation and mean imputation should show worse results as distribution specific estimations were imputed, instead of sample specific.

5.5.2 Quality Evaluation

The average results of the classification after using the missing value handling algorithms (Table 5.3) are shown in several figures. The average accuracy is shown for all three datasets without (Figure 5.6) and with (Figure 5.7) the removal of features with more than 15% missing values and without. The F1-score is also shown for the averaged dataset (Figure 5.8). At last a table to show the accuracy and F1 score is shown when all features with missing values are deleted to detect improvement for missing value imputation, as well as the best missing values handling algorithm for comparison (Table 5.7).

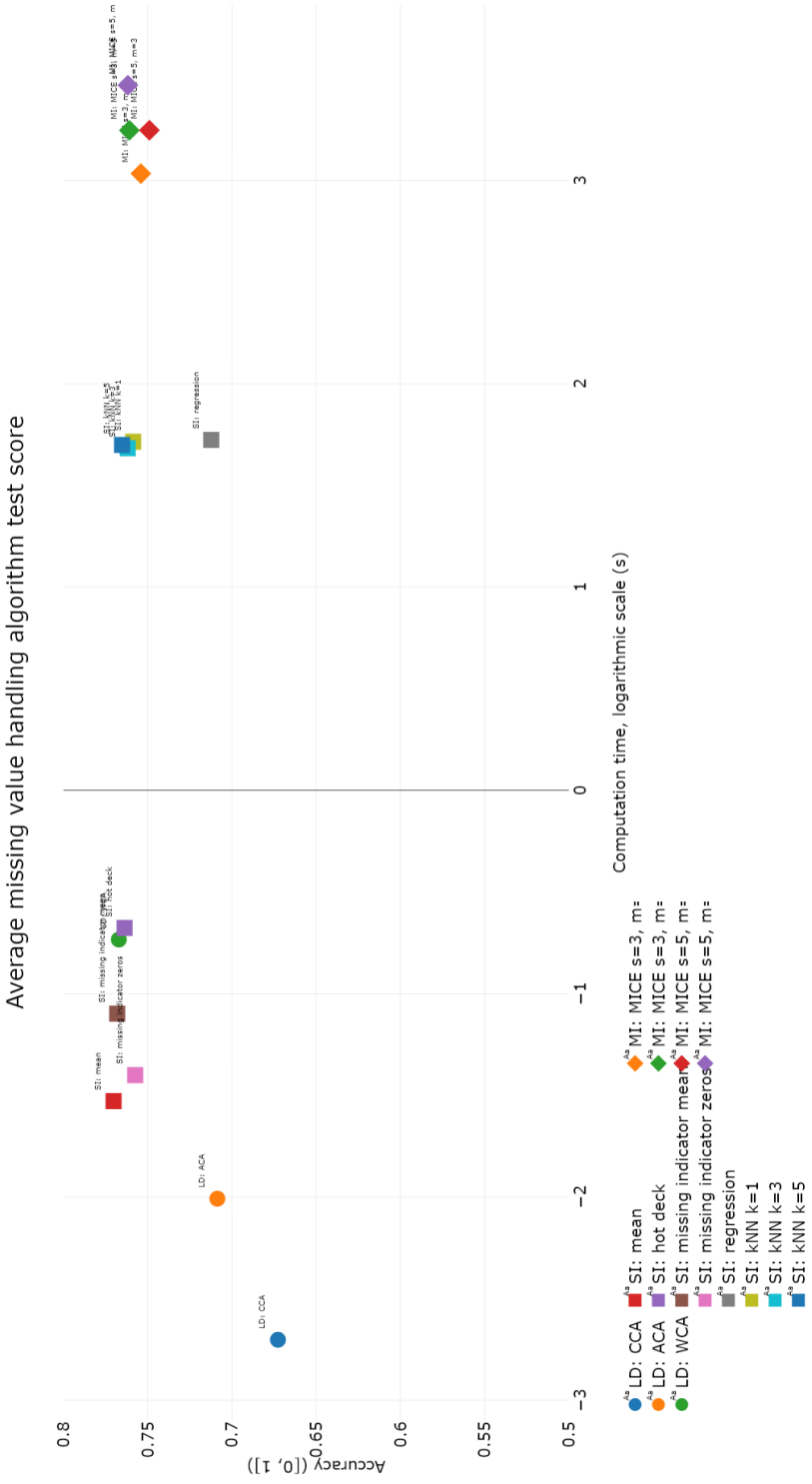


Figure 5.6: A plot showing the average accuracy for the datasets classified after using a missing value handling algorithm. The x-axis shows the computation time for the missing values, done on a 10-logarithmic scale due to the big differences between them. The y-axis shows the accuracy on a scale of 0 to 1. The explanation of every data point can be found both in the text on the top right of the data point as well as in the legend.

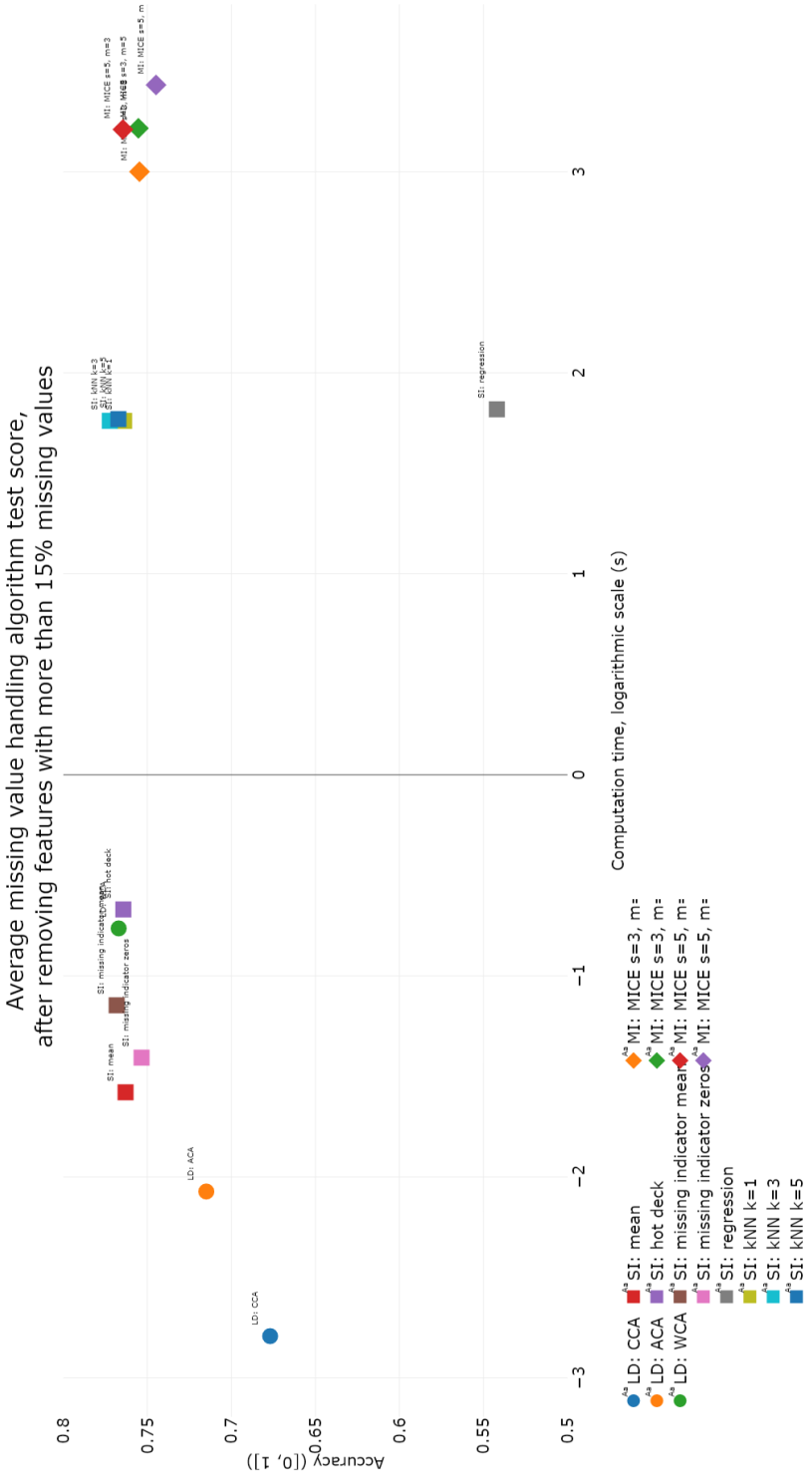


Figure 5.7: A plot showing the average accuracy for the datasets classified after using a missing value handling algorithm and after removing all features with more than 15% missing values. The x-axis shows the computation time for the missing values, done on a 10-logarithmic scale due to the big differences between them. The y-axis shows the accuracy on a scale of 0 to 1. The explanation of every data point can be found both in the text on the top right of the data point as well as in the legend.

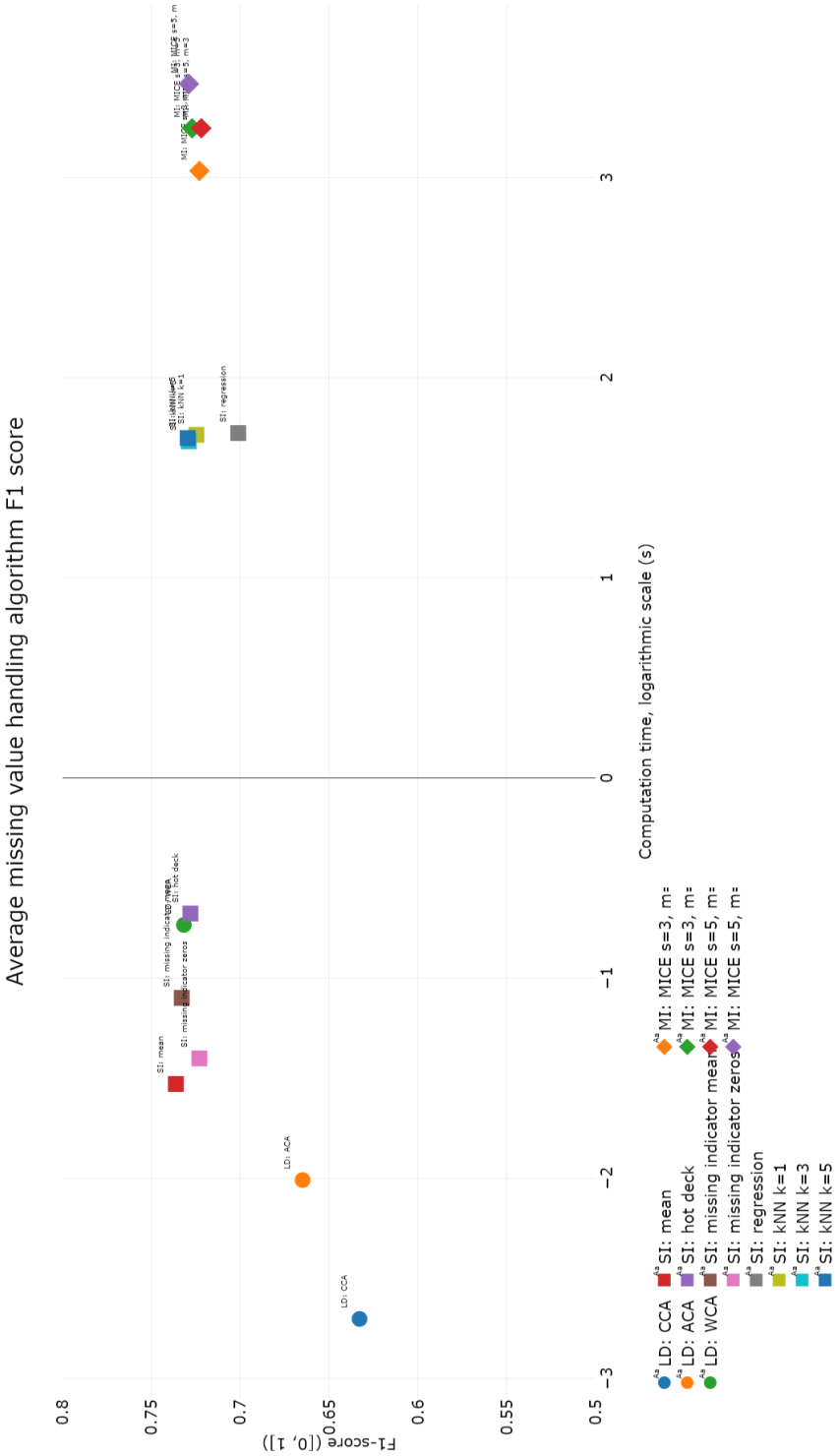


Figure 5.8: A plot showing the F1-score for the datasets classified after using a missing value handling algorithm. The x-axis shows the computation time for the missing values, done on a 10-logarithmic scale due to the big differences between them. The y-axis shows the accuracy on a scale of 0 to 1. The explanation of every data point can be found both in the text on the top right of the data point as well as in the legend.

Table 5.7: The classification results of the datasets and the average after removing all features with missing values, as well as the best results after handling missing values.

	Without missing values		**Best classification**		
Dataset	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**	**Algorithm**
Hepatitis	0.79	0.75	0.87	0.86	Mean imputation
Cirrhosis	0.56	0.50	0.57	0.52	– 1. CCA 2. Missing indicator mean 3. Missing indicator zeros
Cervical	0.74	0.64	0.89	0.83	– Mean imputation
Average	0.70	0.63	0.77	0.74	Mean imputation

The values of the F1 score (Figure 5.8) are lower than the accuracy (Figure 5.6), therefore the F1 score is also used in comparisons. The lack of difference in the accuracy in classification with (Figure 5.7) and without features with more than 15% missing values (Figure 5.6), shows the lack of contribution of these values in the classification. Even though the distribution of the features improves (Subsection 5.5.1), the classification does not show any difference. This was not the case for the dataset with regression imputation, though, as the accuracy severely dropped after removing features with more than 15% missing values from 0.72 to 0.54. Apparently important information was removed for the regression imputation to be successful.

When looking at separate methods, the difference in computation time can immediately be seen. List deletion algorithms and mean, missing indicator and hot deck single imputation algorithms all are very fast (less than a second), followed by k-nearest neighbour and regression single imputation (between 10 and 100 seconds) and at last multiple imputation methods (more than 1000 seconds). All imputation methods and WCA give the best accuracy, with little difference among them. The imputation methods and WCA all show an improvement compared with the accuracy when using only features without missing values (Table 5.7), therefore using features with missing values does improve accuracy. The size of improvement differs per dataset, though, as the improvement for the Cervical dataset is 10 times higher than the improvement of the Cirrhosis dataset.

Since the accuracy and F1-score of the datasets are significantly different (Table 5.7), the results are also plotted per dataset separately (Appendix F.2). The low accuracy for regression imputation seems to originate mainly for the Cervical Cancer dataset. This can be a result of features in the Cervical Cancer dataset mostly consisting of boolean values. Hot encoded boolean values contain less information for regression, making it harder to find good imputation values.

The imputation methods show some differences in classification quality when looking at the results for the datasets separately. The Cirrhosis dataset shows hardly any improvement after using missing value handling algorithms, but its best results are when using a missing indicator method. The Hepatitis dataset result on the other hand is significantly worse when using the missing indicator method. This is supported by the bias evaluation (Subsection 5.5.1) and the dataset description (Subsection 5.2.1), as both showed that data in the Cirrhosis is confirmed MAR, whereas the data in the Hepatitis dataset was least likely to be MAR. This means that when data is very likely to be MAR, missing indicator value is more likely to be beneficial to the outcome. Aside from that, the kNN, regression and MICE imputation methods seem not worth the additional computation time for these datasets, even though in theory these methods make a more detailed guess of the outcome.

5.6 Discussion

Multiple studies have been performed to analyse missing value handling [21, 23, 130, 131, 133, 134]. Most of these focus on complex imputation methods in cases for which imputation is very important. For initial analyses however, less complex methods help more in understanding what should be done and are therefore the basis of further investigation. Therefore, this research is more focused on imputation methods that are easier to understand and more simple to use.

The datasets used in these experiments are datasets with actual missing values. Whether the features with missing values are important for predicting the output however is unknown. Therefore the usefulness of either list deletion or imputation is unknown, too. This is visible in the difference in quality improvement over the datasets, the Cirrhosis dataset having a marginal improvement and the Hepatitis and Cervical Cancer datasets having a significant improvement using imputation methods.

5.7 Conclusions

When comparing Hypothesis H1 to the results, the hypothesis was confirmed for the most part. CCA showed that definitely not all missing values were MCAR and therefore that most features needed more than simply removing instances. WCA showed some improvements, but not for every distribution. The imputation methods showed better results than the list deletion methods. Except for MICE and mean imputation, all were able to remove the distribution bias for all features, but the ones with more than 90% missing values. Interestingly MICE did not perform as well as initially thought. Also, the mean imputation performed well when imputing the mean, median or mode, but also created a bias in the variance of the data and therefore performed poorly on high ratios of missing values.

Even though most imputation methods were able to correctly preserve the feature distributions for features with less than 40% missing values, the quality of distribution preservation was lower for feature with at least 15% missing values. Therefore features with at least 40% missing values should be removed and features with more than 15% missing values should be recommended to be removed.

Hypothesis H2 was only partly right. The list deletion algorithms performed worse than the imputation algorithms, as was expected. Interestingly though, the advanced missing value handling algorithms (MICE, kNN imputation, regression imputation) did not show better results than the relatively simple algorithms (mean imputation, missing indicator imputation). Combined with a significantly higher computation time for the advanced missing value handling algorithms, the simple ones outperformed with these datasets. That part of Hypothesis H2 is therefore disproved.

As discussed, the datasets themselves played a huge part in the quality evaluation. Intuitively, precise imputation with the advanced missing value handling algorithms should outperform the less precise imputation of the simple algorithms. If the more precise imputation is done for an irrelevant feature value, though, the extra information will not produce a better output. Future research can be done on datasets with missing values that are known to be very important for the result, to find out if Hypothesis H2 does hold for these datasets.

Chapter 6

Dataset Exploration

6.1 Introduction

Before actual analysis can be done, a course of action needs to be chosen and the dataset usually needs to be preprocessed. This preprocessing can vary from removing redundant values to normalizing data. The course of action can vary from a type of quality testing to starting with a certain analysis technique. To find the right types of preprocessing and data analyses, initial analysis can be done. The goal of this research is *to find an initial exploration technique that helps finding suitable preprocessing algorithms and provides directions for data analysis.*

A possible way of finding out possible and suitable preprocessing techniques is by using meta-features. Meta-features are dataset specific values that have information on a certain aspect of the dataset [152] (Appendix G.1). With the use of the information available in these meta-features suitable preprocessing and analysis techniques can be found and help the scientist in its research.

6.2 Background

First two datasets are introduced. These datasets are different in multiple ways and several preprocessing and analysis techniques are known to be suitable for these datasets. Secondly, multiple well known preprocessing and analysis techniques are discussed that are used often in the initial phases of research.

6.2.1 Datasets

Two different datasets are used to test the exploration. Each of the datasets has different properties, which makes them useful for testing a wider variety of meta-features. The datasets chosen are the mass spectrometry micro-organisms dataset and the clinical hepatitis dataset (Subsection 3.1):

- *Micro organisms mass spectrometry dataset*
  This dataset is created from mass spectrometry data. Mass spectrometry data has peaks in its data, most values are 0 or close to and a few peaks have a much higher value. This indicates that feature distributions can be very different. Also, the number of features is quite high, and most likely a smaller feature selection can be made.

- *Hepatitis dataset*
  The mortality rate of hepatitis was tested using 19 features over 155 samples. Two class types are distinguished as "died" and "lived" with attributes originating from both clinical and survey samples. On top of that, missing values are known to be present. Two articles that used this dataset were written [65, 66]. This hepatitis dataset is a clinical dataset and

consists of two types of data and therefore potential differences in types and distribution are present.

6.2.2 Preprocessing and Analysis

Before a dataset is properly used in analysis, several possible issues must be addressed. An overview of challenges for biomedical datasets is given earlier (Section 3.1) and several examples on the data heterogeneity, quality and dimensionality challenges are given here. All of these issues have techniques to overcome them. A selection of these issues is discussed, together with ways to find those issues and example methods to address them.

Feature Types

The first issue can be found in heterogeneity within a dataset. Two different feature types are usually found in datasets, categorical and numerical data. Analyses that are based on making choices between different categories prefer categorical data. Examples are Bayesian models [153] and decision trees [154]. These choices are quickly made, due to the already made divisions between the categories. Analyses that use numbers to compute the output prefer numeric data. A good example for this is regression analysis [155] that creates functions for which a numeric value is input to directly compute the output. This analysis does not limit the possibilities in choices and therefore looks to a bigger spectrum of possibilities.

The difference in feature types usually can be seen by the data type of the values. If the data type consists of only numbers, the feature is most likely numeric. If feature values are text based however, the feature is more likely to be categorical.

A dataset can of course also consist of multiple data types. This makes analyses incompatible, because techniques usually only requiring categorical or numeric data. This incompatibility can be removed by transferring one type into the other. Categorical data can for example be changed into numeric data by hot encoding [156], creating one feature for every category and giving every feature value a 0 if that is not the category and a 1 if it is. This can create many additional features, however does remove potential bias by adding a non-existent ordering. Numerical data can be changed into categorical data by splitting up the interval in bins which one can put the numeric values in [157]. Every interval then becomes a category, for which ordering is lost but closely related values are kept together.

Feature Importance Imbalance

A dataset contains measurements for multiple features. When multiple features are numeric these measurements can be very different in intensity due to multiple reasons, such as the choice in units or measurements being on different scales. This difference in intensity reduces the data quality and can create problems in measurements due to bigger values possible having bigger effects on the outcome. Sensitivity can cause problems as well, not being able to process smaller differences between values within one feature than within the other. This creates a measure of unbalance, because features become more important only due to distribution specifics [158, 159].

The detection of possible unbalanced features can be done by comparing distributions properties of a dataset. If distribution specifics such as mean and variance differ greatly, the features are possibly unbalanced, creating bias in analyses.

The removal of this bias is usually by standardisation or normalisation. Standardisation changes distributions in such a way that the distribution specifics become equal. Usually that means that the mean and standard deviation or both are set to a standard value, for examples mean $\mu = 0$ and $\sigma = 1$. Normalisation similarly puts all values of a distribution on a predefined interval, for example interval $[0, 1]$ for which the minimum value becomes 0 and the maximum becomes 1 [158, 159].

Output Imbalance

Every instance is labelled with an output value. These labels are distributed over different classes for classification data. If the classes do not occur with a similar frequency, the analysis techniques may give biased results. These analysis techniques will focus more on classes that are overrepresented and modify the result accordingly. In cases for which misclassification is not equally important for every output class, additional measures must be made to properly show that. This mismatch in data information and analysis reduces the output of the quality.

This output imbalance can be found before actually starting the analysis by looking at the class probabilities. A proper representation would mean that all classes have roughly the same number of instances.

Addressing this imbalance in the analysis of the data can be done by choosing different scoring functions for the quality measurement. Choosing the F1-score over accuracy for example reduces the potential bias in output imbalance. Cohen's kappa removes the agreement by chance, therefore removing initial bias.

Missing Values

The data quality is hardly ever perfect. Mistakes have possibly been made during the design, or entries were incorrect due to errors by the designer. Another problem often occurring in dataset is the presence of missing values. Mistakes can be made somewhere in the production leading into missing entries and most analysis techniques can not work with data including missing values.

Missing values usually are easily spotted. Most of the time a specific value is given to instances, such as 'None', 'NaN' (not a number) or just an empty entry. Other times it is a default value, such as a 0 or a default text value. This detection usually is not a big issue. More important is the way to cope with these missing values.

Several techniques are available to cope with missing values (Chapter 5). Choosing between these techniques may differ between the ratio of missing values present per feature and instance and mainly focuses on either deleting entire features and instances or imputation of a good representing value.

Feature relevance

The dataset consists of a certain number of instances and features. The scientist usually rather has a high number of instances, so the results are more accurate. A higher number of features also means more information is available. The number of useful features usually is limited, however. Moreover, if the number of features far exceeds the number of instances, the data dimensionality can become an issue. Bias may be created due to analysis algorithms not being able to filter out the irrelevant features with the abundance of information.

In case of a feature abundance, feature selection can be useful to remove irrelevant features. Whether feature selection might be useful first can be seen by dividing the number of features by the number of instances, also known as the dimensionality. If the dimensionality is higher than one, feature selection can be used to reduce it. Aside from a high dimensionality, the presence of irrelevant features can be found in several ways:

- *Potential information*
  Every value holds potential information that might contribute to the final product. The potential information can be measured by for example the entropy [81] (Appendix G.1). If the potential information is very low for one or multiple features, they are worth the consideration of being removed.

- *Predictive power*
  Aside from the potential information present, this information will only be useful if relations to the output are present. Mutual information is an example that shows that, by not only looking at the information of the features, but also comparing it with the information of

the output. If the information present in the feature can be used to explain the output, the mutual information between the feature and the output will be higher [84] (Appendix G.1).

Several feature selection techniques are available (Chapter 4). These techniques show different possibilities that try to preserve as few features as possible and still show good results.

Multicollinearity

When looking at features separately, feature selection on feature relevance is effective. This usually does not take relations between features into account. A lower data quality indicates that sometimes features are highly related with each other. This results in overlap in the information present for the outcome. The presence of relations between features is called multicollinearity and reduction of multicollinearity will create more effective information per feature.

Detection of multicollinearity is in principle very simple. All features with each other can be compared to find out how high the relation between the features is. In practice this can be done efficiently for a low number of features, but combinatorial explosions can occur when the number becomes higher, as the number of combinations becomes n^2 with n being the number of features. Therefore this must be done with care for bigger datasets. An example of finding multicollinearity is correlation (Appendix G.1) that compares the difference between the features per instance with the mean difference. Another example is PCA (Appendix G.1) that computes how much variance can be combined in as few newly created features as possible, showing the relation in variances between features.

PCA is an effective way of removing multicollinearity. It does however use newly created features which makes linking input and output directly much harder to understand. Several feature selection techniques also take multicollinearity into account, such as wrapper methods (Chapter 4). Also clustering features can contribute to less multicollinearity, combining the cluster into one feature [160].

6.2.3 Meta-features Package

An implementation that focuses on extracting meta-features from dataset is the package *metalearn*. This package implements most of the aforementioned meta-features. It requires *Python 3.6* and eleven separate packages, most of them being used very often in data analysis (for example *NumPy*, *SciPy*, *scikit-learn* and *Pandas*). All of these packages are available in *Anaconda*, which makes the usage of *metalearn* very easy.

Usage of this package is quite intuitive and for the user mainly consists of one class *Metafeatures*. After initialization of this class, it can compute the meta-features of a dataset with the method *compute*. The input for this method should consists of a dataset and an output labels. Additionally an indication for every feature of the dataset being categorical or numeric can be added if this is not clear from the feature types, a subset of all meta-features can be specified if only those are required and some implementation specific parameters can be specified if needed.

The number of meta-features that can be extracted is very long. Themes which they can be put in are given, as well as a brief explanation. Extra information on the meta-features is available if needed (Appendix G.1) The names of the meta-features are self-explanatory and therefore not given for a better explanation:

- **Basic meta-feature themes**

  1. *Data size*
     The number of instances, features and classes and the dimensionality (number of features divided by the number of instances).
  2. *Data types*
     The number and ratio with regard to the total number of features of both the numeric and the categorical features.

3. *Missing values*
   The number and ratio of the total number of missing values and the number and ratio of both instances and features with missing values.
4. *Output type*
   The average probability of each class, as well as its standard deviation, minimum and maximum and the size of the minority and majority class.
5. *Feature cardinalities*
   The mean, standard deviation, minimum and maximum cardinality of both categorical and numeric features.

- **Statistical Meta-feature themes**

1. *Feature distributions*
   The mean, standard deviation, skewness and kurtosis are all tested for numeric features. The values of all of these features are shown by the mean, standard deviation, minimum, maximum and the first-, second- and third quartile after combining them.
2. *Principal component analysis*
   The ratio of variability explained by the first three principal components and the size of the first three eigenvalues.

- **Information-theoretic meta-feature themes**

1. *Entropy*
   The attribute and joint entropy are computed for both categorical and numeric features. Of these entropies the mean, standard deviation, minimum, maximum and the first-, second- and third quartile values are recorded.
2. *Mutual information*
   For both categorical and numeric features the mutual information is tested. From this the mean, standard deviation, minimum, maximum and the first-, second- and third quartile values are recorded.
3. *Entropy and mutual information*
   The equivalent number of categorical and numeric features is given to explain the class entropy. Also the signal to noise ratio for the dataset is given for both categorical and numeric features.

- **Landmarking meta-feature themes**
  The error rate (1 - accuracy) and the Cohen's kappa κ are computed for a selection of machine learning algorithms. This selection consists of Naive Bayes, k-nearest neighbours with $k = 1$, decision stump (decision tree with one node), random tree (decision tree with random splits and a depth of 1 to three nodes) and linear discriminant analysis.

Advantages of this package is the ease of use. The variety in the number of meta-features is quite big and much information can be retrieved from them. Since all of these meta-features are a numeric value themselves, they can be easily used in a subsequent algorithm that can makes use of these meta-features.

The number of meta-features is a disadvantage for a user, when using it directly. It is hard to find the relevant information from these values as there are so many of them. Also the conclusions that can be taken from these meta-features are sometimes a bit hard to grasp. Additional plots of distributions as well as several examples that indicate this would make it easier to understand the meta-features.

6.3 Hypotheses

The main goal of this chapter is to find out which preprocessing and analysis techniques should be used for datasets. To find those, several meta-features are provided, as well as the package *metalearn* that computes several of those meta-features. For this goal, two hypotheses are made. The first hypothesis is the following:

H1: All mentioned issues, except for multicollinearity, are properly addressed in the meta-features form the package metalearn. Multicollinearity is only partly addressed.

Most of the issues are expected to be addressed with the proposed meta-features. The feature types, feature importance imbalance, output imbalance, missing values and feature relevance all have known meta-features that can indicate these possible issues. Multicollinearity is only partly addressed with PCA, which is expected to not give enough insight in multicollinearity. The second hypothesis is based on follow up after finding the meta-features:

H2: The understanding of the outcome of the meta-features is limited and further information would improve this.

The links between the issues and meta-features are given. The understanding of these links are not provided, though. Figures and examples to explain these links should be useful to add to these meta-features to further solidify the choice of using a preprocessing or analysis technique.

6.4 Methods

To test both hypotheses, two different analyses were created. The first analysis focuses on the results when only using the *metalearn* package, without any additions to it. The second analysis focuses on adding several additional results to support the meta-features and create better understanding.

Due to previous experiments with the two datasets, the issues for both datasets are known and should be represented in the meta-feature analyses:

- *Hepatitis*
  This dataset consists of a mixture of two data types. The features and classes are imbalanced and missing values are present.
- *Micro organisms*
  This dataset has a highly imbalanced feature importance, as well as imbalances in the multiple classes present. A big variation in feature relevance is present, as well as multicollinearity.

6.4.1 Existing Meta-features Evaluation

The first analysis is done to investigate the dataset issues with the *metalearn* package (Section 6.2.3) and all available meta-features in that package. The two datasets used are based on micro organisms and hepatitis (Section 6.2.1) and both contain known issues. The meta-features are then compared with those issues to find out if those can be extracted from them.

6.4.2 Additional Exploration Evaluation

For the second analysis several additional computations are done. These additional computations focus on correlation, low cardinalities, outlier examples and at last visual plots.

The designer of *metalearn* already initialized finding correlations between the values. The meta-features for this correlation were added for datasets with fewer than 1000 features, to prevent

combinatorial explosion problems. The newly created meta-features are the mean and the standard deviation of this correlation. On top of that the three correlation combinations with the lowest and the highest value for this correlation are added.

Sometimes features only have one distinct value as output having a cardinality of one. This feature cannot contribute to the analysis, as there is no information present. To find a list for all features with only one distinct value an additional method in the package is made that counts the number of features with the lowest cardinality. Aside from this count, also the features having this cardinality are given to potentially remove them.

Outlier examples are shown, to give an initial insight in the data. These examples show which features at least require more attention during analysis or even should be removed. These outliers are given for the following examples:

1. The instances and features with the most missing values.
2. The output classes that have the lowest or highest probability.
3. The categorical and numeric features with the lowest or highest cardinality.
4. The categorical and numeric features that have the lowest or highest attribute entropy.
5. The categorical and numeric features that have the lowest or highest joint entropy.
6. The categorical and numeric features that have the lowest or highest mutual information.

In multiple instances, the number of values can much better be explained by using some kind of plot. For the numeric distribution values for example, a boxplot for the means, standard deviations, skewnesses and kurtoses combines all of those meta-features. A histogram shows these shows these meta-features visually for the class counts as well. At last the outlier distributions are better understandable when shown with a boxplot or histogram. Therefore multiple visual plots are made to show this. This plots are generated automatically for the user after adding it to the meta-feature list.

6.5 Results

The results are also split into two different sections, corresponding to the methods.

6.5.1 Existing Meta-features Evaluation

All meta-feature values are computed and given for the hepatitis (Appendix G.2) and micro-organisms (Appendix G.3) datasets. The results are also split into two sections, each dataset evaluated in one subsection. As can be seen, the number of meta-features is very high and therefore difficult to find the relevant ones. Therefore a selection of meta-features is made, based on the (subjective) estimation that a conclusion can be derived from those.

Hepatitis Dataset

First the important meta-features of the hepatitis dataset are given:

- *Data size* Number of instances: 155
  Number of features: 19
  Number of classes: 2

  The number of instances is somewhat limited, but is not too low to hurt the analysis. The number of features is low enough to really need feature selection and the number of classes shows a boolean result.

- *Data types*
  Number of numeric features: 6
  Number of categorical features: 13

  Both numeric and categorical features are present. This means the type of analysis will matter and possibly hot encoding or binning needs to be used for categorical features for further analysis.

- *Missing values*
  Number of missing values: 167
  Number of instances with missing values: 75
  Ratio of instances with missing values: 0.48
  Number of features with missing values: 15
  Ratio of features with missing values: 0.79

  Values are missing and these missing values are spread out significantly over the instances (ratio of 0.48) and features (ratio of 0.79) and therefore something more than deletion would be advised to do.

- *Output type*
  Minimum of class probability: 0.21
  Maximum of class probability: 0.79

  The two classes deviate significantly in class probability (0.21 to 0.79), so the dataset is not balanced. This can create possible bias when using plain accuracy in quality measurements.

- *Feature cardinalities*
  Minimum categorical feature cardinality: 2
  Maximum categorical feature cardinality: 2

  The categorical feature cardinality shows that all categorical features are boolean, which limits the possible information present in these categorical features.

- *Feature distributions*
  Mean of means numeric features: 49.92
  Standard deviation of means numeric features: 42.57
  Mean of standard deviations numeric features: 29.74
  Standard deviation of standard deviations numeric features: 29.73
  Mean of skewnesses numeric features: 1.28
  Mean of kurtoses numeric features: 4.49

  Both the means and the standard deviations of the numeric features differ significantly from each other. Therefore scaling on both means and standard deviation per feature would be balance out feature importances. Also the skewness and kurtosis are both much higher than expected, showing that these values do not follow normal distributions.

- *Principal component analysis*
  Explained variation component 1: 0.69
  Explained variation component 2: 0.26

  Apparently more 95% of the variance in the data can be explained in only two principal components. Therefore apparently a lot of multicollinearity is present.

- *Entropy and mutual information*
  Mean mutual information categorical features: 0.04
  Mean mutual information numeric features: 0.08
  Categorical noise to signal ratio: 13.60
  Numeric noise to signal ratio: 11.90

  The mutual information of the categorical features is low, averaging at a mutual information of 0.04. The numeric features contain about twice as much information with an average mutual information of 0.08. Also relatively a lot of noise is present, compared with the signal.

- *Landmarking*
  Naive Bayes error rate: 0.22
  Naive Bayes kappa: 0.48
  Decision stump error rate: 0.21
  Linear discriminant error rate: 0.17
  Linear discriminant kappa: 0.46

  The error rates can show possible bias, as indicated from the deviation in class probabilities. Of all algorithms, the linear discriminant and Naive Bayes already show an error rate of around 0.2 and a kappa of 0.5, showing potential for a good machine learning model. The decision stump and random tree with depth of one shows that only using one feature can already give a quality of 0.8 (1 - the error rate), being quite high.

The meta-features confirm the presence of missing values and show that these missing values can cause a problem, due to their spreading. The feature types (categorical and numeric) both are found and specific issues that need focus for the categorical values (low cardinality) and the numeric features (distribution differences) are found, too, as well as the information they provide. Additional useful information is found in PCA, showing that the 19 features share much variation and that using simple analysis techniques show promising results. A better focus on locations of multicollinearity and mutual information is missing however, as of these 19 features no specific information on the features is present.

Micro-organisms Dataset

Secondly the most important meta-features are given for the Micro-organisms dataset:

- *Data size*
  Number of instances: 571
  Number of features: 1300
  Number of classes: 20
  Dimensionality: 2.28

  The number of features for this dataset is very high in comparison with the number of instances. The dimensionality is preferably much lower, which already indicates that feature selection would be a good approach for this dataset. The number of classes in the output is high in comparison with the number of instances as well, so attention to the class probabilities is useful.

- *Data types*
  Number of categorical features: 0

  The number of categorical features is 0, so only numeric features are present. This indicates that there is no categorical information in the data.

- *Output type*
  Minimum of class probability: 0.02
  Maximum of class probability: 0.1

  The classes are not balanced. This indicates that using the accuracy might give bias.

- *Feature Cardinality*
  Mean cardinality numeric features: 45.29
  Minimum cardinality numeric features: 1

  The mean cardinality is lower than expected for numeric values with 571 instances. It also shows that at least one feature has a cardinality of 1 and can therefore be removed from the data, not having any meaning.

- *Feature distributions*
  Mean of means numeric features: 67437.29
  Standard deviation of means numeric features: 191740.68
  Mean of standard deviations numeric features: 343176.13
  Standard deviation of standard deviations numeric features: 700237.17
  Mean of skewnesses numeric features: 10.67
  Mean of kurtoses numeric features: 177.27

  The mean and standard deviation of both the means and standard deviations of the numeric features show that scaling is needed to balance the feature importances. Also, the mean skewness and kurtosis show that the distribution is highly irregular and cannot be seen as a normal distribution.

- *Entropy and mutual information*
  Maximum mutual information numeric features: 0.27
  Third quartile mutual information numeric features: 0.06
  Equivalent number of numeric features: 70.27
  Numeric noise to signal ratio: 1.64

  The maximum mutual information is relatively higher than the third quartile mutual information. This can indicate that several outlier features are significantly more useful in predicting the output. The equivalent number of numeric features is quite high and indicates on average a high number of features is needed to predict the output. At last there seems to be more noise than signal, so noise reduction might be useful to add.

- *Landmarking*
  Naive Bayes error rate: 0.20
  Naive Bayes kappa: 0.79
  Decision stump error rate: 0.83

  The Naive Bayes error rate and kappa are significantly better than for the other algorithms. So analysis in the area of Naive Bayes seems like a good start. Also the decision stump error rate shows a very high error rate. This high error rate is due to the splitting method in decision stumps, only being able to split the instances in two groups, even though 20 different groups are present.

The high number of features was known beforehand and the meta-features confirm that this issue needs to be handled. At start it seems that several features can already be removed because of the low cardinality of 1 and the very low mutual information. Also, due to the data being gathered by mass spectrometry the distributions are very different, which indicated normalisation

to be useful. The number of features needed to create a good prediction seems to be high, knowing the equivalent number of numeric features. At last basic Naive Bayes already shows a good result in this, therefore more in depth analysis in that area might be useful.

6.5.2 Additional Exploration Evaluation

All additional computation results are given for the hepatitis and the micro-organisms dataset. The results are also split into two sections, each dataset evaluated in one subsection.

Hepatitis Dataset

The additions are shown and discussed per different addition theme. First the correlation is discussed, secondly the minimum cardinality, followed by the outliers and at last the plots.

- *Correlation*
  Mean correlation: 0.08
  Standard deviation correlation: 0.20

  The mean correlation of the dataset is not very high and indicates that multicollinearity should not create big problems. The relatively high standard deviation does indicate that there might be some features that have a very high correlation with each other. This indicates further investigation.

- *Minimum cardinality*
  Minimum cardinality feature count: 13

  Whereas the minimum cardinality being 2, the minimum cardinality count could already be deduced to 13. This is because all 13 categorical values are boolean.

- *Outliers*

  All outliers are computed and the lowest and highest three outliers are shown (Table 6.1). Some of these outliers seem to need care:

  1. The feature *STEROID* misses 43% of its values, does contain the most information in both attribute and joint entropy, but does not have the most mutual information. The ratio of missing values is much higher in this feature than in the others. Also, considering the number of instances not being high (155), the removal of this feature might be a good idea.
  2. The feature *ALBUMIN* has the highest negative correlation with the features *ASCITES, ALK_PHOSPHATE* and *BILIRUBIN*, whereas the feature *MALAISE* has a high correlation with both *FATIGUE* and *ANOREXIA*. To reduce multicollinearity these features could potentially be removed.
  3. The features *AGE* and *ALK_PHOSPHATE* both have very high potential information, but do not have high mutual information.
  4. The features *PROTIME*, *ALBUMIN*, *BILIRUBIN*, *ASCITES* and *SPIDERS* all have high mutual information and of these *BILIRUBIN*, *ALBUMIN* and *ASCITES* even show a low potential information, so most likely not having a high amount of noise. These five most likely are very useful in predicting the output.

- *Plots*

  The values for the plots are computed and visualized. Whereas these plots do not show any additional information, they do show earlier mentioned problems:

Table 6.1: The outliers for the hepatitis dataset

Outlier Type	Lower bound			Upper bound		
	First Outlier (name)	Second Outlier (name)	Third Outlier (name)	Third Outlier (name)	Second Outlier (name)	First Outlier (name)
Ratio of instances with missing values	–	–	–	0.37 (80)	0.37 (10)	0.74 (49)
Ratio of features with missing values	–	–	–	0.10 (ALBUMIN)	0.19 (HISTOLOGY)	0.43 (STEROID)
Class probabilities	0.21 (1)	–	–	–	–	0.79 (0)
Categorical cardinality	2 (SEX)	2 (ANTIVIRALS)	2 (HISTOLOGY)	2 (SPIDERS)	2 (ASCITES)	2 (VARICES)
Numeric cardinality	30 (ALBUMIN)	35 (BILIRUBIN)	45 (PROTIME)	49 (AGE)	84 (ALK_PHOSPHATE)	85 (SGOT)
Correlation combinations	-0.55 (ASCITES, ALBUMIN)	-0.40 (ALK_PHOSHATES, ALBUMIN)	-0.38 (BILIRUBIN, ALBUMIN)	0.47 (LIVER_BIG, LIVER_FIRM)	0.60 (FATIGUE, MALAISE	0.60 (MALAISE, ANOREXIA)
Categorical attribute entropy	0.33 (SEX)	0.37 (VARICES)	0.39 (ASCITES)	0.68 (LIVER_FIRM)	0.69 (HISTOLOGY)	0.69 (STEROID)
Numeric attribute entropy	0.60 (SGOT)	0.64 (BILIRUBIN)	1.12 (ALBUMIN)	1.19 (PROTIME)	1.27 (ALK_PHOSPHATE)	1.32 (AGE)
Categorical joint entropy	0.81 (ASCITES)	0.82 (SEX)	0.82 (VARICES)	1.14 (HISTOLOGY)	1.16 (LIVER_FIRM)	1.19 (STEROID)
Numeric joint entropy	1.05 (BILIRUBIN)	1.09 (SGOT)	1.47 (ALBUMIN)	1.52 (PROTIME)	1.72 (ALK_PHOSPHATE)	1.80 (AGE)
Categorical mutual information	0.00 (LIVER_FIRM)	0.00 (LIVER_BIG)	0.01 (ANOREXIA)	0.06 (HISTOLOGY	0.08 (SPIDERS)	0.09 (ASCITES)
Numeric mutual information	0.01 (SGOT)	0.03 (AGE)	0.03 (ALK_PHOSPHATE)	0.09 (BILIRUBIN)	0.14 (ALBUMIN)	0.17 (PROTIME)

1. *Class distribution* (Figure 6.1)
   The class distribution clearly visualizes the imbalance between the output classes and can give the user ideas of how to deal with this.

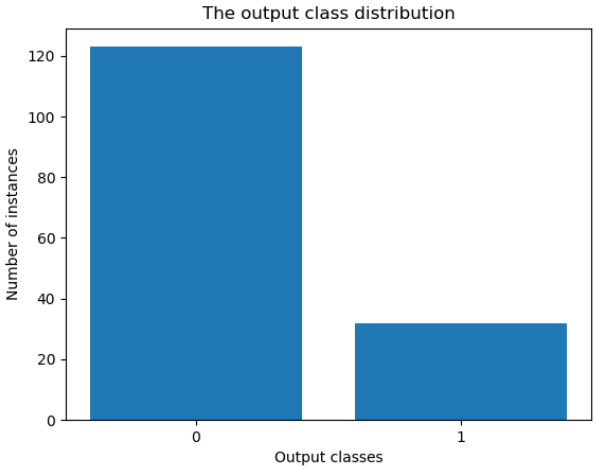


Figure 6.1: The class distribution of the output of the hepatitis dataset.

2. *Numeric feature distributions* (Figure 6.2)
   These distributions show very clearly how the feature values are distributed and that normalisation and standardisation should be considered to remove the feature value imbalance.

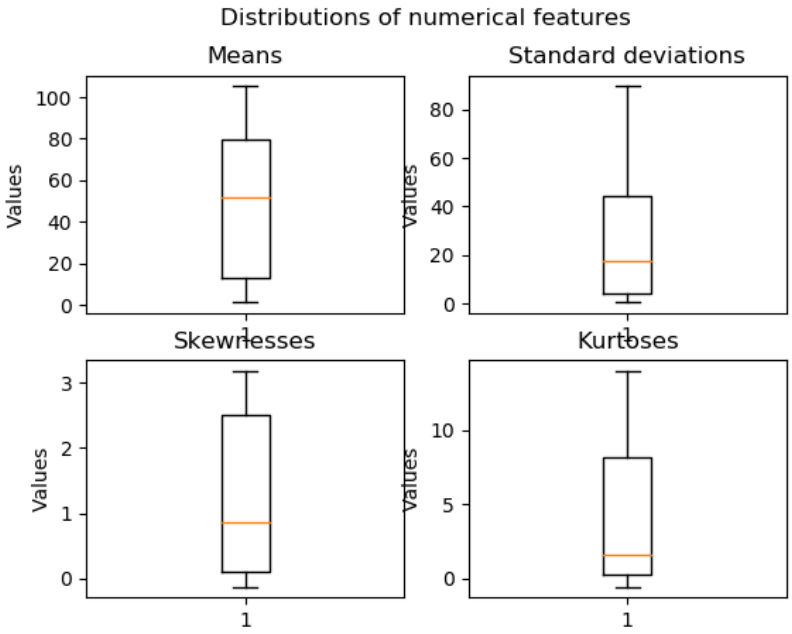


Figure 6.2: The boxplots of the means, standard deviations, skewnesses and kurtoses from the features of the hepatitis dataset.

3. *Categorical and numeric cardinality* (Figures 6.3 and 6.4)
   The distributions for the cardinality outliers mentioned at the start of this section are shown, but not very relevant in this case. No additional information can be found in this case.

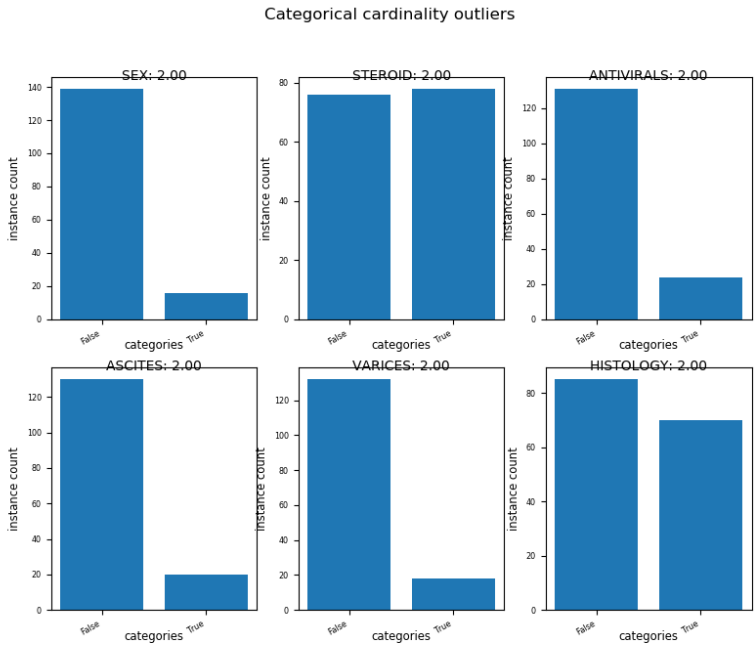


Figure 6.3: Histograms of the categorical features with either the highest or the lowest cardinality in the hepatitis dataset.

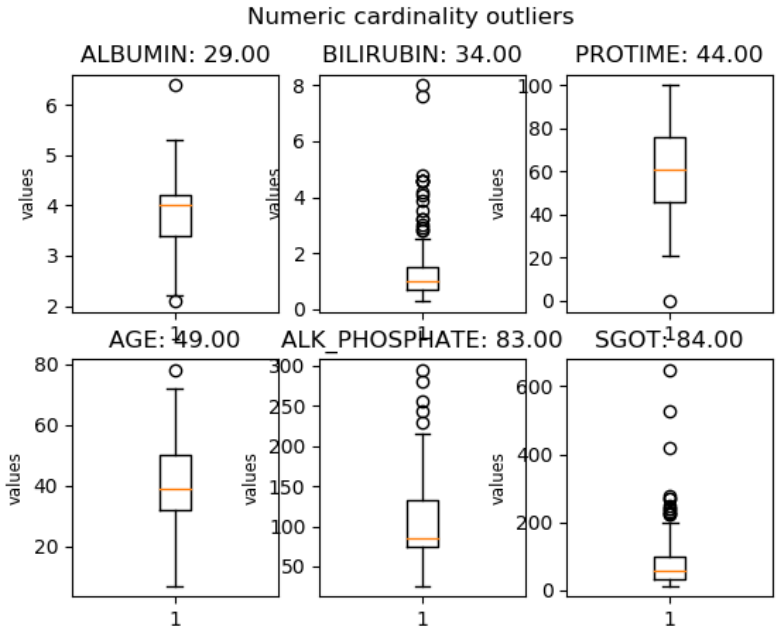


Figure 6.4: Boxplots of the numeric features with either the highest or the lowest cardinality in the hepatitis dataset.

4. *Attribute entropy outliers* (Figures 6.5 and 6.6)
   Especially the figure for categorical distributions show very well how features differ in

entropy. The features *SEX*, *VARICES* and *ASCITES* show a strong bias towards the value FALSE. The other three maximum outliers are far more balanced.

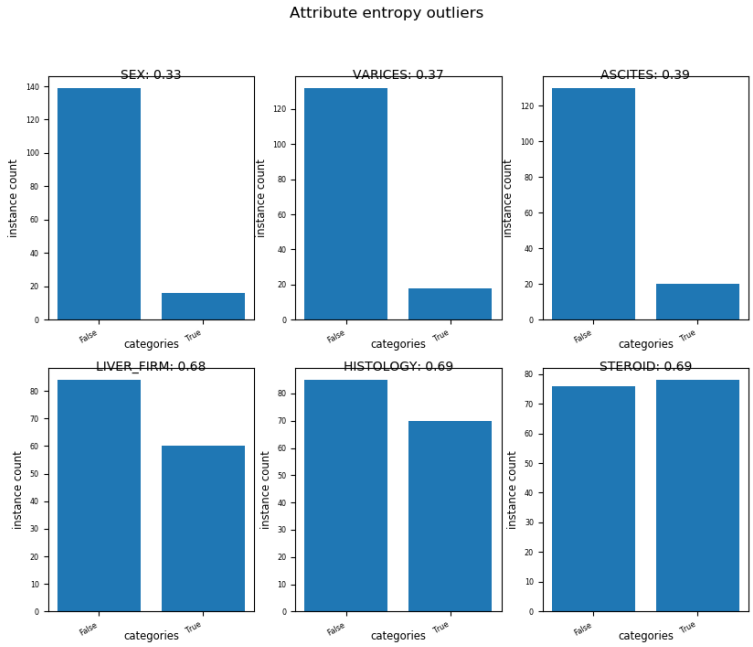


Figure 6.5: Histograms of the categorical features with either the highest or the lowest attribute entropy in the hepatitis dataset.

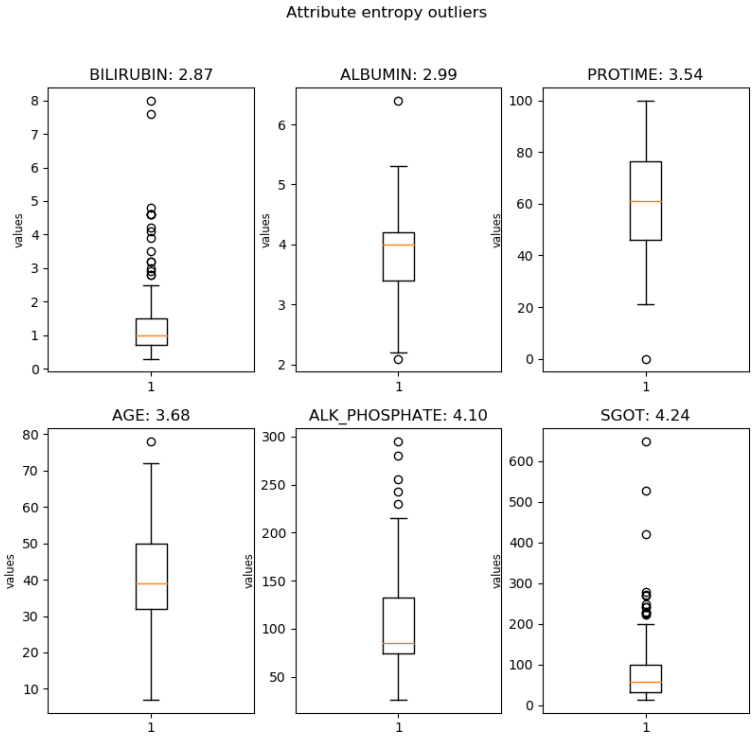


Figure 6.6: Boxplots of the numeric features with either the highest or the lowest attribute entropy in the hepatitis dataset.

Looking at the overall additional results, the following characteristics of *metalearn* are improved with regard to exploration. More insight in correlation is given by using the correlation meta-features and several features are known to most likely be irrelevant or to the contrary very relevant. Also the understanding of the issues is better with the outliers, histograms and the boxplots.

Micro-organisms Dataset

The additions are shown and discussed per different theme. First the correlation is discussed, followed by the minimum cardinality. Thirdly the outliers are discussed and lastly the plots.

- *Correlation:*

  Due to the imposed restriction of combinatorial explosion, correlation will not be checked for more than 1000 features, and this dataset has 1300 features.

- *Minimum cardinality*
  Minimum cardinality feature count: 218

  A total of 218 features have a cardinality of one and therefore do not contain any information.

- *Outliers:*
  (Table 6.2)

  All outliers are computed and the lowest and highest three outliers are shown (Table 6.2). Some results need further discussion:

1. The 20 possible class probabilities show no extreme outliers. Imbalance is still present, but there is no majority problem.
2. The feature *V1261* has the most potential information and has the second to most mutual information and therefore is a feature to keep an eye on.
3. Again the features with one distinct value are found in the lowest outliers.
4. The closeness in entropy and mutual information for the top three features shows that far more features are interesting to look at and that these features would be useful.

Table 6.2: The outliers for the micro-organisms dataset

	Lower bound			**Upper bound**		
Outlier Type	**First Outlier (name)**	**Second Outlier (name)**	**Third Outlier (name)**	**Third Outlier (name)**	**Second Outlier (name)**	**First Outlier (name)**
Ratio of instances with missing values	–	–	–	–	–	–
Ratio of features with missing values	–	–	–	–	–	–
Class probabilities	0.02 (7)	0.02 (6)	0.03 (9)	0.09 (19)	0.09 (3)	0.11 (11)
Categorical cardinality	–	–	–	–	–	–
Numeric cardinality	1 (V1)	1 (V447)	1 (V1090)	259 (V1261)	294 (V719)	296 (V838)
Correlation combinations	–	–	–	–	–	–
Categorical attribute entropy	–	–	–	–	–	–
Numeric attribute entropy	0.0 (V1)	0.0 (V1099)	0.0 (V212)	0.60 (V1163)	0.62 (V1002)	0.83 (V1261)
Categorical joint entropy	–	–	–	–	–	–
Numeric joint entropy	2.91 (V1)	2.91 (V1071)	2.91 (V782)	3.34 (V1163)	3.35 (V1002)	3.49 (V1261)
Categorical mutual information	–	–	–	–	–	–
Numeric mutual information	0.00 (V1)	0.00 (V251)	0.00 (V248)	0.25 (V787)	0.25 (V1261)	0.27 (V1008)

- *Plots*

  The values for the plots are computed and visualized. Whereas these plots do not show any additional information, they do further clarify earlier mentioned problems (Subsection 6.5.1)

  Class count histogram: Figure 6.7

  The class count histogram shows that most of the classes are around 25 to 30 instances. There are however seven classes that have significant more or less instances.

  Distribution boxplots: Figure 6.8

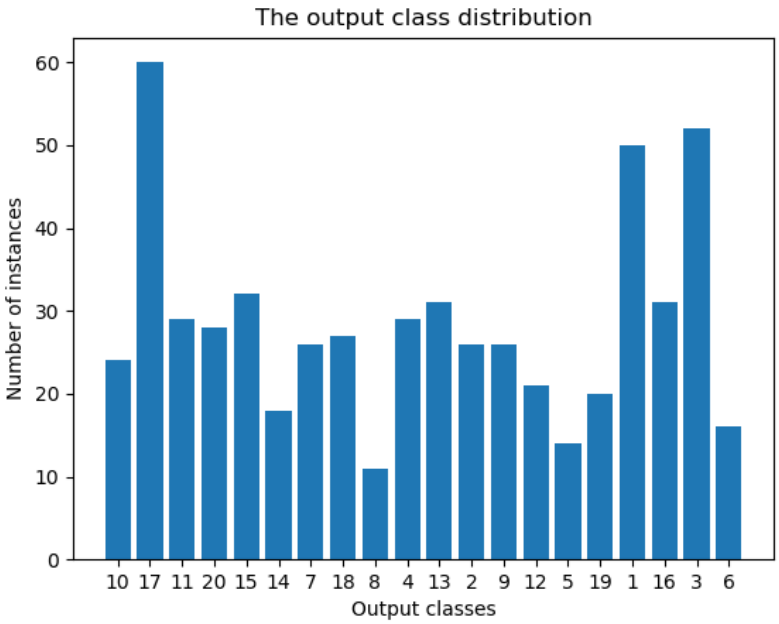


Figure 6.7: The class distribution of the output of the micro-organisms dataset.

These boxplots show very well the difference in values for every feature. The creation of a boxplot of the means and standard deviation and the high number of outliers, show that normalisation or standardisation seems a good preprocessing action.

Numeric cardinality outlier boxplots: Figure 6.9

The numeric cardinality boxplots show very well how the feature values are distributed. Due to this dataset originating from mass spectrometry, most values are 0. All values that are not 0 can become significantly high, with outliers having a factor 10^7. Again the outliers with the lowest cardinality have just one value, 0.

Numeric attribute entropy boxplots: Figure 6.10

The attribute entropy outliers cannot directly be seen in the box plot. This plot does not seem very useful in this case.

First, finding out that 218 features in the dataset that contain no information is very useful in initial understanding of the dataset. The outliers mainly show that most likely many more features are useful in finding a good result. The histograms and boxplots are more useful as they give a better view in the class distribution. They also visualize the distributions of both the means, standard deviations and also the feature distributions, giving more understanding and insights.

6.6 Discussion

There are numerous ways biomedical datasets are gathered and therefore numerous issues that can be found in the dataset. The six discussed issues are well known and therefore a good representation, however there are more issues. For example the issue of having too few instances and trying to generate more is not discussed nor the possibility of anomalies being present in the data. These two issues may be addressed in future research.

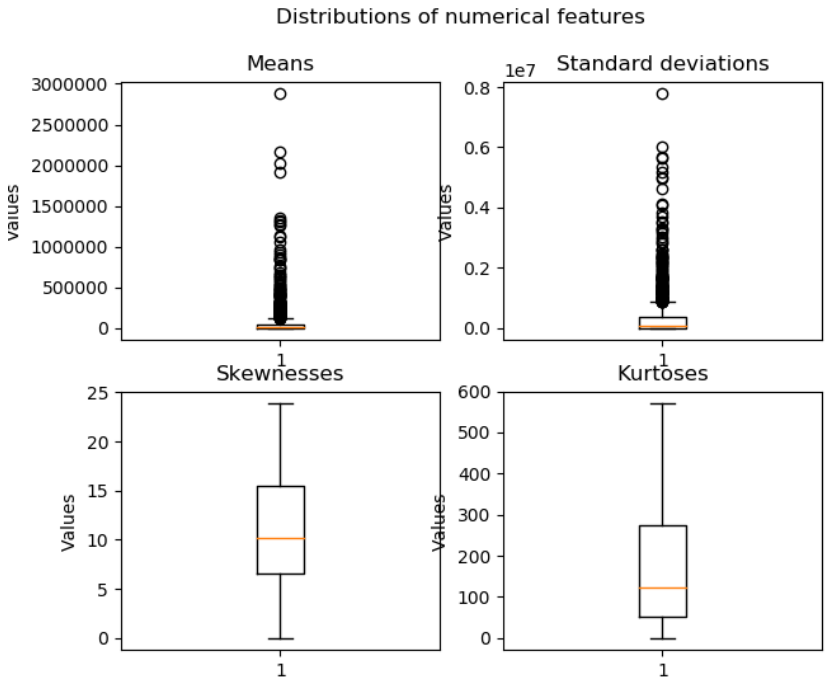


Figure 6.8: The boxplots of the means, standard deviations, skewnesses and kurtoses from the features of the micro-organisms dataset.

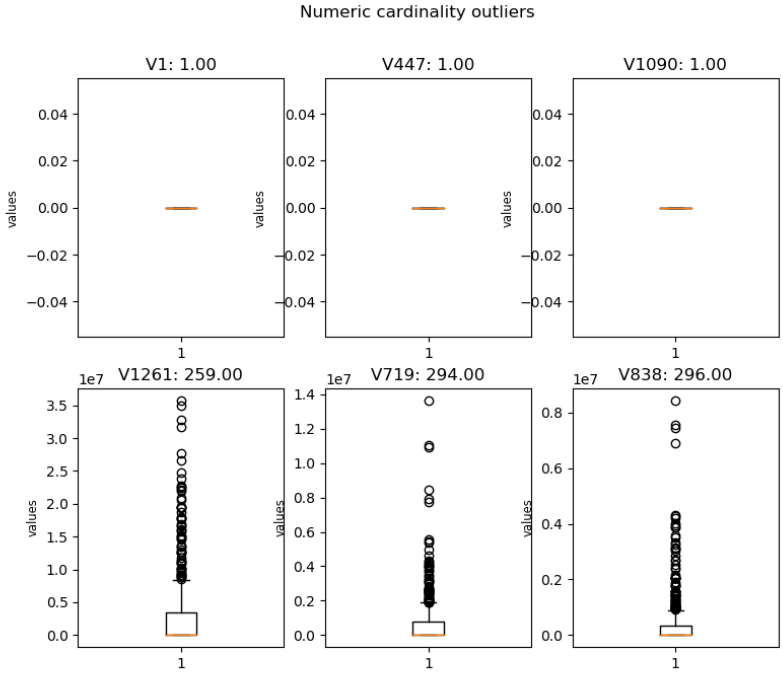


Figure 6.9: Boxplots of the numeric features with either the highest or the lowest cardinality in the Micro-organisms dataset.

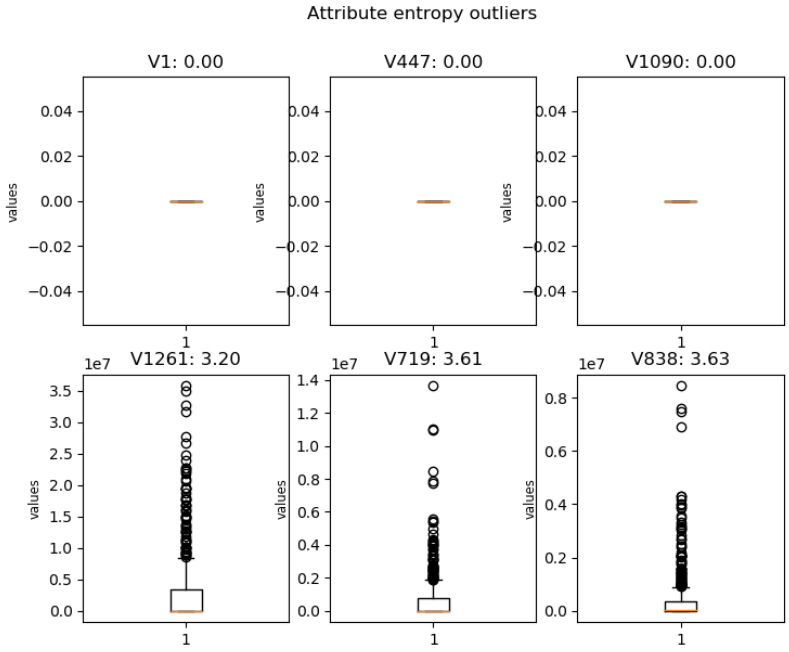


Figure 6.10: Boxplots of the numeric features with either the highest or the lowest attribute entropy in the Micro organisms dataset.

6.7 Conclusions

In this chapter the package *metalearn* was analysed and additional content for the package was made to further improve its capabilities. The analysis was done with regard to the possibility to locate dataset issues with the meta-features present.

The first hypothesis H1 is confirmed, in that all issues except for multicollinearity were found in the two datasets with the current meta-features. Multicollinearity needed more addressing besides these meta-features. The information from the meta-features was tough to understand, though. The sheer number of the meta-features made it hard to find the issues, and the lack of examples and visualization made it hard to understand how these issues are manifested in the dataset. Therefore H2 is also confirmed.

The problem posed in the confirmed H2 and the lack of multicollinearity testing in H1 was partly removed by the addition of new meta-features, outliers and histograms and boxplots. These made a better understanding, showed several features that needed more attention and also showed better if multicollinearity could be a problem for the dataset. Therefore the additions are useful in dataset exploration.

Metalearn with the added changes was able to sufficiently locate dataset issues. On top of that its ease of use and simplicity in understanding result in *metalearn* to be a useful addition to the framework.

Chapter 7

Framework Overview

7.1 Introduction

With the results of the feature selection, missing value handling and data exploration research, a guideline framework is made to incorporate these results. The framework is based on four phases: a dataset type challenge explanation, a dataset exploration, a preprocessing step and automated analysis. Every phase has its own framework questions for which information or methods are available. A layout of the framework is also given to point out all its options (Figure 7.1).

This framework is mainly for guiding a data analyst into the first phases of his analysis. The dataset is understood and preprocessed quicker and some initial machine learning analyses are given. The steps exploration, preprocessing and analysis in the framework are also automated with the simple preprocessing and machine learning algorithms for ease of use. Further analysis in both preprocessing and machine learning however is advised, as usually datasets need a more specific analysis to find good results.

This framework is set up for datasets for which the data is given in a matrix. In this matrix every column corresponds to a feature and every row corresponds to a data sample. Data should be adjusted such that it satisfies the required input format. Finally it is worth emphasizing that the framework is developed with the assumption that classification problems are solved. Some techniques are also applicable for regression however and those parts in the framework can still be used for those.

7.2 Dataset type

Before analysing any of the data, some information can be retrieved from the dataset origin. Different types of datasets usually encompass potential issues being present (Section 3.1). An initial framework question is therefore implied to address these potential issues:

FQ1: What type of dataset is used?

Several different datasets are used and all potential issues should be investigated. Four dataset types however are previously investigated and for each of these types a specific focus on a selection of issues is given. Both the dataset types and the possible extra focus on dataset issues are shown (Figure 7.1).

A *Micro-array dataset*
Micro-array data usually consists of a high number of genes being investigated. Most of those genes are not of use in this work, therefore feature selection is a useful way of removing the irrelevant genes. Due to this high number of genes possible computation time issues can occur in some algorithms. These computation time issues would also be reduced when only

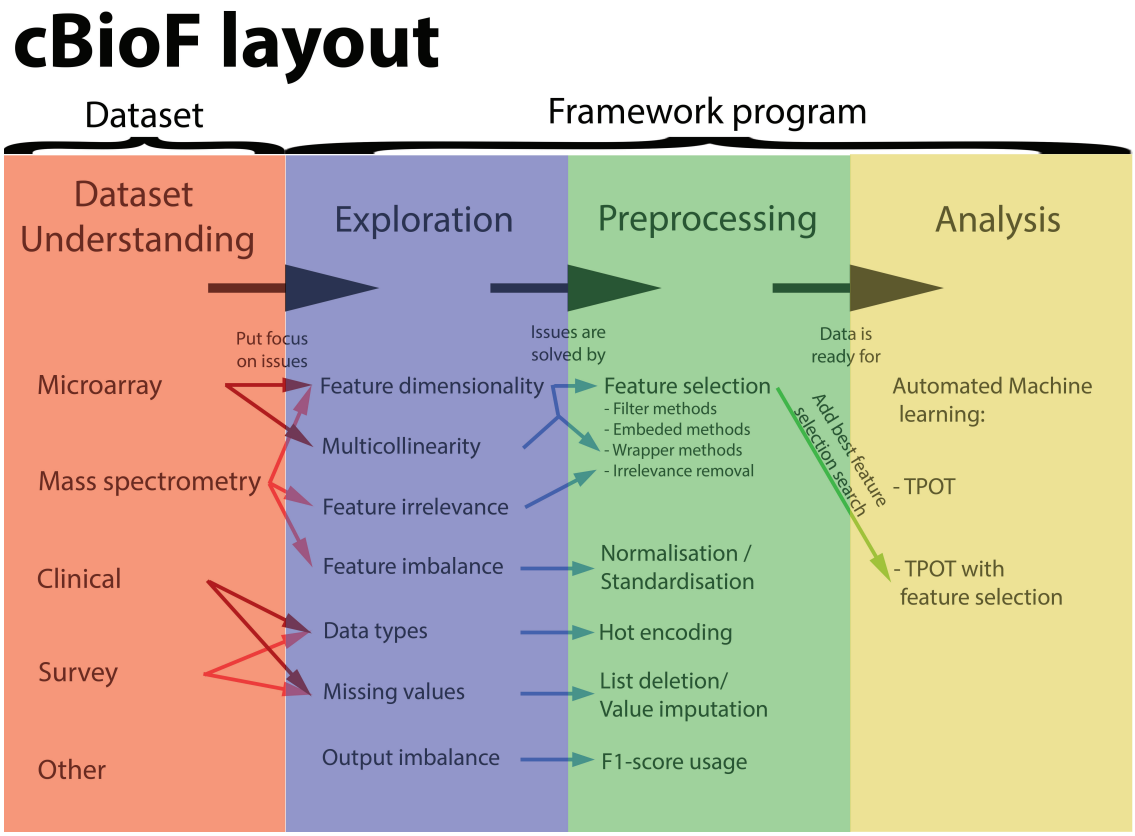


Figure 7.1: A graphical layout of the framework showing all aspects presented.

a few features are selected. Furthermore, genes are most likely very related to each other as well, showing high values for multicollinearity.

B *Mass spectrometry dataset*
Mass spectrometry usually consists of a high number proteins being investigated, resulting in a high number of features. Feature selection therefore is a useful way of removing irrelevant proteins. Mass spectrometry often does not detect any presence of a protein, when they are tested for. Therefore initially several proteins can be removed for not containing any information. This should be explored at the start. At last, peaks in the mass spectrometry data are very high in variance and the dataset would benefit from normalisation to remove this difference.

C *Clinical dataset*
There are very different clinical datasets, therefore potential issues are harder to pin down. Usually clinical datasets are not created by the data analyst however, therefore possibly creating mismatches on that part. The data types within the dataset can be different, having categorical, ordinal and numerical data. Hot encoding categorical data helps future analysis in this case. Also missing values are sometimes present, so missing value handling might be useful.

D *Survey dataset*
Survey dataset usually do not lack a number of instances, but do have other problems. Missing values are often present because of people not properly filling out all answers and therefore missing value handling is something to keep in mind. Survey datasets mostly contain categorical and ordinal data, but can also have numerical data. Hot encoding the categorical and ordinal data will make analyses with numeric methods possible.

7.3 Dataset Exploration

Exploration of the data is the next step in the framework to perform. This exploration will show whether several specific issues are present (Chapter 6). This results in the following framework question:

FQ2: What possible dataset issues can be present?

This is done by computing meta-features of the dataset using *metalearn*. These meta-features lead to seven different dataset issues. These issues are added to the framework layout (Figure 7.1). Links to preprocessing methods to solve them are added as well.

A *Feature dimensionality*
The dimensionality is checked as well as the mutual information present between data and output. If the dimensionality is high (more features than labels), features with a very low mutual information with the output can undesirably overshadow more useful features.

B *Multicollinearity*
Multicollinearity is present if the features contain common information. This is found by doing principal component analysis and correlation checking. If the first three principal components contain a significant amount of explained variance or if features have a high correlation, multicollinearity most likely is present.

C *Feature irrelevance*
To find out if there are features that do not contain any information, the cardinality for all features is checked. If features are present with a cardinality of one, irrelevant features are present. These features are not useful in any way and should better be removed.

D *Feature imbalance*
If the mean and standard deviation are significantly different in numeric features, bias can be present when analysing the dataset. How to cope with this is different for a normal distribution compared with not normal distributions. Therefore skewness and kurtosis can be used to find out if the features follow a normal distribution or not.

E *Data types*
The type of features can be different within the dataset. Since all analyses done in this study use numeric data, categorical data are not useful to have in the dataset.

F *Missing values*
Missing values can be present in the dataset. Since the presentation of missing values can be very different, it also might be hard to find these missing values.

G *Output imbalance*
By looking at the output classes, possible imbalance can be found. This imbalance shows possible bias when creating prediction results regarding false positive and negatives.

The exploration and preprocessing of these seven issues give the user a lot of insight in the data, as well as the possibility to quickly overcome them. Therefore the user can quickly start trying out analysis techniques to get the desired results.

7.4 Preprocessing

After exploration of the dataset, the issues that are found must be removed. The following framework question is about dataset preprocessing:

FQ3: What techniques can be used to overcome dataset issues?

Five possible solutions are given to overcome these issues. In the framework, these solutions can also be done automatically and in those cases generic solutions are used. The solutions are also shown, together with their influence in analysis, if present (Figure 7.1).

A *Feature selection*
Feature selection is a logical step to filter out unuseful features and preserve useful ones when a high dimensionality is present. For preprocessing this is fixed by doing a filter method or an embedded method. If multicollinearity is also present, a wrapper method can better be used, as it also takes into account multicollinearity (Chapter 4). At last, if irrelevant features are present, they can be simply removed by removing all features with a cardinality of 1. In automated preprocessing a simple filter method with ANOVA is used for feature selection without multicollinearity, otherwise a wrapper method with forward selection is used after ordering the data from most to least useful.

B *Normalisation/Standardisation*
To remove the feature imbalance, the data can be normalised or standardised. It is standardised when not following a normal distribution and normalised when it is following a normal distribution. When it is a normal distribution, the mean and standard deviation are investigated separately to find out if the data needs to be normalised for both.

C *Hot encoding*
To change categorical features into numeric ones, all categorical data are hot encoded into numeric data.

D *List deletion/Value imputation*
To handle missing values, several list deletion algorithms and value imputation algorithms are available. If only a relatively low number of instances or features have missing values, CCA

or ACA is the best methods to use. Otherwise a value imputation algorithm is preferred (Chapter 5). CCA or ACA are used when the number of missing values is at most 10% for either instances or features respectively, when preprocessing automatically. Otherwise a mean imputation method is used. In that mean imputation however, features are removed if they have more than 15% missing values, due to not properly being represented.

E *Robust score usage* When this imbalance is present, the user is advised to use an alternative score system than the standard accuracy, for example the F1-score.

7.5 Automated Analysis

After exploring and preprocessing the dataset, the next step would be to find a good analysis technique to find the desired results. Machine learning is a good approach for that, as it tries to predict the behaviour of a phenomenon by using the available data. Which technique should be used to predict this behaviour however is different for every dataset and result and therefore multiple analyses might be needed to find the correct one. Therefore the last framework question is implied:

FQ4: What technique can be best used to analyse the dataset?

One way to find the correct technique is using automated machine learning (Chapter 4, Figure 7.1). Automated machine learning tries to automatically find the best machine learning technique to predict the behaviour. On top of that, it also combines techniques and also tests out additional preprocessing methods for possible better results. The package *TPOT* is a tool that implements automated machine learning and can be used.

There is no best feature selection algorithm. Because of that a *TPOT* extension is made to select the best feature selection algorithm for the dataset. This extension multiplies the output with a penalty on the number of features. Also two additional features are added for further enhancing the search for the best feature selection algorithm. First the possibility of a feature selection favoured start is given, so the automated machine learning starts with a bias in feature selection. Secondly a new set of feature selection algorithms is given that focuses on preserving at most 200 features (Chapter 4). This way the user can select which combination of feature selection rules are needed for his dataset.

Chapter 8

Conclusions

This work discusses three research topics, each of them having their own conclusions. In this chapter these conclusions are combined and recommendations for future research are given. These conclusions are used in the corresponding parts of the framework (Chapter 7), which is the main contribution of this thesis. In what follows the main contributions and conclusions are summarized for every research line.

- *Feature Selection* (Chapter 4)
  Experiments were conducted involving feature selection methods. This resulted in four conclusions.
  - No significant difference in quality was found between using mutual information or T-test/ANOVA when using filter methods. Mutual information took longer to compute, but T-test/ANOVA assumed normal distributions, so both should be considered. Also for filter methods, no more than 200 features were needed to explain the mass spectrometry and microarray datasets. Additional features added little to the prediction quality and were therefore not needed.
  - Wrapper methods unexpectedly outperformed filter methods and embedded methods on everything but computation time. This difference in performance was due to wrapper methods removing multicollinearity. Filter and embedded methods performed very similar and seem to be the best choice when a lower computation time is desired and no dependencies are expected to be present.
  - There are differences between wrapper methods. Backwards and stochastic methods took much more computation time than forward methods and therefore are not worth using. This difference in computation time was also found comparing floating methods and forward and PTA methods. The difference in computation time was much lower, though, and floating methods did result in a higher output quality. The α threshold value, indicating the sensitivity of adding and removing features to the optimal feature set, was inversely correlated with the quality. A higher α indicated a lower quality, but preserved fewer features.
  - The Tree-based Pipeline Optimisation Tool ($TPOT$) showed a significantly better performance when using the improvements and for both very high number of features (50.000) and lower number of features (1.000-10.000) a better performance was found.
- *Missing Value Handling* (Chapter 5)
  Experiments involving missing value handling methods were done as well. These experiments resulted in three conclusions.
  - The feature distributions were not preserved enough when using list deletion methods. Imputation methods preserved feature distributions better, especially k-nearest neighbour imputation.

  - A conclusion was made on how to handle features with high percentages of missing values, indicating two thresholds for ratio of missing values. Features with at least 40% missing values should be removed, because of the bias created. Features with at least 15% missing values are highly recommended to be removed.
  - Conclusions were made based on the quality of the result. List deletion methods performed worse than imputation methods, as expected. Interestingly mean imputation outperformed more complex imputations, having similar quality but much lower computation time.

- *Dataset exploration* (Chapter 6)
  Two conclusions were made based on the analyses on dataset exploration.
  - Almost all discussed dataset issues could be found using the meta-features provided by the package *metalearn*. Multicollinearity was the only issue that was not clearly investigated.
  - Additional meta-features as well as the addition of outliers and plots gave the user a better insight in his dataset, as well as a better investigation in multicollinearity. This extra insight made the user have a better focus on what the possibilities are present for his dataset.

8.1 Future Work

In feature selection the first experiment shows that when using filter methods only 200 features must be preserved. This absolute number is not intuitive and much lower than expected. More research can be done on these preserved 200 features and why no other features are needed to predict the output. Cluster analyses for example may show insights in this phenomenon.

Also in feature selection only separate feature selection methods were tested. Whereas the filter methods needed very little computation time, the feature subset size and quality of the wrapper methods are much more interesting for further research. A combination of a filter and a wrapper method might create a better result. If first the majority of the features is filtered out by a filter method and then a wrapper method computes the best combination of feature, the result could be relatively quicker than the wrapper method while still maintaining a high enough quality. This combination is worth investigating in future research.

For missing value handling the datasets played a huge part in evaluation quality. Intuitively, precise imputation with the advanced missing value handling algorithms should outperform the less precise imputation of the simple algorithms. If the more precise imputation is done for an irrelevant feature value, though, the extra information will not produce a better output. Future research can be done on datasets with missing values that are known to be very important for the result to find out if this assumption holds for these datasets.

Dataset issues mentioned in this thesis are addressed in the framework and preprocessing methods or recommendations are given. A preprocessing method for multicollinearity however is only given in combination with feature selection, no preprocessing method only focusing on multicollinearity is present. Future research is needed to find a technique to cope with multicollinearity when feature selection is not needed.

There are more possible dataset issues than the aforementioned ones. Examples are anomalies in the data or having too few samples to analyse the data. These issues are not explored and the framework cannot properly remove these issues by preprocessing. Therefore it would benefit from future research such that these unexplored issues and possibilities to overcome them can be added.

Bibliography

[1] N. Gehlenborg, S. I. O'donoghue, N. S. Baliga, A. Goesmann, M. A. Hibbs, H. Kitano, O. Kohlbacher, H. Neuweger, R. Schneider, D. Tenenbaum, *et al.*, "Visualization of omics data for systems biology," *Nature methods*, vol. 7, no. 3s, p. S56, 2010.

[2] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, *et al.*, "Minimum information about a microarray experiment (miame)—toward standards for microarray data," *Nature genetics*, vol. 29, no. 4, p. 365, 2001.

[3] J. S. Cottrell and U. London, "Probability-based protein identification by searching sequence databases using mass spectrometry data," *electrophoresis*, vol. 20, no. 18, pp. 3551–3567, 1999.

[4] K. Dettmer, P. A. Aronov, and B. D. Hammock, "Mass spectrometry-based metabolomics," *Mass spectrometry reviews*, vol. 26, no. 1, pp. 51–78, 2007.

[5] B. Liu, X. Zhou, Y. Wang, J. Hu, L. He, R. Zhang, S. Chen, and Y. Guo, "Data processing and analysis in real-world traditional chinese medicine clinical data: challenges and approaches," *Statistics in medicine*, vol. 31, no. 7, pp. 653–660, 2012.

[6] D. F. Sittig, A. Wright, J. A. Osheroff, B. Middleton, J. M. Teich, J. S. Ash, E. Campbell, and D. W. Bates, "Grand challenges in clinical decision support," *Journal of biomedical informatics*, vol. 41, no. 2, pp. 387–392, 2008.

[7] G. Magni, C. Caldieron, S. Rigatti-Luchini, and H. Merskey, "Chronic musculoskeletal pain and depressive symptoms in the general population. an analysis of the 1st national health and nutrition examination survey data," *Pain*, vol. 43, no. 3, pp. 299–307, 1990.

[8] P. Bertolazzi, G. Felici, P. Festa, and G. Lancia, "Logic classification and feature selection for biomedical data," *Computers & Mathematics with Applications*, vol. 55, no. 5, pp. 889–899, 2008.

[9] G. Piatetsky-Shapiro and P. Tamayo, "Microarray data mining: facing the challenges," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 2, pp. 1–5, 2003.

[10] A. Lommen, "Metalign: interface-driven, versatile metabolomics tool for hyphenated full-scan mass spectrometry data preprocessing," *Analytical chemistry*, vol. 81, no. 8, pp. 3079–3086, 2009.

[11] A. Holzinger, M. Dehmer, and I. Jurisica, "Knowledge discovery and interactive data mining in bioinformatics-state-of-the-art, future challenges and research directions," *BMC bioinformatics*, vol. 15, no. 6, p. I1, 2014.

[12] M. Wilkins, "Proteomics data mining," *Expert review of proteomics*, vol. 6, no. 6, pp. 599–603, 2009.

[13] D. Teodoro, R. Choquet, E. Pasche, J. Gobeill, C. Daniel, P. Ruch, and C. Lovis, "Biomedical data management: a proposal framework.," in *MIE*, pp. 175–179, Citeseer, 2009.

[14] M. Y. Galperin, "The molecular biology database collection: 2008 update," *Nucleic Acids Research*, vol. 36, no. suppl1, pp. D2–D4, 2008.

[15] A. Sturn, J. Quackenbush, and Z. Trajanoski, "Genesis: cluster analysis of microarray data," *Bioinformatics*, vol. 18, no. 1, pp. 207–208, 2002.

[16] A. Karnovsky, T. Weymouth, T. Hull, V. G. Tarcea, G. Scardoni, C. Laudanna, M. A. Sartor, K. A. Stringer, H. Jagadish, C. Burant, *et al.*, "Metscape 2 bioinformatics tool for the analysis and visualization of metabolomics and gene expression data," *Bioinformatics*, vol. 28, no. 3, pp. 373–380, 2011.

[17] D. Tabas-Madrid, R. Nogales-Cadenas, and A. Pascual-Montano, "Genecodis3: a non-redundant and modular enrichment analysis tool for functional genomics," *Nucleic acids research*, vol. 40, no. W1, pp. W478–W483, 2012.

[18] F. Faul, E. Erdfelder, A.-G. Lang, and A. Buchner, "G* power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences," *Behavior research methods*, vol. 39, no. 2, pp. 175–191, 2007.

[19] R. Baumgartner and R. L. Somorjai, "Data complexity assessment in undersampled classification of high-dimensional biomedical data," *Pattern Recognition Letters*, vol. 27, no. 12, pp. 1383–1389, 2006.

[20] W. Welthagen, R. A. Shellie, J. Spranger, M. Ristow, R. Zimmermann, and O. Fiehn, "Comprehensive two-dimensional gas chromatography–time-of-flight mass spectrometry (gc× gc-tof) for high resolution metabolomics: biomarker discovery on spleen tissue extracts of obese nzo compared to lean c57bl/6 mice," *Metabolomics*, vol. 1, no. 1, pp. 65–73, 2005.

[21] A. R. T. Donders, G. J. Van Der Heijden, T. Stijnen, and K. G. Moons, "A gentle introduction to imputation of missing values," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1087–1091, 2006.

[22] M. H. Cartwright, M. J. Shepperd, and Q. Song, "Dealing with missing software project data," in *Software Metrics Symposium, 2003. Proceedings. Ninth International*, pp. 154–165, IEEE, 2003.

[23] J. S. Haukoos and C. D. Newgard, "Advanced statistics: missing data in clinical research—part 1: an introduction and conceptual framework," *Academic Emergency Medicine*, vol. 14, no. 7, pp. 662–668, 2007.

[24] R. Deneer, "Scoring co-morbidity severity in bariatric patients based on biomarkers: a data mining approach," 2017.

[25] R. P. Nair, K. C. Duffin, C. Helms, J. Ding, P. E. Stuart, D. Goldgar, J. E. Gudjonsson, Y. Li, T. Tejasvi, B.-J. Feng, *et al.*, "Genome-wide scan reveals association of psoriasis with il-23 and nf-κb pathways," *Nature genetics*, vol. 41, no. 2, pp. 199–204, 2009.

[26] M. Suárez-Farinas, K. Li, J. Fuentes-Duculan, K. Hayden, C. Brodmerkel, and J. G. Krueger, "Expanding the psoriasis disease profile: interrogation of the skin and serum of patients with moderate-to-severe psoriasis," *Journal of Investigative Dermatology*, vol. 132, no. 11, pp. 2552–2564, 2012.

[27] J. Bigler, H. A. Rand, K. Kerkof, M. Timour, and C. B. Russell, "Cross-study homogeneity of psoriasis gene expression in skin across a large expression range," *PLoS One*, vol. 8, no. 1, p. e52242, 2013.

[28] J. Kim, R. Bissonnette, J. Lee, J. C. da Rosa, M. Suárez-Fariñas, M. A. Lowes, and J. G. Krueger, "The spectrum of mild to severe psoriasis vulgaris is defined by a common activation of il-17 pathway genes, but with key differences in immune regulatory genes," *Journal of Investigative Dermatology*, vol. 136, no. 11, pp. 2173–2182, 2016.

[29] Y. Yao, L. Richman, C. Morehouse, M. De Los Reyes, B. W. Higgs, A. Boutrin, B. White, A. Coyle, J. Krueger, P. A. Kiener, *et al.*, "Type i interferon: potential therapeutic target for psoriasis?," *PloS one*, vol. 3, no. 7, p. e2737, 2008.

[30] M. Suárez-Fariñas, S. J. Tintle, A. Shemer, A. Chiricozzi, K. Nograles, I. Cardinale, S. Duan, A. M. Bowcock, J. G. Krueger, and E. Guttman-Yassky, "Nonlesional atopic dermatitis skin is characterized by broad terminal differentiation defects and variable immune abnormalities," *Journal of Allergy and Clinical Immunology*, vol. 127, no. 4, pp. 954–964, 2011.

[31] S. Tintle, A. Shemer, M. Suárez-Fariñas, H. Fujita, P. Gilleaudeau, M. Sullivan-Whalen, L. Johnson-Huang, A. Chiricozzi, I. Cardinale, S. Duan, *et al.*, "Reversal of atopic dermatitis with narrow-band uvb phototherapy and biomarkers for therapeutic response," *Journal of Allergy and Clinical Immunology*, vol. 128, no. 3, pp. 583–593, 2011.

[32] J. K. Gittler, A. Shemer, M. Suárez-Fariñas, J. Fuentes-Duculan, K. J. Gulewicz, C. Q. Wang, H. Mitsui, I. Cardinale, C. de Guzman Strong, J. G. Krueger, *et al.*, "Progressive activation of t h 2/t h 22 cytokines and selective epidermal proteins characterizes acute and chronic atopic dermatitis," *Journal of Allergy and Clinical Immunology*, vol. 130, no. 6, pp. 1344–1354, 2012.

[33] K. J. Cios and G. W. Moore, "Uniqueness of medical data mining," *Artificial Intelligence in Medicine*, vol. 26, no. 1, pp. 1 – 24, 2002. Medical Data Mining and Knowledge Discovery.

[34] I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang, and L. Hua, "Data mining in healthcare and biomedicine: A survey of the literature," *Journal of Medical Systems*, vol. 36, pp. 2431–2448, Aug 2012.

[35] H. Chen, S. S. Fuller, C. Friedman, and W. Hersh, *Medical informatics: knowledge management and data mining in biomedicine*, vol. 8. Springer Science & Business Media, 2006.

[36] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in Bioinformatics*, p. bbx044, 2017.

[37] D. Blythe, "Rise of the graphics processor," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 761–778, 2008.

[38] C. Turkay, F. Jeanquartier, A. Holzinger, and H. Hauser, *On Computationally-Enhanced Visual Analysis of Heterogeneous Data and Its Application in Biomedical Informatics*, pp. 117–140. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[39] A. Holzinger and I. Jurisica, *Knowledge Discovery and Data Mining in Biomedical Informatics: The Future Is in Integrative, Interactive Machine Learning Solutions*, pp. 1–18. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[40] W. Dubitzky, M. Granzow, and D. P. Berrar, *Fundamentals of data mining in genomics and proteomics*. Springer Science & Business Media, 2007.

[41] Y. Peng, Z. Wu, and J. Jiang, "A novel feature selection approach for biomedical data classification," *Journal of Biomedical Informatics*, vol. 43, no. 1, pp. 15 – 23, 2010.

[42] D. Dunbar and G. Humphreys, "A spatial data structure for fast poisson-disk sample generation," in *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 503–508, ACM, 2006.

[43] L. Devroye, "Sample-based non-uniform random variate generation," in *Proceedings of the 18th conference on Winter simulation*, pp. 260–265, ACM, 1986.

[44] L. J. Van't Veer, H. Dai, M. J. Van De Vijver, Y. D. He, A. A. Hart, M. Mao, H. L. Peterse, K. Van Der Kooy, M. J. Marton, A. T. Witteveen, *et al.*, "Gene expression profiling predicts clinical outcome of breast cancer," *nature*, vol. 415, no. 6871, p. 530, 2002.

[45] D. A. Roff and P. Bentzen, "The statistical analysis of mitochondrial dna polymorphisms: chi 2 and the problem of small samples.," *Molecular biology and evolution*, vol. 6, no. 5, pp. 539–545, 1989.

[46] R. Bellazzi, M. Diomidous, I. N. Sarkar, K. Takabayashi, A. Ziegler, A. T. McCray, *et al.*, "Data analysis and data mining: current issues in biomedical informatics," *Methods of information in medicine*, vol. 50, no. 6, p. 536, 2011.

[47] D. Otasek, C. Pastrello, A. Holzinger, and I. Jurisica, *Visual Data Mining: Effective Exploration of the Biological Universe*, pp. 19–33. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[48] L. Marenco, T.-Y. Wang, G. Shepherd, P. L. Miller, and P. Nadkarni, "Qis: A framework for biomedical database federation," *Journal of the American Medical Informatics Association*, vol. 11, no. 6, pp. 523–534, 2004.

[49] V. Y. Bichutskiy, R. Colman, R. K. Brachmann, and R. H. Lathrop, "Heterogeneous biomedical database integration using a hybrid strategy: a p53 cantcer research database," *Cancer informatics*, vol. 2, p. 277, 2006.

[50] W. Sperzel, R. Abarbanel, S. Nelson, M. Erlbaum, D. Sherertz, M. Tuttle, N. Olson, and L. Fuller, "Biomedical database inter-connectivity: an experiment linking mim, genbank, and meta-1 via medline.," in *Proceedings of the Annual Symposium on Computer Application in Medical Care*, p. 190, American Medical Informatics Association, 1991.

[51] F. Aubry, S. Badaoui, H. Kaplan, and R. D. Paola, "Design and implementation of a biomedical image database (bdim)," *Medical Informatics*, vol. 13, no. 4, pp. 241–248, 1988.

[52] D. Windridge and M. Bober, *A Kernel-Based Framework for Medical Big-Data Analytics*, pp. 197–208. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.

[53] S. Selvaraj and J. Natarajan, "Microarray data analysis and mining tools," *Bioinformation*, vol. 6, no. 3, p. 95, 2011.

[54] M. Afzal, I. Manzoor, and O. P. Kuipers, "A fast and reliable pipeline for bacterial transcriptome analysis case study: serine-dependent gene regulation in streptococcus pneumoniae," *Journal of visualized experiments: JoVE*, no. 98, 2015.

[55] M. Wojnarski, A. Janusz, H. S. Nguyen, J. Bazan, C. Luo, Z. Chen, F. Hu, G. Wang, L. Guan, H. Luo, *et al.*, "Rsctc'2010 discovery challenge: Mining dna microarray data for medical diagnosis and treatment," in *International Conference on Rough Sets and Current Trends in Computing*, pp. 4–19, Springer, 2010.

[56] R. Matthiesen and O. N. Jensen, "Analysis of mass spectrometry data in proteomics," in *Bioinformatics*, pp. 105–122, Springer, 2008.

[57] J. T. Watson and O. D. Sparkman, *Introduction to mass spectrometry: instrumentation, applications, and strategies for data interpretation.* John Wiley & Sons, 2007.

[58] A. C. Neves, C. L. Morais, T. P. Mendes, B. G. Vaz, and K. M. Lima, "Mass spectrometry and multivariate analysis to classify cervical intraepithelial neoplasia from blood plasma: an untargeted lipidomic study," *Scientific reports*, vol. 8, no. 1, p. 3954, 2018.

[59] I. Guyon, S. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the nips 2003 feature selection challenge," in *Advances in Neural Information Processing Systems 17* (L. K. Saul, Y. Weiss, and L. Bottou, eds.), pp. 545–552, MIT Press, 2005.

[60] P. Mahé, M. Arsac, S. Chatellier, V. Monnin, N. Perrot, S. Mailler, V. Girard, M. Ramjeet, J. Surre, B. Lacroix, A. van Belkum, and J.-B. Veyrieras, "Automatic identification of mixed bacterial species fingerprints in a maldi-tof mass-spectrum," *Bioinformatics*, vol. 30, no. 9, pp. 1280–1286, 2014.

[61] M. T. Madigan, J. M. Martinko, J. Parker, *et al.*, *Brock biology of microorganisms*, vol. 13. Pearson, 2017.

[62] S. J. Pocock, *Clinical trials: a practical approach*. John Wiley & Sons, 2013.

[63] A. Bendele, T. McAbee, G. Sennello, J. Frazier, E. Chlipala, and D. McCabe, "Efficacy of sustained blood levels of interleukin-1 receptor antagonist in animal models of arthritis: comparison of efficacy in animal models with human clinical data," *Arthritis & Rheumatism: Official Journal of the American College of Rheumatology*, vol. 42, no. 3, pp. 498–506, 1999.

[64] G. Kan, C. A. Visser, J. J. Koolen, and A. J. Dunning, "Short and long term predictive value of admission wall motion score in acute myocardial infarction. a cross sectional echocardiographic study of 345 patients.," *Heart*, vol. 56, no. 5, pp. 422–427, 1986.

[65] P. Diaconis and B. Efron, "Computer-intensive methods in statistics," *Scientific American*, vol. 248, no. 5, pp. 116–131, 1983.

[66] B. Cestnik, "Kononenkoj., bratkoj.(1987): Assistant-86: A knowledge elicitation tool for sophisticated users," *Progress in machine learning*.

[67] P. A. Murtaugh, E. R. Dickson, G. M. Van Dam, M. Malinchoc, P. M. Grambsch, A. L. Langworthy, and C. H. Gips, "Primary biliary cirrhosis: prediction of short-term survival based on repeated patient visits," *Hepatology*, vol. 20, no. 1, pp. 126–134, 1994.

[68] L. Misery, V. Sibaud, C. Merial-Kieny, and C. Taieb, "Sensitive skin in the american population: prevalence, clinical data, and role of the dermatologist," *International journal of dermatology*, vol. 50, no. 8, pp. 961–967, 2011.

[69] K. Fernandes, J. S. Cardoso, and J. Fernandes, "Transfer learning with partial observability applied to cervical cancer screening," in *Iberian conference on pattern recognition and image analysis*, pp. 243–250, Springer, 2017.

[70] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, SciPy Austin, TX, 2010.

[71] Y. Yan and J. Yan, "Hands-on data science with anaconda: Utilize the right mix of tools to create high-performance data science applications," 2018.

[72] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[73] E. Jones, T. Oliphant, and P. Peterson, "{SciPy}: open source scientific tools for {Python}," 2014.

[74] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[75] W. McKinney, *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. " O'Reilly Media, Inc.", 2012.

[76] R. S. Olson and J. H. Moore, "Tpot: A tree-based pipeline optimization tool for automating machine learning," in *Workshop on Automatic Machine Learning*, pp. 66–74, 2016.

[77] L. Yu and H. Liu, "Feature selection for high-dimensional data: A fast correlation-based filter solution," in *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 856–863, 2003.

[78] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," 2000.

[79] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, pp. 1137–1145, Montreal, Canada, 1995.

[80] M. Kearns and D. Ron, "Algorithmic stability and sanity-check bounds for leave-one-out cross-validation," *Neural computation*, vol. 11, no. 6, pp. 1427–1453, 1999.

[81] A. Agresti, *Categorical data analysis*, vol. 482. John Wiley & Sons, 2003.

[82] A. Edwards, G. Elwyn, and A. Mulley, "Explaining risks: turning numerical data into meaningful pictures," *Bmj*, vol. 324, no. 7341, pp. 827–830, 2002.

[83] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[84] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[85] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE Transactions on neural networks*, vol. 5, no. 4, pp. 537–550, 1994.

[86] N. J.-M. Blackman and J. J. Koval, "Interval estimation for cohen's kappa as a measure of agreement," *Statistics in medicine*, vol. 19, no. 5, pp. 723–741, 2000.

[87] I. S. Lim, P. de Heras Ciechomski, S. Sarni, and D. Thalmann, "Planar arrangement of high-dimensional biomedical data sets by isomap coordinates," in *Computer-Based Medical Systems, 2003. Proceedings. 16th IEEE Symposium*, pp. 50–55, IEEE, 2003.

[88] Y. Peng, Z. Wu, and J. Jiang, "A novel feature selection approach for biomedical data classification," *Journal of Biomedical Informatics*, vol. 43, no. 1, pp. 15–23, 2010.

[89] J. Biesiada and W. Duch, "Feature selection for high-dimensional data—a pearson redundancy based filter," in *Computer recognition systems 2*, pp. 242–249, Springer, 2007.

[90] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005.

[91] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, vol. 179, no. 8, pp. 1040–1058, 2009.

[92] H. Liu, J. Li, and L. Wong, "A comparative study on feature selection and classification methods using gene expression profiles and proteomic patterns," *Genome informatics*, vol. 13, pp. 51–60, 2002.

[93] I. Guyon and A. Elisseeff, *An Introduction to Feature Extraction*, pp. 1–25. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[94] C. Catal and B. Diri, "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem," *Information Sciences*, vol. 179, no. 8, pp. 1040 – 1058, 2009.

[95] L. C. Molina, L. Belanche, and À. Nebot, "Feature selection algorithms: A survey and experimental evaluation," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pp. 306–313, IEEE, 2002.

[96] Y. Saeys, I. Inza, and P. Larrañaga, "A review of feature selection techniques in bioinformatics," *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.

[97] W. Duch, *Filter Methods*, pp. 89–117. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[98] R. M. Heiberger and B. Holland, *Statistical analysis and data display*. Springer, 2004.

[99] D. Donoho and J. Jin, "Higher criticism thresholding: Optimal feature selection when useful features are rare and weak," *Proceedings of the National Academy of Sciences*, vol. 105, no. 39, pp. 14790–14795, 2008.

[100] J. D. Storey and R. Tibshirani, "Statistical significance for genomewide studies," *Proceedings of the National Academy of Sciences*, vol. 100, no. 16, pp. 9440–9445, 2003.

[101] J. P. Higgins, S. G. Thompson, J. J. Deeks, and D. G. Altman, "Measuring inconsistency in meta-analyses," *BMJ: British Medical Journal*, vol. 327, no. 7414, p. 557, 2003.

[102] J. Reunanen, *Search Strategies*, pp. 119–136. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[103] B. Alsallakh and L. Ren, "Powerset: A comprehensive visualization of set intersections," vol. 23, pp. 1–1, 01 2016.

[104] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, and V. Christophides, "Massively-parallel feature selection for big data," *arXiv preprint arXiv:1708.07178*, 2017.

[105] Y.-J. Huang, D.-Y. Chan, D.-C. Cheng, Y.-J. Ho, P.-P. Tsai, W.-C. Shen, and R.-F. Chen, "Automated feature set selection and its application to mcc identification in digital mammograms for breast cancer detection," *Sensors*, vol. 13, no. 4, pp. 4855–4875, 2013.

[106] A. Senawi, H.-L. Wei, and S. A. Billings, "A new maximum relevance-minimum multicollinearity (mrmmc) method for feature selection and ranking," *Pattern Recognition*, vol. 67, pp. 47 – 61, 2017.

[107] A. El Akadi, A. Amine, A. El Ouardighi, and D. Aboutajdine, "A new gene selection approach based on minimum redundancy-maximum relevance (mrmr) and genetic algorithm (ga)," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, pp. 69–75, IEEE, 2009.

[108] M. Radovic, M. Ghalwash, N. Filipovic, and Z. Obradovic, "Minimum redundancy maximum relevance feature selection approach for temporal gene expression data," *BMC bioinformatics*, vol. 18, no. 1, p. 9, 2017.

[109] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[110] T. Jirapech-Umpai and S. Aitken, "Feature selection and classification for microarray data analysis: Evolutionary methods for identifying predictive genes," *BMC Bioinformatics*, vol. 6, p. 148, Jun 2005.

[111] T. N. Lal, O. Chapelle, J. Weston, and A. Elisseeff, *Embedded Methods*, pp. 137–165. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[112] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.

[113] K. Jong, E. Marchiori, M. Sebag, and A. Van Der Vaart, "Feature selection in proteomic pattern data with support vector machines," in *Computational Intelligence in Bioinformatics and Computational Biology, 2004. CIBCB'04. Proceedings of the 2004 IEEE Symposium on*, pp. 41–48, IEEE, 2004.

[114] J. Prados, A. Kalousis, J.-C. Sanchez, L. Allard, O. Carrette, and M. Hilario, "Mining mass spectra for diagnosis and biomarker discovery of cerebral accidents," *Proteomics*, vol. 4, no. 8, pp. 2320–2332, 2004.

[115] X. Zhang, X. Lu, Q. Shi, X.-q. Xu, E. L. Hon-chiu, L. N. Harris, J. D. Iglehart, A. Miron, J. S. Liu, and W. H. Wong, "Recursive svm feature selection and sample classification for mass-spectrometry and microarray data," *BMC bioinformatics*, vol. 7, no. 1, p. 197, 2006.

[116] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[117] P. Geurts, M. Fillet, D. De Seny, M.-A. Meuwis, M. Malaise, M.-P. Merville, and L. Wehenkel, "Proteomic mass spectra classification using decision tree based ensemble methods," *Bioinformatics*, vol. 21, no. 14, pp. 3138–3145, 2005.

[118] B. Wu, T. Abbott, D. Fishman, W. McMurray, G. Mor, K. Stone, D. Ward, K. Williams, and H. Zhao, "Comparison of statistical methods for classification of ovarian cancer using mass spectrometry data," *Bioinformatics*, vol. 19, no. 13, pp. 1636–1643, 2003.

[119] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[120] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "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, ACM, 2013.

[121] P. Gijsbers, "Automatic construction of machine learning pipelines," 2017.

[122] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka," *Journal of Machine Learning Research*, vol. 17, pp. 1–5, 2016.

[123] J. R. Koza, "Genetic programming," 1997.

[124] P. Brazdil, J. Gama, and B. Henery, "Characterizing the applicability of classification algorithms using meta-level learning," in *European conference on machine learning*, pp. 83–102, Springer, 1994.

[125] R. Vilalta, C. G. Giraud-Carrier, P. Brazdil, and C. Soares, "Using meta-learning to support data mining.," *IJCSA*, vol. 1, no. 1, pp. 31–45, 2004.

[126] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, "Development of metalearning systems for algorithm recommendation," *Metalearning: Applications to Data Mining*, pp. 31–59, 2009.

[127] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites," *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997.

[128] Y.-W. Chen and C.-J. Lin, "Combining svms with various feature selection strategies," in *Feature extraction*, pp. 315–324, Springer, 2006.

[129] M. A. Hall and L. A. Smith, "Practical feature subset selection for machine learning," 1998.

[130] P. A. Patrician, "Multiple imputation for missing data," *Research in Nursing & Health*, vol. 25, no. 1, pp. 76–84, 2002.

[131] J. A. Sterne, I. R. White, J. B. Carlin, M. Spratt, P. Royston, M. G. Kenward, A. M. Wood, and J. R. Carpenter, "Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls," *Bmj*, vol. 338, p. b2393, 2009.

[132] I. Myrtveit, E. Stensrud, and U. H. Olsson, "Analyzing data sets with missing data: An empirical evaluation of imputation methods and likelihood-based methods," *IEEE Transactions on Software Engineering*, vol. 27, no. 11, pp. 999–1013, 2001.

[133] A. B. Pedersen, E. M. Mikkelsen, D. Cronin-Fenton, N. R. Kristensen, T. M. Pham, L. Pedersen, and I. Petersen, "Missing data and multiple imputation in clinical epidemiological research," *Clinical Epidemiology*, vol. 9, p. 157, 2017.

[134] T. E. Raghunathan, J. M. Lepkowski, J. Van Hoewyk, and P. Solenberger, "A multivariate technique for multiply imputing missing values using a sequence of regression models," *Survey methodology*, vol. 27, no. 1, pp. 85–96, 2001.

[135] S. Chevret, S. Seaman, and M. Resche-Rigon, "Multiple imputation: a mature approach to dealing with missing data," *Intensive care medicine*, vol. 41, no. 2, pp. 348–350, 2015.

[136] D. B. Rubin, "Inference and missing data," *Biometrika*, vol. 63, no. 3, pp. 581–592, 1976.

[137] I. R. White, P. Royston, and A. M. Wood, "Multiple imputation using chained equations: issues and guidance for practice," *Statistics in medicine*, vol. 30, no. 4, pp. 377–399, 2011.

[138] Y. He, A. M. Zaslavsky, M. Landrum, D. Harrington, and P. Catalano, "Multiple imputation in a large-scale complex survey: a practical guide," *Statistical methods in medical research*, vol. 19, no. 6, pp. 653–670, 2010.

[139] S. Van Buuren, "Multiple imputation of discrete and continuous data by fully conditional specification," *Statistical methods in medical research*, vol. 16, no. 3, pp. 219–242, 2007.

[140] S. Van Buuren, H. C. Boshuizen, D. L. Knook, *et al.*, "Multiple imputation of missing blood pressure covariates in survival analysis," *Statistics in medicine*, vol. 18, no. 6, pp. 681–694, 1999.

[141] G. J. Van der Heijden, A. R. T. Donders, T. Stijnen, and K. G. Moons, "Imputation of missing values is superior to complete case analysis and the missing-indicator method in multivariable diagnostic research: a clinical example," *Journal of clinical epidemiology*, vol. 59, no. 10, pp. 1102–1109, 2006.

[142] M. J. Azur, E. A. Stuart, C. Frangakis, and P. J. Leaf, "Multiple imputation by chained equations: what is it and how does it work?," *International journal of methods in psychiatric research*, vol. 20, no. 1, pp. 40–49, 2011.

[143] P. Royston *et al.*, "Multiple imputation of missing values," *Stata journal*, vol. 4, no. 3, pp. 227–41, 2004.

[144] E. Martín-Merino, A. Calderón-Larrañaga, S. Hawley, B. Poblador-Plou, A. Llorente-García, I. Petersen, and D. Prieto-Alhambra, "The impact of different strategies to handle missing data on both precision and bias in a drug safety study: a multidatabase multinational population-based cohort study," *Clinical epidemiology*, vol. 10, p. 643, 2018.

[145] N. J. Horton and S. R. Lipsitz, "Multiple imputation in practice: comparison of software packages for regression models with missing variables," *The American Statistician*, vol. 55, no. 3, pp. 244–254, 2001.

[146] P. D. Allison, "Multiple imputation for missing data: A cautionary tale," *Sociological methods & research*, vol. 28, no. 3, pp. 301–309, 2000.

[147] P. Royston, I. R. White, *et al.*, "Multiple imputation by chained equations (mice): implementation in stata," *J Stat Softw*, vol. 45, no. 4, pp. 1–20, 2011.

[148] M. Bartlett, "The effect of non-normality on the t distribution," in *mathematical proceedings of the cambridge philosophical society*, vol. 31, pp. 223–231, Cambridge University Press, 1935.

[149] M. B. Brown and A. B. Forsythe, "Robust tests for the equality of variances," *Journal of the American Statistical Association*, vol. 69, no. 346, pp. 364–367, 1974.

[150] A. Satorra and P. M. Bentler, "A scaled difference chi-square test statistic for moment structure analysis," *Psychometrika*, vol. 66, no. 4, pp. 507–514, 2001.

[151] N. R. Draper and H. Smith, *Applied regression analysis*, vol. 326. John Wiley & Sons, 2014.

[152] P. Kluegl, M. Atzmueller, and F. Puppe, "Meta-level information extraction," in *Annual Conference on Artificial Intelligence*, pp. 233–240, Springer, 2009.

[153] P. Congdon, *Bayesian models for categorical data*. John Wiley & Sons, 2005.

[154] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE transactions on systems, man, and cybernetics*, vol. 21, no. 3, pp. 660–674, 1991.

[155] A. Gelman and J. Hill, *Data analysis using regression and multilevel/hierarchical models*. Cambridge university press, 2006.

[156] C. Guo and F. Berkhahn, "Entity embeddings of categorical variables," *arXiv preprint arXiv:1604.06737*, 2016.

[157] D. T. Larose and C. D. Larose, *Discovering knowledge in data: an introduction to data mining*. John Wiley & Sons, 2014.

[158] L. Al Shalabi and Z. Shaaban, "Normalization as a preprocessing engine for data mining and the approach of preference matrix," in *Dependability of Computer Systems, 2006. DepCos-RELCOMEX'06. International Conference on*, pp. 207–214, IEEE, 2006.

[159] S. Patro and K. K. Sahu, "Normalization: A preprocessing stage," *arXiv preprint arXiv:1503.06462*, 2015.

[160] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, pp. 321–352, Springer, 2005.

[161] R. F. Woolson and W. R. Clarke, *Statistical methods for the analysis of biomedical data*, vol. 371. John Wiley & Sons, 2011.

[162] R. Sapsford and V. Jupp, *Data collection and analysis*. Sage, 2006.

[163] M. J. Fisher and A. P. Marshall, "Understanding descriptive statistics," *Australian Critical Care*, vol. 22, no. 2, pp. 93 – 97, 2009.

[164] W. D. Dupont and W. D. Dupont, *Statistical modeling for biomedical researchers: a simple introduction to the analysis of complex data*. Cambridge University Press, 2009.

[165] R. Shiavi, *Introduction to applied statistical signal analysis: Guide to biomedical and electrical engineering applications*. Academic Press, 2010.

[166] N. O'Rourke and L. Hatcher, *A step-by-step approach to using SAS for factor analysis and structural equation modeling*. Sas Institute, 2013.

[167] D. Muijs, *Doing quantitative research in education with SPSS*. Sage, 2010.

[168] R. A. Muenchen, *R for SAS and SPSS users*. Springer Science & Business Media, 2011.

[169] W. L. Martinez and A. R. Martinez, *Computational statistics handbook with MATLAB*, vol. 22. CRC press, 2007.

[170] J. Kromme, "Python & r vs. spss & sas," March 2017. https://www.r-bloggers.com/python-r-vs-spss-sas/.

[171] K. Jain, "Python vs. r (vs. sas) – which tool should i learn?," September 2017. https://www.analyticsvidhya.com/blog/2017/09/sas-vs-vs-python-tool-learn/.

[172] K. Willems, "What is the best statistical programming language? infograph," June 2014. https://www.datacamp.com/community/tutorials/statistical-language-wars-the-infograph.

[173] Support, "R vs sas vs spss – top 3 data analytics tools comparison," March 2017. http://data-flair.training/blogs/r-sas-spss-data-analytics-tools-comparison/.

[174] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[175] H. Greenspan, B. van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1153–1159, 2016.

[176] R. Rojas, *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[177] S.-C. Wang, *Artificial Neural Network*, pp. 81–100. Boston, MA: Springer US, 2003.

[178] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A cpu and gpu math compiler in python," in *Proc. 9th Python in Science Conf*, pp. 1–7, 2010.

[179] L. Rampasek and A. Goldenberg, "Tensorflow: Biology's gateway to deep learning?," *Cell systems*, vol. 2, no. 1, pp. 12–14, 2016.

[180] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint*, 2016.

[181] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[182] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *CoRR*, vol. abs/1408.5093, 2014.

[183] R. Collobert, S. Bengio, and J. Mariéthoz, "Torch: a modular machine learning software library," tech. rep., Idiap, 2002.

[184] V. Kovalev, A. Kalinovsky, and S. Kovalev, "Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy?," 2016.

[185] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of caffe, neon, theano, and torch for deep learning," 2016.

[186] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," *CoRR*, vol. abs/1608.07249, 2016.

[187] J. Fox, Y. Zou, and J. Qiu, "Software frameworks for deep learning at scale," *Internal Indiana University Technical Report*, 2016.

[188] A. Parvat, J. Chavan, S. Kadam, S. Dev, and V. Pathak, "A survey of deep-learning frameworks," in *Inventive Systems and Control (ICISC), 2017 International Conference on*, pp. 1–7, IEEE, 2017.

[189] B. J. Erickson, P. Korfiatis, Z. Akkus, T. Kline, and K. Philbrick, "Toolkits and libraries for deep learning," *Journal of Digital Imaging*, pp. 1–6, 2017.

[190] S. R. Rapp, S. R. Feldman, M. L. Exum, A. B. Fleischer, and D. M. Reboussin, "Psoriasis causes as much disability as other major medical diseases," *Journal of the American Academy of Dermatology*, vol. 41, no. 3, pp. 401–407, 1999.

[191] S. Jowett and T. Ryan, "Skin disease and handicap: an analysis of the impact of skin conditions," *Social science & medicine*, vol. 20, no. 4, pp. 425–429, 1985.

[192] R. Edgar, M. Domrachev, and A. E. Lash, "Gene expression omnibus: Ncbi gene expression and hybridization array data repository," *Nucleic acids research*, vol. 30, no. 1, pp. 207–210, 2002.

[193] A. L. Dixon, L. Liang, M. F. Moffatt, W. Chen, S. Heath, K. C. Wong, J. Taylor, E. Burnett, I. Gut, M. Farrall, *et al.*, "A genome-wide association study of global gene expression," *Nature genetics*, vol. 39, no. 10, p. 1202, 2007.

[194] E. D. Roberson and A. M. Bowcock, "Psoriasis genetics: breaking the barrier," *Trends in Genetics*, vol. 26, no. 9, pp. 415–423, 2010.

[195] I. Leigh, H. Navsaria, P. Purkis, I. McKay, P. Bowden, and P. Riddle, "Keratins (kl6 and kl7) as markers of keratinocyte hyperproliferation in psoriasis in vivo and in vitro," *British Journal of Dermatology*, vol. 133, no. 4, pp. 501–511, 1995.

[196] J. Vegfors, S. Petersson, A. Kovacs, K. Polyak, and C. Enerbäck, "The expression of psoriasin (s100a7) and cd24 is linked and related to the differentiation of mammary epithelial cells," *PloS one*, vol. 7, no. 12, p. e53119, 2012.

[197] K. Kainu, K. Kivinen, M. Zucchelli, S. Suomela, J. Kere, A. Inerot, B. S. Baker, A. V. Powles, L. Fry, L. Samuelsson, *et al.*, "Association of psoriasis to pglyrp and sprr genes at psors4 locus on 1q shows heterogeneity between finnish, swedish and irish families," *Experimental dermatology*, vol. 18, no. 2, pp. 109–115, 2009.

[198] J. G. Bergboer, P. L. Zeeuwen, and J. Schalkwijk, "Genetics of psoriasis: evidence for epistatic interaction between skin barrier abnormalities and immune deviation," *Journal of Investigative Dermatology*, vol. 132, no. 10, pp. 2320–2331, 2012.

[199] H. Michibata, H. Chiba, K. Wakimoto, M. Seishima, S. Kawasaki, K. Okubo, H. Mitsui, H. Torii, and Y. Imai, "Identification and characterization of a novel component of the cornified envelope, cornifelin," *Biochemical and biophysical research communications*, vol. 318, no. 4, pp. 803–813, 2004.

[200] S. Marrakchi, P. Guigue, B. R. Renshaw, A. Puel, X.-Y. Pei, S. Fraitag, J. Zribi, E. Bal, C. Cluzeau, M. Chrabieh, *et al.*, "Interleukin-36–receptor antagonist deficiency and generalized pustular psoriasis," *New England Journal of Medicine*, vol. 365, no. 7, pp. 620–628, 2011.

[201] H. Niehues, L. C. Tsoi, D. A. van der Krieken, P. A. Jansen, M. A. Oortveld, D. Rodijk-Olthuis, I. M. van Vlijmen, W. J. Hendriks, R. W. Helder, J. A. Bouwstra, *et al.*, "Psoriasis-associated late cornified envelope (lce) proteins have antibacterial activity," *Journal of Investigative Dermatology*, vol. 137, no. 11, pp. 2380–2388, 2017.

[202] S. Jiang, T. E. Hinchliffe, and T. Wu, "Biomarkers of an autoimmune skin disease—psoriasis," *Genomics, proteomics & bioinformatics*, vol. 13, no. 4, pp. 224–233, 2015.

[203] L.-D. Sun, H. Cheng, Z.-X. Wang, A.-P. Zhang, P.-G. Wang, J.-H. Xu, Q.-X. Zhu, H.-S. Zhou, E. Ellinghaus, F.-R. Zhang, *et al.*, "Association analyses identify six new psoriasis susceptibility loci in the chinese population," *Nature genetics*, vol. 42, no. 11, p. 1005, 2010.

[204] A. R. Djalilian, D. McGaughey, S. Patel, E. Y. Seo, C. Yang, J. Cheng, M. Tomic, S. Sinha, A. Ishida-Yamamoto, and J. A. Segre, "Connexin 26 regulates epidermal barrier and wound remodeling and promotes psoriasiform response," *The Journal of clinical investigation*, vol. 116, no. 5, pp. 1243–1253, 2006.

[205] B. Algermissen, J. Sitzmann, P. LeMotte, and B. Czarnetzki, "Differential expression of crabp ii, psoriasin and cytokeratin 1 mrna in human skin diseases," *Archives of dermatological research*, vol. 288, no. 8, p. 426, 1996.

[206] D. Bruch-Gerharz, O. Schnorr, C. Suschek, K.-F. Beck, J. Pfeilschifter, T. Ruzicka, and V. Kolb-Bachofen, "Arginase 1 overexpression in psoriasis: limitation of inducible nitric oxide synthase activity as a molecular mechanism for keratinocyte hyperproliferation," *The American journal of pathology*, vol. 162, no. 1, pp. 203–211, 2003.

[207] F. O. Nestle, C. Conrad, A. Tun-Kyi, B. Homey, M. Gombert, O. Boyman, G. Burg, Y.-J. Liu, and M. Gilliet, "Plasmacytoid predendritic cells initiate psoriasis through interferon-α production," *Journal of Experimental Medicine*, vol. 202, no. 1, pp. 135–143, 2005.

[208] S. Suomela, L. Cao, A. Bowcock, and U. Saarialho-Kere, "Interferon α-inducible protein 27 (ifi27) is upregulated in psoriatic skin and certain epithelial cancers," *Journal of Investigative Dermatology*, vol. 122, no. 3, pp. 717–721, 2004.

[209] A. Chiricozzi, E. Guttman-Yassky, M. Suárez-Farinas, K. E. Nograles, S. Tian, I. Cardinale, S. Chimenti, and J. G. Krueger, "Integrative responses to il-17 and tnf-α in human keratinocytes account for key inflammatory pathogenic circuits in psoriasis," *Journal of Investigative Dermatology*, vol. 131, no. 3, pp. 677–687, 2011.

[210] M. A. Lowes, T. Kikuchi, J. Fuentes-Duculan, I. Cardinale, L. C. Zaba, A. S. Haider, E. P. Bowman, and J. G. Krueger, "Psoriasis vulgaris lesions contain discrete populations of th1 and th17 t cells," *Journal of Investigative Dermatology*, vol. 128, no. 5, pp. 1207–1211, 2008.

[211] P. A. Jansen, D. Rodijk-Olthuis, E. J. Hollox, M. Kamsteeg, G. S. Tjabringa, G. J. de Jongh, I. M. van Vlijmen-Willems, J. G. Bergboer, M. M. van Rossum, E. M. de Jong, *et al.*, "β-defensin-2 protein is a serum biomarker for disease activity in psoriasis and reaches biologically relevant concentrations in lesional skin," *PloS one*, vol. 4, no. 3, p. e4725, 2009.

[212] J. E. Gudjonsson, J. Ding, A. Johnston, T. Tejasvi, A. M. Guzman, R. P. Nair, J. J. Voorhees, G. R. Abecasis, and J. T. Elder, "Assessment of the psoriatic transcriptome in a large sample: additional regulated genes and comparisons with in vitro models," *Journal of Investigative Dermatology*, vol. 130, no. 7, pp. 1829–1840, 2010.

[213] L. M. Roesner, P. Kienlin, G. Begemann, O. Dittrich-Breiholz, and T. Werfel, "Inflammatory marker analysis in psoriatic skin under topical phosphodiesterase 4 inhibitor treatment," *Journal of Allergy and Clinical Immunology*, vol. 140, no. 4, pp. 1184–1187, 2017.

[214] C. Castiello, G. Castellano, and A. M. Fanelli, "Meta-data: Characterization of input features for meta-learning," in *International Conference on Modeling Decisions for Artificial Intelligence*, pp. 457–468, Springer, 2005.

[215] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

[216] N. Bourbakis, A. Esposito, and D. Kavraki, "Extracting and associating meta-features for understanding people's emotional behaviour: Face and speech," *Cognitive Computation*, vol. 3, no. 3, pp. 436–448, 2011.

[217] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[218] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[219] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier, "Meta-learning by landmarking various learning algorithms.," in *ICML*, pp. 743–750, 2000.

Appendix A

Abbreviations

Abbreviations used in the report as well as a location to find more information of these concepts if possible.

- *ACA* - Available Case Analysis (Subsection 5.2.2)
- *ANOVA* - ANalysis of VAriances (Appendix B)
- *CCA* - Complete Case Analysis (Subsection 5.2.2)
- *kNN* - k-Nearest Neighbours (TO DO STILL)
- *LD* - List Deletion (Subsection 5.2.2)
- *MAR* - Missing At Random (Subsection 5.2.2)
- *MCAR* - Missing Completely At Random (Subsection 5.2.2)
- *MI* - Mutual Information (Section 3.3)
- *MICE* - Multiple Imputation Chained Equations (Subsection 5.2.2)
- *ML* - Machine Learning
- *MuI* - Multiple Imputation (Subsection 5.2.2)
- *NMAR* - Not Missing At Random (Subsection 5.2.2)
- *PTA* - Pick-l-Take-Away-r method (Subsection 4.2.2)
- *RF* - Random Forests (Subsection 4.2.2)
- *SI* - Single Imputation (Subsection 5.2.2)
- *SVM* - Support Vector Machines (Subsection 4.2.2)
- *TPOT* - Tree-based Pipeline Optimisation Tool (Subsection 4.2.3)
- *WCA* - Weighted Case Analysis (Subsection 5.2.2)

Appendix B

Biomedical Data Analysis

B.1 Introduction

At the Computational Biology department (cBio) of Biomedical Engineering (BME), many requests are made to analyse gathered data. This data usually stems from research in hospitals, but also data from other research groups and publicly known data. With the vast number of data sets that are available, a framework on data analysis would be valuable. This framework would improve efficiency of this data analysis, as it would help choosing the best analysis approaches.

Before this framework is made however, first extensive research must be done on the already available analysis techniques. Many different approaches are used by researchers. These should be searched for and compared with each other, first using literature and second by actual use. This document focuses mainly on the literature part and explores the different techniques by analysing the articles.

At first the focus will mainly be on statistical methods. Different techniques will be explained as well as the programs that can be used for these statistical techniques. The second topic will be about machine learning, and more specifically deep learning.

B.2 Statistical Analysis

Statistical analysis consists of statements derived from large data sets that are often visualized in tables, graphs and charts. These statements are made on the basis of hypotheses and the analyses that either back up or reject those. The available data sets need to have a reasonably sized population so the statistical hypotheses can be tested. [161]

The statistical approach is mainly used for drawing conclusions on the data. When focusing on summarizing the data, inductive conclusions can be made which then can be used in the biomedical world. Techniques that are specifically useful for biomedical data are all combined in the term biostatistics. [161]

To be able to use statistical analysis, the data should be quantitative. The data should be a population of data points and usually the size influences the accuracy of the outcome. Since most biomedical data stems from experiments on patients, the data usually is quantitative and can be analysed by statistical techniques. [162]

In this statistical analysis chapter, first some concepts are discussed. Since for every data set the to be used computations are different, only the basic ones are used to keep the report from irrelevant explanations. Afterwards, five different programs or frameworks are discussed for their approach to statistical analyses. Afterwards these five are compared by literature and by personal testing.

B.2.1 Statistical Research Topics

Statistical research on its own has many applications. A selection of those that are relevant for Biomedical data are discussed in here. Important in statistical analysis is the presence of hypotheses. When doing this analysis, the researcher has an expectation of the outcome. An example of this would be having a data set of people of age and number of grandchildren. The hypothesis could be: H_0*: There is a correlation between people's age and the number of grandchildren.* Then at the end this or an alternate hypothesis H_1 can be either validated or rejected. [98]

Descriptive Statistics

Descriptive statistics is a collection of methods for data analysis based around its variables. Data set variables are either numeric (categorical or grouped), ordinal (Ranked relatively) or continuous (Ranked with equal intervals). Descriptive statistics takes these variables and computes details of their distribution (Table B.1). [163] These then summarize the data for the scientist analysing it. The newly created summary can be used for further statistical research, for example by comparing different populations or variables. [161]

Table B.1: Descriptive Statistics examples [163]

Variable distribution	**Analysis terms Center oriented**	**Analysis terms Spreading oriented**
Numeric	Mode	Frequency distribution
Ordinal	Median Mode	Frequency distribution Percentiles Minimum/Maximum Range
Continuous	Mean Median Mode	Frequency distribution Percentiles Quartiles Minimum/Maximum Range Standard Deviation

Aside from the aforementioned summary of the variables, descriptive statistics also consists of methods to properly visualize those. For numeric and ordinal data, histograms (Figure B.1) and bar charts are often use to show in a easily understandable way how they are distributed. For continuous data box plots (Figure B.2) can be used as well to show specific details of the distribution. [164]

Probability Concepts

When using biomedical data, it usually contains some phenomena that seem random and affect data values. Modelling and analysis of those data phenomena is hard, due to his unpredictability. A probabilistic approach can be used to tackle this. [165]

Values that seems to have a random part can be measured multiple times. When looking at multiple measurements an average can be found within their distribution in the sample space. The probability of a value to be within a certain interval is always a number between 0 and 1. The probability of a value to be in the sample space is 1, as it the complete interval. [165]

Another aspect of probability is about the several distributions. These show the total probability for a value to be on a certain interval. There are two different kinds of probability distributions: discrete and continuous. The difference between these two is the spacing between the possible values. Values from continuous distributions can take every value, whereas values from discrete distributions are limited and the possible values have spacings between them. [98]

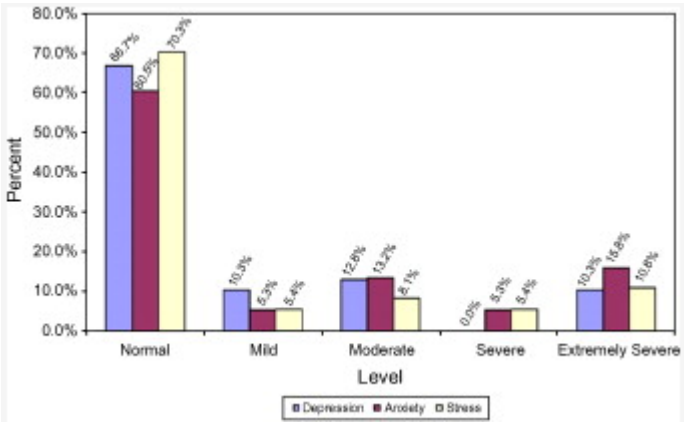


Figure B.1: An example of a histogram of ordinal data. The depression, anxiety and stress scale of women with an acute coronary syndrome is shown. the data shows that over sixty percent rates themselves as normal. [163]

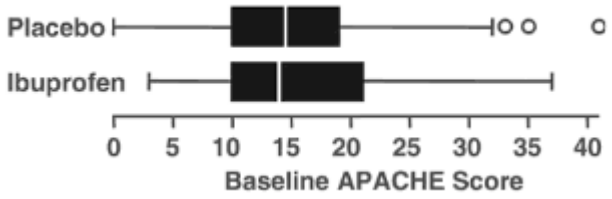


Figure B.2: An example of a box plot of continuous data. The distribution of the APACHE score (Acute Physiology And Chronic Health Evaluation) was given for users of ibuprofen and placebo. The box plot doe snot only show the minimum, maximum and mean, but also quartiles and possible outliers. [164]

Binomial distribution is the most common example of a discrete distribution (Figure B.3). This distribution can be seen as a number of n independent trials that have a success rate of p, with p being a number on the interval $(0, 1)$. This distribution has a mean of $\mu = np$ and standard deviation of $\sigma = \sqrt{np(1-p)}$. [98]

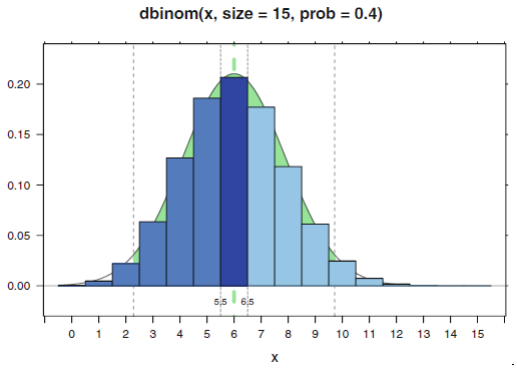


Figure B.3: An example of a binomial distribution, a discrete probability distribution. In this example there were $n = 15$ trials with a success rate of $p = 0.4$. [98]

The most common continuous probability distribution is called a normal distribution (Figure B.4). This distribution is bell shaped and symmetric around the mean μ, with a standard deviation

σ. The standard normal distribution Z has a $\mu = 0$ and $\sigma = 1$ and can be made with a modification on a regular normal distribution X: $Z = \frac{X-\mu}{\sigma}$. [98]

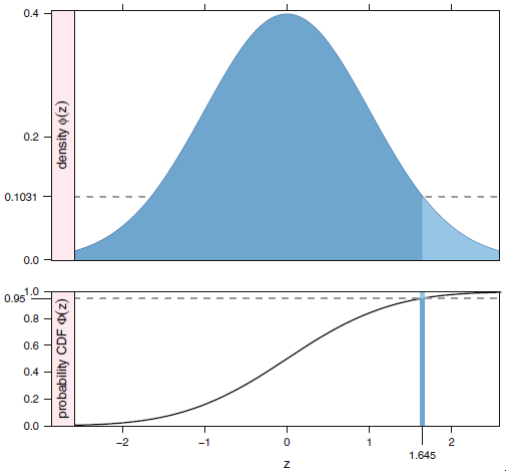


Figure B.4: An example of a normal distribution, a continuous probability distribution. This example is the standard normal distribution with mean $\mu = 0$ and standard deviation $\sigma = 1$. [98]

The third distribution that will be discussed is the student's t-distribution, another continuous distribution (Figure B.5). This one is very similar to the normal distribution discussed earlier. The student's t distribution has a sample size of n and a sample standard deviation of s. The standard distribution can be found with the formula $Z = \frac{X-\mu}{s/\sqrt{n}}$ The larger the sample size n, the closer s will get to σ of a normal distribution, so an infinite sample size n is a normal distribution. [98]

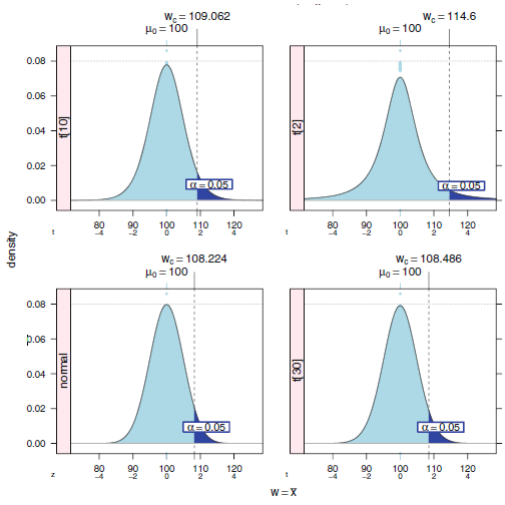


Figure B.5: An example of a 3 student's t distributions and a normal distribution. With an sample size n becoming higher, the standard deviation s comes closer to normal distribution standard deviation σ. [98]

Several tests are used to find out if a data set can be fits a certain distribution. The "Goodness of fit" is then tested. The Chi-Square and Kolmogorov Smirnov are such tests. There may be more powerful and specialized ones though, for example the Shapiro Wilk for normality testing. [98]

Interval testing

On the previously discussed probability distributions specific intervals can be computed. A confidence interval can show the interval where a certain value should be in. This can be done for an unknown mean of a distribution or an unknown population proportion in a t distribution. These intervals are made similar to this one, the confidence interval of the mean, with mean $\hat{y}$, confidence number z, percentage cut-off α, standard deviation σ and population sample n (Equation B.1). [98]

$$(\hat{y} - z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}, \hat{y} + z_{\frac{\alpha}{2}} \frac{\sigma}{\sqrt{n}}) \tag{B.1}$$

Group comparisons

The previously discussed confidence intervals can be used to find out whether two data sets are similar in terms of mean and variance. If the confidence intervals of the means of both data sets then the two data sets could be describing the same distribution. Its independence can then be proven by comparing their confidence intervals. [98]

The independence has two different parts. Two data sets can be independent comparing their means and they can be independent comparing their variances. The more important one of the two is when means are compared, as then they can be substantially different. To test whether two data sets are independent on their mean, a series of tests can be done (B.2). For these tests it makes a difference whether the unknown variance is common, different or the data is paired. [98]

Table B.2: Formulas to compute whether two samples are independent by means. The parameters $s_{\Delta\bar{y}}$ and t_{calc} can be three different values (Table B.3) [98]. The values for the mean μ,

		Tests		**Confidence Interval**	
H0	**H1**	**Rejection region**	p-value	**Lower**	**Upper**
$\mu_1 \leq \mu_2$	$\mu_1 > \mu_2$	$t_{calc} > t_\alpha$	$P(t > t_{calc})$	$((\bar{y}_1 - \bar{y}_2) - t_\alpha s_{\Delta\bar{y}},$	$\infty)$
$\mu_1 \geq \mu_2$	$\mu_1 < \mu_2$	$t_{calc} < -t_\alpha$	$P(t < t_{calc})$	$(-\infty,$	$(\bar{y}_1 - \bar{y}_2) + t_\alpha s_{\Delta\bar{y}})$
$\mu_1 = \mu_2$	$\mu_1 \neq \mu_2$	$\lvert t_{calc}\rvert > t_{\frac{\alpha}{2}}$	$P(t > \lvert t_{calc}\rvert)$	$((\bar{y}_1 - \bar{y}_2) - t_{\frac{\alpha}{2}} s_{\Delta\bar{y}}$,	$(\bar{y}_1 - \bar{y}_2) + t_{\frac{\alpha}{2}} s_{\Delta\bar{y}})$

Table B.3: The values of $s_{\Delta\bar{y}}$ and t_{calc} for data sets with common unknown variance, uncommon unknown variance and paired data. [98]

Data set	$s_{\Delta\bar{y}}$	t_{calc}
Common variance	$s_{\Delta\bar{y}} = s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}$	$t_{calc} = \frac{\bar{y}_1 - \bar{y}_2}{s_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}$
Different variance	$s_{\Delta\bar{y}} = s_{\bar{y}_1 - \bar{y}_2}$	$s_{(\bar{y}_1 - \bar{y}_2)} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$, and $t_{calc} = \frac{\bar{y}_1 - \bar{y}_2}{s_{(\bar{y}_1 - \bar{y}_2)}}$
Paired data	$s_{\Delta\bar{y}} = \bar{s}_d$	$s_{\bar{d}} = s_d/\sqrt{n}$, and $t_{calc} = \frac{\bar{d}}{s_{\bar{d}}}$

To find out whether variances of two data sets could be common, a new value F is introduced with the formula $F = \frac{s_1^2}{s_2^2}$. Since this $F = 1$ is the most ideal situation (both variances being equal), an equation can be made in which the boundaries can be specified (Equation B.2). In this equation $F_{low} = F_{1-\frac{\alpha}{2},n_1-1,n_2-1}$ and $F_{high} = F_{1-\frac{\alpha}{2},n_2-1,n_1-1}$, the upper percentage points with the $n_1 - 1$ and $n_2 - 1$ or $n_2 - 1$ and $n_1 - 1$ degrees of freedom respectively. [98]

$$(\frac{s_1^2}{s_2^2}\frac{1}{F_{low}}, \frac{s_1^2}{s_2^2}F_{high}) \tag{B.2}$$

If data is paired (Table B.2), to test if their means are paired is usually done with a t-test. A similar version, the two sample t-test, is the way to go when the distribution is known of the data set. A specific way to tests multiple groups at the same time is the one-way analysis of variance

(ANOVA). This t-test is preferred over other test due to his power compared to the other tests. [98]

Data can also not follow any specific distribution. In those cases Non-parametric statistical procedures will be used. An example would be the Sign test, when a certain median is taken and found out if there are enough data points above and under that median. This Sign test can also be used with paired populations and find out whether there is a significant number of data points from one of the sets lower than the data points from the other set. An upgraded version of this Sign test is called Wilcoxon Signed-Ranks Test, which also takes the magnitude of the data into account. [98]

If the data is not paired, the Mann-Whitney Test may be a good choice. This test is related to the two sample t-test, however uses medians instead of means. The more than two sample testing version of the Mann-Whitney Test is the Kruskal-Wallis test. This test should be used if some of the populations do not seem to be uniformly distributed enough for an ANOVA test. [98]

B.2.2 Statistical Programs

SAS

Statistical Analysis System (*SAS*) is a system used for statistical analysis. It can be used to perform a big variety of those with ease. Scientist can learn *SAS* quickly to perform small statistical computations for quick results. Within *SAS* a variety of algorithms are available for analysis. This simplicity of calling those algorithms makes *SAS* very time efficient. [166]

SPSS

SPSS is the statistical program with the most diverse users. *SPSS* is very user-friendly and easy to use when not familiar in either programming or when the there is only need for basic statistical analysis. Due to these advantages in educational research *SPSS* is widely used, especially by the social studies in need of these analysis techniques. [167]

R

R is one of the more well known statistical computing language for people who have experience in programming. It provides a language as well as statistical procedures to use for their analysis. Both *SAS* and *SPSS* created extensions for *R*, so it can be used in theirs as well. This is less intuitive, due those two being very similar. [168]

Python

Python is widely known for being an open source programming language with a relatively low bar to learn it. It was not often used for scientific analyses. Due to the maturity of several useful library as *NumPy* and *SciPy* however, *Python*'s market share is growing. Its open source element increases the usability over time as well. At last, since *Python* is also used for other projects not involving statistics, scientists may already know the language. [70]

MATLAB

Matlab is not necessarily known for its statistical analysis, but more for its good way of handling biomedical data. Biomedical scientists and engineers are often familiar with *MATLAB* and its clear way of writing commands to analyse data. For statistical analysis extensions are available, an example would be the computational statistics toolbox. Also on the internet, many freely available user-created functions can be found, which can be used to the scientist's advantage. [169]

Comparisons programs

Many comparisons are made for scientists which statistical program is best to use. A good scheme was made by Jeroen Kromme, a data scientist of Cmotions that is involved with bringing data analysis to other scientists (Figure B.6). He compares the fairly accessible programs *SA*S and *SPSS* with the more potential programs *R* and *Python*. [170]

Another article has been written by Kunal Jain that compares the commercially used *SAS* with the academic *R* and the high potential *Python*. He rates the programs on different aspects with what he shows that *Python* has the best potential of the three (Figure B.7). [171]

Combinations of the statistical programs is also possible. *SAS* and *SPSS* are very similar to each other and switching between those is not hard to do. Both of these use extensions that allows the user to work with *R* functions. This is useful as *R* has more variety in their functions compared with *SAS* and *SPSS*. [168]

These explained surveys and more [172] [173] show that these programs all have a different approach. *SPSS* is used by the scientists and companies that don't need very advanced techniques. *SAS* extends the the usability of *SPSS* by a little and is provided by support, therefore companies that need a bit more advanced statistical analyses prefer to use *SAS*. *R* is a widely known useful

	SAS	SPSS	R	Python
Advantages	1. High adoption rate in major industries 2. Flow based interface with drag and drop 3. Official support 4. Handling large datasets 5. 'PROC SQL'	1. Used a lot in universities 2. Good user interface with extensive documentation 3. Click & Play functionality 4. Writing code made easy using the ´paste´ button. 5. Official support	1. Big community who creates libraries 2. Free 3. Early adopter in explanatory and predictive modeling. 4. Easy to connect to data sources, including NoSQL and webscraping.	1. Scalability 2. General purpose language 3. Easy to learn 4. Good in machine learning 5. Big community 6. Free
Disadvantages	1. Relatively high cost 2. For not-standard options not in interface, you'll need to write the code 3. Slow adapting to new techniques 4. Different programs for visualization or Data Mining	1. Relatively high cost 2. different licenses for different functionalities. 3. Syntax limited 4. Slow adapting to new techniques 5. Slow in handling large datasets	1. Can be slow with big datasets 2. Steep learning curve 3. No official support 4. No user interface	1. Not as strong in explanatory modeling 2. Choice of version: 2.7 or 3.5? 3. No user interface 4. No official support

Figure B.6: Advantages and Disadvantages of the statistical programs *SAS*, *SPSS*, *R* and *Python*. [170]

Parameter	SAS	R	Python
Availability/Cost	3	5	5
Ease of Learning	4.5	2.5	3.5
Data Handling Capabilities	4	4	4
Graphical Capabilities	3	4.5	4.5
Advancement in Tools	4	4.5	4.5
Job Scenario	4	4.5	4.5
Customer Service Support and Community	4	3.5	3.5
Deep Learning Support	2	3	4.5
Total	**28.5**	**31.5**	**34**

Figure B.7: Grades for *SAS*, *R* and *Python* on different aspects of using a program for statistical analysis. [171]

program that is mainly used in the academic world by scientists that know a bit of programming and need more in-depth analyses. *Python* is not widely used, yet but is on the rise due to its popularity for other programming sciences, as well.

Personal comparisons

All of these statistical programs have been tested personally, too. This was done in past projects for courses at the TU/e The final comparisons of the articles match the outcome of those findings.

SPSS and *SAS* can do numerous things statistically, however it is still very basic compared to R and *Python*. Since biomedical engineers are known to process their data in *MATLAB* and *Python*, it is not worth it to pre-process data to use in *SAS* and *SPSS* when *MATLAB* and *Python* can do about the same things. If the only need is fairly simple statistical computations, *MATLAB* will be sufficient for that. If the there is need for more complex computations, *SAS* and *SPSS* most likely would not be able to do those easily anyway

If more complex statistical analyses are needed, it would be smarter to look at *R* and *Python* for use. They both work on a similar level and both have an advantage over the other. If a scientist is not familiar with complex statistical analyses or with either language, *R* might be better for him. *R* is more widely used and therefore more material is available for help. Scientists that do know *Python* or are already working with it, are better of doing the statistical analyses with *Python*. There is no need learning a new language when it would only take up more time.

B.3 Deep Learning

As computers are becoming smarter and smarter, the idea of computers learning how to tackle various problems is bigger than before. Also the size of data sets is growing two, which makes current ways of extracting information harder to use. The idea of giving computers a way to learn from experience and teach them how to cope with examples for treating real data the right way is called deep learning. [174] Deep learning is structurally used more in the biomedical environment to explain several concepts, especially considering images and the usage of neural networks. [175]

The whole idea of a computer gaining experience from finding patterns in the data is called machine learning. Deep learning is a sub section of machine learning as deep learning tries to solve problems by combining multiple small solutions. These small solutions are then connected in a network, that is often called a artificial neural network (ANN), resembling the neural network in a human brain. [174]

B.3.1 Basic Machine Learning

As deep learning is a specific kind of machine learning, knowing the important basics of machine learning is necessary. This is divided in three different parts, the task, performance measurement and the experience gained. [174]

Tasks

The task of a machine learning program is to solve problems that can not be solved by humans only. This way of solving requires a different mindset in programming. The program must be made so it knows how to solve a problem instead of manually telling the program how to solve it. Tackling a problem in the case of machine learning usually is done by having it process examples that are a collection of features. The example is a data point which details are described by those features. What needs to be done with those examples can be different. [174]

One task the program could do is specify a specific label from a number of categories. The examples all should be divided between several groups and machine learning tries to find the correct group for every example. This can be done multiple ways. Examples are in a discrete way (every example has one group) or in a probabilistic way (every example has a certain chance to be part of some groups). [174]

Another task the program could do is predict a numerical value for the examples. This is similar to classification only the answer is not limited to a number of categories. Instead it can be any value on a numerical interval. [174]

Classification and regression are the most commonly used tasks within machine learning. Other less used tasks could be to translate data to a structured representation, to detect errors in the examples or remove noise from the data. All off these tasks could be tackled using machine learning. [174]

Performance Measurement

When using machine learning, the performance of the algorithm is important. The program should not only use the data to create output, but this output must be correct as well. To measure this correctness, the accuracy is often measured. This accuracy is the percentage of examples that give the correct output. An error rate can be used as a performance measurement too, the number of examples that give an incorrect output. [174]

Data is used when training a machine learning algorithm. The algorithm is made to be as effective as possible for this training data, therefore the accuracy is supposed to be high for that data. This training data is usually just a subset of all possible data though and therefore the actual accuracy may be lower. To find this out, a separate test set can be made alongside the training set. This test set is then only used to find this accuracy. [174]

The performance measurements also require further thoughts than the accuracy and test sets. It may very well be that some categories in classifications are more important to be right (e.g. high risk patients) or the penalty for big deviations in regression may be bigger than needed. These things all need to be considered before measuring the accuracy. [174]

Experience

There are superficially spoken two kinds of machine learning, unsupervised or supervised. The difference between those is the experience they are allowed to have during its training. This training is done with a specific idea in mind that differs for these two categories. [174]

Unsupervised machine learning algorithms have the experience of a dataset with many features. With these features it can create structures within them. An example for this unsupervised learning would be clustering, to group data in different clusters according to their features. [174]

When using supervised data an additional label or target is given in addition to it regular features. With these, the algorithm knows what outcome every data point should get and learns to achieve that output with the available features. [174]

Challenges

Several challenges arise when using machine learning. Obvious challenges are using the right algorithm, having a data set with enough data that is also sufficiently detailed and and retrieving a good enough accuracy. Eventually the goal is to have a high accuracy on the created test set. The performance quality on the test set is called generalization. [174]

There are three type of errors when working with an algorithm. At first there is the training error, the error rate for training data. This training error is always better than the test error, the error rate for the test data. The difference between those is called the generalization error. When an algorithm is made, the training error should be kept as low as possible as well as the generalization error. [174]

The challenge to have the training error as low as possible is to not under fit, which means that the algorithm does not achieve a low enough training error. The challenge for generalization error is to no over fit, which means that the test error is not close enough to the training error. [174]

B.3.2 Neural Networks

The name neural networks stems from the connections between neurons in the brain, a big research area in modern day biology. In the mathematical world, artificial neural networks can be made for modelling data analysis. These models that are created with the neural networks use a specific equation (Equation B.3). The output (computation) is a combination of already stored information (storage), new information (transmission) and process it with available methods in the model (processing). [176]

$$\text{computation} = \text{storage} + \text{transmission} + \text{processing}. \tag{B.3}$$

Zooming closer into neural networks, the idea of a single neuron can be shown (Figure B.8). These neurons have three different important parts. The first part are the input values x_i, which are first modified a bit with weights w_i. The third part is the actual function f that creates output for the weighted input values. These neurons then put together create a network with eventually an output (Figure B.9). [176]

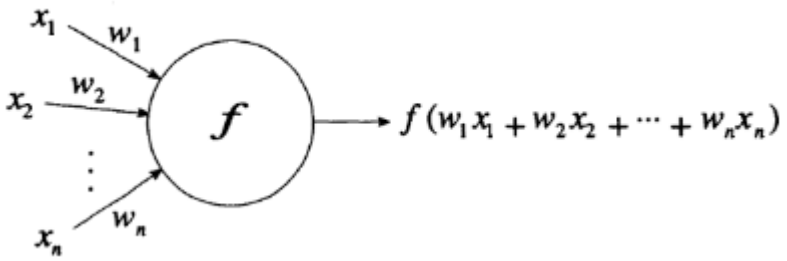


Figure B.8: An example of a basic neuron in a neural network. Input $x_1...x_n$ together with a weight $w_1...w_n$ are put into function f to create output. [176]

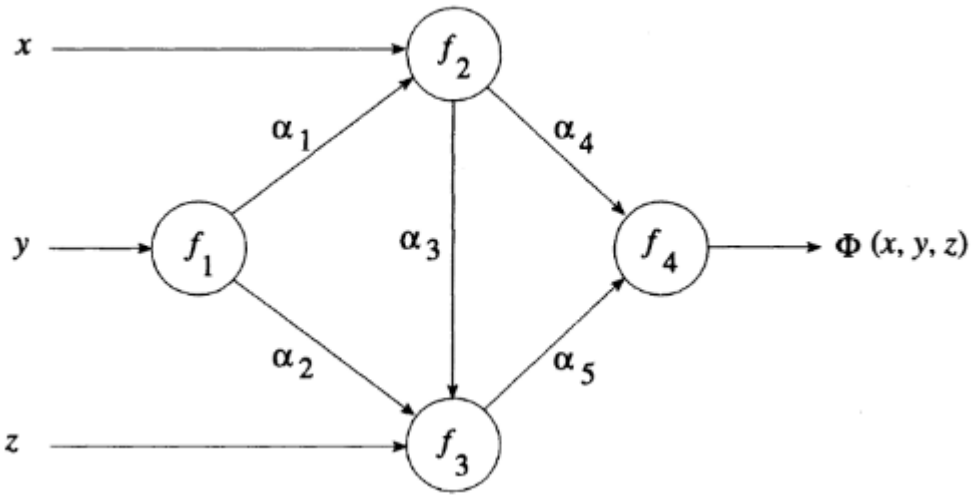


Figure B.9: An example of a neural network consisting of neurons (Figure B.8). [176]

These created networks are divided in layers. The first layer is the collection of all neurons that have an external input value. The last layer is the collection of all neurons that generate output values. All layers in between are called hidden layers, as a user cannot see those neurons from the outside. [177]

As explained in the section about machine learning, the neural network must be trained (Section B.3.1). This can be done after the architecture is defined. Again, with the data, a training and a test set must be made, as well. With those, the neural network can be properly trained to create the right output when giving values as an input. [177]

Convolutional Neural Networks

A more specific variant of neural networks is a convolutional neural network (CNN). CNNs combine neural networks and convolutions to reduce noise in the signal and approximate the actual input better. Convolutions use kernels to use the values close to the input it wants to approximate. It also uses pooling, which makes the input smaller by approximating a location and its nearby inputs into one. [174]

B.3.3 Deep Learning Frameworks

A wide selection of deep learning frameworks are available. Most of these frameworks are implemented in *Python*, with some exceptions. A selection of the five most popular are explained, together with their advantages and disadvantages.

Scikit-learn Python

Python itself has a package that can be used for machine learning, named Scikit-learn. It is high-level interactive and is maturing in a quick rate, due to its usage in both the academic world as in the industry. It also has some C++ libraries incorporated, but makes most use of the three packages *NumPy*, *SciPy* and *Cython*. Due to his high-level and ease of use, a drawback is its computational efficiency and therefore is not the best choice for testing large data sets. [74]

Theano

Theano is one of the open source machine learning frameworks written in *Python*. [178] Its programming is declarative. [179] This one uses the famous *NumPy* syntax for higher computation speed in its language. It was developed mainly to simplify the implementation of the the algorithms used for well performing machine learning. Two tasks that are intensive for the processor, either training a multi-layer neural or a convolutional network. *Theano* works through a pipeline on compilation (Figure B.10). At first it puts the graph in standard form (Canonicalization). Next it improves the stability of the computation (stabilization). Thirdly it replaces expressions with faster ones (Specialization). Fourth it moves the computation to the GPU, if compiled for GPU (GPU transfer). In the last step it loads *Python* modules with specialized implementations (Code Generation). [178]

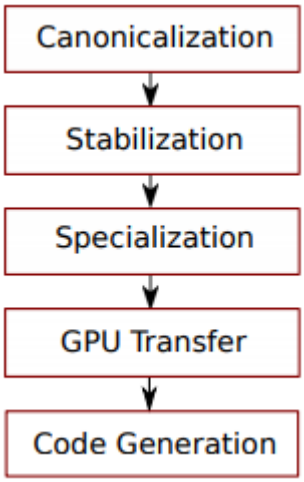


Figure B.10: The steps *Theano* takes in compilation for functions for GPU. [178]

Theano has been seen as a base for many other neural network creation frameworks. Examples are *Keras*, *Lasagne*, *GroundHog*, *Blocks* and *Pylearn2*. It utilizes from the benefits of being written in *Python*, such as creating a friendly environment for fast and easy interaction with data. However, since threading is done by *Python*, it is not possible to have multiple interpreters going on at the same time. The graph optimization time, code compilation and memory usage time all can be improved. [180]

TensorFlow

TensorFlow, also open sourced, is developed for experimentation on new models, training those models with the help of large datasets and at last move those models into production. [181] *TensorFlow* is declarative and made in C++, while also having interfaces in *Python*. [179] Google makes regular use of *TensorFlow*, as an example. However many more programmers use TensorFlow for their applications. It is a follow-up of the earlier system *DistBelief* and simplified and generalized that. *TensorFlow* is mainly supporting training and inference on a larger scale. It uses a data flow graph for representing the computation within the algorithm and the operation state of the algorithm. [181]

While executing a *TensorFlow* application, two different phases can be distinguished. The first phase of the two makes a to be trained neural network and the update rules in the form of a data flow graph. The places for input are reserved by place holders for state representation. The second phase is the earlier neural network after optimization. Because of these two phases, the execution phase can be optimized with the information of the computation. There will not be any intermediate results slowing down the process. [181]

Not only *Theano*, but *TensorFlow* as well has some wrappers, too. *Keras* (also for *Theano*) and *Pretty Tensor* both use *TensorFlow* as a base for creating neural networks. [179]

Caffe

Caffe is a framework based on the *C++* library together with *Python* and *MATLAB*. *Caffe*, too, is open sourced and its programming is imperative. [179] It is very useful for researches as its code is modular and its network definitions are well separated. The test coverage of *Caffe* is useful as well, as every new module has a test and is not accepted without a test. The bindings with *Python* and *MATLAB* makes constructing networks and classifying inputs easier. At last does *Caffe* provide pre-trained reference models. This makes it easier to reproduce research. [182]

Torch

Torch is an object-oriented framework for machine learning and is implemented in *Lua*, but also has interfaces in *C*. *Torch* is also imperative. [179] A modular strategy was used to simplify any modifications of existing algorithms. There are four classes chosen. The first class is about data handling (*DataSet*). The second one is the black box that gives an output (*Machine*). The third is the class that is used to find the optimal set of parameters (*Trainer*). The last one is about printing the measures of interest (*Measurer*). These classes give a very clear general idea. Several examples of usable machines are the gradient machines, support vector machines and distributions. [183]

Comparing Frameworks

Kovalev compared the likings of *Theano* (with *Keras* extension), *Torch*, *Caffe*, *TensorFlow* and the not discussed *Deeplearning4J* based on convergence and prediction time, classification accuracy and framework complexity (lines of code). Both the depth (number of internal layers) and the width (number of neurons within fixed number of layers). When regarding training, prediction and testing times, all but *Deeplearning4J* were performing quite well. When watching Classification accuracy, *Torch* and *TensorFlow* performed visibly unstable and *Caffe* seemed to be a bit worse overall. The last comparison, framework complexity was best for *Theano*, followed by *TensorFlow* and then *Caffe*. From best to worst the article rated the following order: Theano (with *Keras*), *TensorFlow*, *Caffe*, *Torch*, *Deeplearning4J*. [184]

The creator of *Caffe* compared its own framework with other known frameworks. The focus was mainly on basic parts of the programs, namely BSD licence, core and binding languages CPU and GPU coverage, Open source, training and the availability of pre-trained models. *Caffe* has the advantage in comparison with the earlier described *Theano* and *Torch* regarding binding languages and the availability of pre-trained models. [182]

Framework	License	Core language	Binding(s)	CPU	GPU	Open source	Training	Pretrained models	Development
Caffe	BSD	C++	Python, MATLAB	✓	✓	✓	✓	✓	distributed
cuda-convnet [7]	unspecified	C++	Python		✓	✓	✓		discontinued
Decaf [2]	BSD	Python		✓		✓	✓	✓	discontinued
OverFeat [9]	unspecified	Lua	C++,Python	✓				✓	centralized
Theano/Pylearn2 [4]	BSD	Python		✓	✓	✓	✓		distributed
Torch7 [1]	BSD	Lua		✓	✓	✓	✓		distributed

Figure B.11: The comparisons between different machine learning frameworks from *Caffe* creator point of view. [182]

Bahrampour tested *Caffe*, *Theano*, *Torch* and the not discussed *Neon* on their extensibility, hardware utilisation and speed for both CPU and GPU usage. He stated that *Caffe*, *Theano* and *Torch* were at that moment the top three well developed and used frameworks. *Neon* was added because of its potential. The speed was measured by a forward time (check time it takes for a pre-selected batch) and a gradient computation time (time for each measurable parameter). The conclusion of the research had six parts: *Theano* and then *Torch* are the most extensible. For both CPU- and GPU-based training and deployment *Theano* and *Torch* are fighting for the number one spot. *Torch* would benefit from expansion in both documentation and debugging, though. [185]

Another research that compared deep learning framework was Shi. The frameworks *Caffe*, *TensorFlow* and *Torch* were tested by him, as well as the not discussed *CNTK* from Microsoft and *MXNet*. The tests that were done about speed were on CPUs, one GPU and multiple GPUs on both synthetic and real data. The overall outcome was that *Caffe*, *CNTK* and *MXNet* performed better than *TensorFlow* and *Torch*, although *TensorFlow*'s production ramped up with more threads. [186]

Similar to Bahrampour [185], Fox compared *TensorFlow*, *Caffe*, *Theano*, *Torch* and the not discussed *CNTK*, *Deeplearning4j*, *MXNet* and *H2O*, for them being open-source, relatively mature and adopted by the community. He mainly discussed the different frameworks and afterwards made a table with all those findings (Figure B.12). [187]

Parvat had a more listings approach in comparing the frameworks *Theano*, *Caffe*, *Torch*, *TensorFlow* and the not discussed *Deeplearning4J* and *NVIDIA cuDNN*. This time they were compared on their platform, interface, modelling capability, support for *CUDA*, *OpenMP*, *OpenCL* and support for pre-trained models. He concluded that *TensorFlow* was very flexible, due to being more than a deep learning framework. *Theano* is very helpful for creating models fast, especially with the libraries, and therefore *TensorFlow* and *Theano* are the most flexible. Torch is good because of the pre-trained models, however misses being written in a mainstream language. *Caffe* is useful when working with images, and also has a lot of pre-trained CNN models. *Deeplearning4j* has its advantage being the only one working with *Java* and *Scala*. At last *TensorFlow* has the advantage of being able to work with a distributed environment. [188]

A last article to discuss about comparing deep learning frameworks is written by Erickson. Erickson focused on language, environment, speed and maturity. *Caffe*, being the most mature one and very fast, has a disadvantage when tuning hyperparameters. *TensorFlow* is regarded as hard to use directly, however has good performance and tools for help. *Theano* is similar as *TensorFlow*, in that it has good performance, however is a bit harder to learn. Both of these frameworks can be made more user friendly with a library as *Keras*. *Keras* uses the performance of either *TensorFlow* or *Theano* and can be used easily to create models with efficient code. Also *Torch* is given a good maturity level and good documentation. [189]

As can be seen by the many articles written about these frameworks, there is not a clear distribution as to which framework should be used when. It seems that *Theano* and *TensorFlow*, especially with help from libraries such as *Keras* are regarded a bit better than the others. On the other hand *Caffe* has some advantages, too.

Platform	Tensorflow	CNTK	Deeplearning4j	MXNet	H2O	Caffe	Theano	Torch
Release Date	2016	2016	2015	2015	2014	2014	2010	2011 (deep learning)
Core Language	C++	C++	C++	C++	Java	C++	C++	C
API	C++, Python	NDL	Java, Scala	C++, Python, R, Scala, Matlab, Javascript, Go, Julia	Java, R, Python, Scala, Javascript, web-UI	Python, Matlab	Python	Lua
Synchronization Model	Sync or async	Sync	Sync	Sync or async	Async	Sync	Async	Sync
Communication Model	Parameter server	MPI	Iterative MapReduce	Parameter server	Distributed fork-join	N/A	N/A	N/A
Multi-GPU	✓	✓	✓	✓	✓	✓	✓	✓
Multi-node	✓	✓	✓	✓	✓	✗	✗	✗
Data Parallelism	✓	✓	✓	✓	✓	✓	✓	✓
Model Parallelism	✓	N/A	✗	✓	✗	✗	✓	✓
Deep Learning Models	DBN, CNN, RNN	DBN, CNN, RNN	DBN, CNN, RNN	DBN, CNN, RNN	DBN	DBN, CNN, RNN	DBN, CNN, RNN	DBN, CNN, RNN
Programming Paradigm	Imperative	Imperative	Declarative	Both	Declarative	Declarative	Imperative	Imperative
Fault Tolerance	Checkpoint-and-recovery	Checkpoint-and-resume	Checkpoint-and-resume	Checkpoint-and-resume	N/A	N/A	Checkpoint-and-resume	Checkpoint-and-resume
Visualization	Graph (interactive), training monitoring	Graph (static)	Training monitoring	None	None	Summary Statistics	Graph (static)	Plots

Figure B.12: Comparisons of eight frameworks. The comparisons were implementation based.

Appendix C

Case Study 1 - Bariatric Co-morbidities

C.1 Introduction

The number of bariatric surgeries is increasing. worldwide. Although initially thought otherwise, this type of surgery has added benefits on top of losing weight. Among those benefits the remission of metabolic co-morbidities can be named. Due to binary labelling of those co-morbidities, valuable information is lost. On top of that the labelling is not clearly defined either. To obtain more and better results, this binary labelling could be replaced by a continuous severity score. Ruben Deneer conducted a research on trying to achieve a successful replacement.

The available data used for the research stemmed from the Catharina Hospital in Eindhoven (Subsection C.1.1). This extensive data set was a combination of two data sets and consisted of 41 markers measured pre- and post-surgery for 2367 patients that underwent gastric sleeve or gastric bypass surgery. These 41 markers are divided in several sub-panels that each describe different processes in the body. Conditions for these co-morbidities that were tested for the severity score were type II diabetes mellitus (T2DM), hypertension and dyslipidemia. Extensive literature research was done to connect them with 41 markers to find possible relations (Subsection C.1.2).

To better specify the research on the data, a main goal was created: *to use data mining techniques to develop score that can objectively quantify the severity of co-morbidities present in bariatric patients based on biomarkers, both before and after surgery.* [24] This means that it is not the goal to predict the outcome of the two types of surgery, but it is to quantify the improvement of the co-morbidities before and after the surgery.

Since the data is not available other than within the hospital, a second research is proposed to complement Ruben Deneer's. The creation of this research consists of four parts. The first part was creating a research question, hypothesis and an own analysis approach without the knowledge of Deneer's (section C.2). Since no data is available, this can only be made globally. It should be detailed enough to compare it with Deneer's own approach, though. Secondly a summary is given what was done in Deneer's research (section C.3). Thirdly the two are compared and remarks are given on the original approach (section C.4). At last both are combined to create a final proposal for a possible follow-up research (section C.5).

Over the course of this proposal report, many references are made to the thesis of Ruben Deneer. These will not always be included for redundancy. Only when things are directly quoted from the thesis, a reference will be placed. Examples would be tables or goals.

C.1.1 Data Sets

Two data sets were used in the study. The first one is called "The Dutch Audit for Treatment of Obesity" (DATO). This data set is a national database that houses all registrations and health

statuses of pre- and post treatment bariatric surgery patients in the Netherlands. Several basic variables are noted, such as height, weight, BMI and date. Before surgery, the co-morbidities T2DM, hypertension and dyslipidemia were given a binary label of "Yes/No". After surgery they were given one of the following labels:

1. **Cured** No co-morbidity any more
2. **Improved** Less affected by co-morbidity
3. **Same** No change in co-morbidity status
4. **Worse** More affected by co-morbidity
5. **Denovo** Diagnosed co-morbidity while not present before surgery
6. **Not present** No co-morbidity present

The second data set came from a laboratory database, stored in health records. This extensive data set consisted of 3 clinical and 38 blood markers measured pre- and 6, 12 and 24 months post-surgery. The tests pre-surgery had some additional markers on top of the 41 ones. These markers can be divided in the following categories: (Table **??**) Complete blood count, liver function, kidney function, inflammation, lipid spectrum, coagulation, glucose metabolism, thyroid function and at last minerals and vitamins. The data sets of the patients that underwent bariatric surgery can be extracted from these.

One part of the research is to combine these two data sets. Some challenges arise when doing so. Such a challenge is obviously to find the right connection between them, using the survey and lab data of the same patient. These challenges must be solved before asking the question what markers can say about the severity of co-morbidities.

A second challenge would be to define what must be done with non-matching data. Pre-treatment for example more markers were used than post-treatment. These missing ones might be more useful for scoring the co-morbidity severity than the known markers. There also might be measurements that were missing or corrupted, whereas other markers might still be useful enough for a result. How to tackle this missing data challenge should be defined properly.

C.1.2 Co-morbidity Marker Relation

For some of the markers it is clear they affect the severity score of a co-morbidity. Co-morbidity is diagnosed from these markers and therefore a relation is imminent. For others more subtle relations are present, that in first sight would not be recognised. Both are important for assigning a co-morbidity severity score.

All three co-morbidities can be derived from specific markers. T2DM is known for higher glucose and insulin levels in the blood, while hardly finding any C-peptides. Ideally these would be used to find T2DM, however these values vary significantly over the day. Therefore help from other markers should be helpful. Hypertension is best measured with blood pressure measurements (BP). These however are not reliable for the nervous patients in the hospital and other markers must be found for help. Thirdly dyslipidemia obviously can be found using markers in the lipid spectrum (Table **??**), however there may be other markers contributing to dyslipidemia detection, as well.

For T2DM, hypertension and to a lesser extend dyslipidemia, more knowledge on relations between them and markers would benefit the creation of a severity score. Literature research is done to understand more of those subtle relations and know that those exist when trying to create a morbidity score. The outcome of this literature search shows that quite some relations can be found between co-morbidities and the markers (Table C.2). For most it is definitely worth to investigate any relation.

Table C.1: The markers present in the bariatric laboratory data set [24]

	Before Surgery/Pre-Op/Screening	After Surgery/Post-Op/Follow-up
Complete blood count	hemoglobin	hemoglobin
	hematocrit	hematocrit
	erythrocytes	erythrocytes
	mean corpuscular hemoglobin	mean corpuscular hemoglobin
	mean corpuscular volume	mean corpuscular volume
	thrombocytes	thrombocytes
	leukocytes	leukocytes
Liver function	bilirubin	bilirubin
	aspartate aminotransferase	aspartate aminotransferase
	alanine aminotransferase	alanine aminotransferase
	lactate dehydrogenase	lactate dehydrogenase
	alkaline phosphatase	alkaline phosphatase
	gamma-glutamyltransferase	gamma-glutamyltransferase
Kidney function	urea	urea
	creatinine	creatinine
	potassium	potassium
	sodium	sodium
	calcium	calcium
	phosphate	phosphate
	albumin	albumin
Inflammation	C-reactive protein	C-reactive protein
Lipid spectrum	total cholesterol	total cholesterol
	high-density lipoprotein-cholesterol	high-density lipoprotein-cholesterol
	total/high-density cholesterol ratio	total/high-density cholesterol ratio
	low-density lipoprotein-cholesterol	low-density lipoprotein-cholesterol
	triglycerides	triglycerides
Coagulation	prothrombin time	prothrombin time
Glucose metabolism	hemoglobin A1c (IFCC)	hemoglobin A1c (IFCC)
	glucose	glucose
	insulin	-
	C-peptide	-
Thyroid function	parathyroid hormone	parathyroid hormone
	thyroid-stimulating hormone	-
	free T4	-
	cortisol	-
Minerals and vitamins	iron	iron
	ferritin	ferritin
	folic acid	folic acid
	zinc	-
	magnesium	-
	vitamin A	-
	vitamin B1	vitamin B1
	vitamin B6	vitamin B6
	25-OH vitamin D	25-OH vitamin D
	vitamin B12	vitamin B12

Table C.2: Relations found in literature between the co-morbidities and markers. [24]. The number shows how many citations were found that state a relation exist, nothing means no relation. A 'D' means that the disease is diagnosed from those markers.

	Diabetes	**Hypertension**	**Dyslipidemia**
hemoglobin	3	1	
hematocrit	2	2	
erythrocytes	1	1	
MCH		1	
MCV		1	
thrombocytes			
leukocytes	4	3	1
bilirubin	3	1	1
ASAT	1	1	1
ALAT	4	1	1
LD	1		
Alkaline phospatae	2		
gamma GT	4	2	1
urea	1	3	
creatinine		2	1
potassium		2	
sodium		1	
calcium	2	1	1
phosphate			
albumin			
CRP	3	4	1
cholesterol		2	D
HDL-cholesterol		2	D
chol/HDL ratio			D
LDL-cholesterol		2	D
triglycerides		2	D
prothrombin time			
hemoglobin A1c (IFCC)	D		
glucose	D	2	
parathormone	1	2	
iron			
ferritin	3	1	1

C.2 First Proposal

To start of a research proposal, the first thing that needs to be done is to create a research question that fits the intended goal. This main question can then be divided in several sub questions for explaining the steps how to answer it. Next hypotheses should be made for the main and sub questions. Third and last the methods of how to answer the research questions are discussed. Since this is only a first proposal, the methods part will be limited, and extended in future proposals.

C.2.1 Research Questions

As stated earlier (Section C.1) the goal was to develop a severity score for the co-morbidities using data mining. A research question for this goal would then be:

> *Can a severity score for co-morbidities present in bariatric patients based on biomarkers be developed?*

This main research question includes all aspects of the goal. It should be specified in several different sub-questions ot make it clearer. A first sub-question could be about the relation between co-morbidities and biomarkers, which is a major part on making this score.

1. *Which relations are present between co-morbidities and the biomarkers?*

A second sub-question should be about retrieving a score for the co-morbidities from the biomarkers. This score retrieval should use the outcome of the first sub-question to efficiently find the correct score.

2. *How can a severity score of co-morbidities be obtained from biomarkers?*

This second question mainly focuses on techniques to achieve a sufficient answer to the main research question. When both sub-questions are sufficiently answered, then the main one should definitely be, as well.

C.2.2 Hypothesis

The data set available for this research is quite extensive. The number of biomarkers is high, there are many relations between them, co-morbidities are known and a high number of patients has been tested for both. These pre-conditions are very beneficial to finding positive answers for the research questions.

Knowing many relations between the co-morbidities and biomarkers beforehand implies that a lot of them should be present in the data set. to the first sub-question which relations are present, the answer would be two-fold. First strong relations between the co-morbidities and on the other hand biomarkers that help diagnosing them are expected. This would for example be the case for dyslipidemia and biomarkers of the lipid spectrum. Weaker relations are expected for all found in literature (Table C.2).

As for the second sub-question how a score can be defined from biomarkers, several different options are possible. When knowing there is a relation between biomarkers and the co-morbidity, by machine learning combined with regression a function can be made predicting a score outcome for the biomarkers. This score is then compared with the categorical labelling of co-morbidities to find out if they match. Through training the model and afterwards testing it, the score is validated.

With the hypotheses for both sub-questions the main research question can be answered, too. The hypothesis for this one would be that it is possible to find a severity score with the biomarkers. A model can made using the biomarkers as an input and retrieving a severity score as output.

C.2.3 Methods

Since the materials are all explained in Deneer's thesis, that part and most of the pre-processing is skipped in methods. One thing that needs to be discussed is the handling of markers that are measured pre-surgery, but not post-surgery. These could be removed as being redundant. They may, however show a significant difference between availability or absence of a co-morbidity. This is a vital result for possible further bariatric surgeries and co-morbidity scoring. Therefore at least some tests should be done with those pre-surgery biomarkers. For post-surgery measurements they can be removed from the tests or they can be given a dummy value as being unknown.

The first research question can be easily and efficiently answered with statistical analysis. This approach can be used to compare two groups of values, whether the co-morbidity is present or not. Since for this question a score is not yet needed, dividing the patients between having a co-morbidity or not suffices. The markers that show a significant difference in mean or variance should be noted and given extra care in future researching to create a score. Aside from comparing original measurements of the markers, combinations between them should be tested for statistical analysis as well. Some are already present, for example the *cholesterol / HDL-cholesterol ratio*. There may be more interesting combinations that could show a significant difference for either of the three co-morbidities.

Since the data needs some manipulation to find those markers that are significantly different for presence or absence of co-morbidities, a more advanced statistical program that can do those manipulation would be best to use. This means that the best choices would be *R*, *MATLAB* or *Python*. *Python* with *SciPy* was chosen of those three for also being a good choice for researching the second sub-question and because the actual users knows *Python* quite well. The outcome of this research can most likely immediately used in the second one for the next sub-question

The second research question is a bit more complex to answer. The quickest one is to use a way of machine learning to create this score. Some sort of regression is the first thing that comes to mind and simple support vector machines (SVMs) seem to be a good way to start. However obtaining a continuous score from categorical, or in the case of pre-surgery even binary, labels. A score interval needs to be set up and these binary or categorical values should be translated to a certain score on the interval. The way to score these values can be proposed, but most likely needs more insight to find out what is best. At first just a categorical learning system seems best. Both a selection of from sub-question resulting useful markers should be used as well as the complete set to find a good score.

At the start of researching the second sub-question a standard machine learning framework would suffice. This means that the aforementioned *Python* with the *scikit-learn* package would be best to start with. When initial results indicate that a certain way of handling this data in another way would be more efficient, *Python* is used by many other machine learning programs, such as Theano, TensorFlow and both their packages. Therefore *Python* would be a great way to start the project of which the direction can change heavily.

C.3 Original research

Now that an initial proposal is made, the original research should be discussed. The research is divided into two parts. The first is all about the material and how this was preprocessed. The second part focused more on modelling using the data sets. Multiple approaches have been used to show different aspects of the data.

C.3.1 Preprocessing

As discussed earlier (subsection C.1.1), the data was retrieved from two data sets, the DATO and lab set. These two were not perfectly lined up and therefore merging those required some specific rules. At first, any entries that were double on the same date were removed if having conflicting information. Entries that had surgery before 2012 were removed, too, due to not having a pre-screening test. Secondly all DATO and lab sets were paired up if they did not have more than 90 days between them. When double matches occurred, the closest ones were taken. Also only pairs were taken when they were on one of the four periods (pre-, 6, 12 or 24 post-surgery) and others were omitted.

After merging both data sets the lab data was prone to preprocessing. Ten biomarkers were dropped from the set, due to missing values. Five (Zinc, vitamin A, free T5, methylmalonic acid, LDL) were dropped due to severe missing data or explanations by other data. Five others (TSH, insulin, C-peptide, magnesium and cortisol) were dropped due to only being available in pre-surgery screening. CRP was dichomotized, due to being truncated and other variables were changed to their cut-off value when only a small portion of them was truncated, too. Inconsistencies in DATO meant that the patient would be dropped. After all these markers were being removed, a total of 60 variables were present in the data set (figure **??**) and a number of 2367 patients with a 2.35 of data sets per patient on average (figure **??**). The number of variables used for modelling was reduced to 49. From those, a number of variables were removed again. Weight and height (incorporated in BMI), vitamins (were supplemented), iron (variation too high), year, type of and months before/after surgery (no value, or found elsewhere) were all removed. Furthermore Creatinine was replaced by glomerular filtration rate (GFR), the ASAT/ALAT ratio was added, the prothrombin time (PT) was replaced by the international normalized ratio (INR) and at last calcium was corrected by adding an albumin factor.

C.3.2 Modelling

After preprocessing the data, a series of steps should be taken to create a sufficient model for the issue. First an ordinal outcome definition was made for scoring the co-morbidities. Next, statistical learning models were created, based on logistic regression and all aspects of this model were discussed to find, validate and visualize a correct model.

To make one scale for all co-morbidities a new ordinal outcome was created. This scoring was made with three different categories, being: no, one or multiple co-morbidities present. The choice for this scoring was made with the idea that people with more co-morbidities also had a higher severity compared with patients that had less or none. The multiple co-morbidities category was a merge from having two and three, in order to make the categories more balanced in size. This way none has most patients (1329) and one and multiple still had a high enough number (569 and 469 respectively).

To create a model in this research, a classification method should be used. A well known one for that was logistic regression. Maximum likelihood estimation was used to estimate regression coefficients for the variables. Multicollinearity had to be limited as much as possible, as it was assumed to be absent for logistic regression. A correlation matrix was created to find multi-collinearity. Special interactions were included as well, for example the Body mass index (BMI) and presence for co-morbidities as being a prerequisite for the surgery. For ordinal outcomes of the logistic regression, both proportional odds (PO) and continuation ratio (CR) were used. 10-fold cross validation was done to validate the model and an assessment of model fit was done

Variable	Description	Type	Used in modeling
patient	Unique patient ID	character	No
OK.date	Date of surgery	date	No
period	Time period, i.e. Pre-op, 6 Months, 12 Months or 24 Months	factor	Yes: predictor
time	Months before or after surgery	integer	No
gender	Male or female	factor	Yes: predictor
age.at.surgery	Age at sugery	continuous	Yes: predictor
height	Height in meters	continuous	No
weight	Weight in kg	continuous	No
BMI	BMI in kg/m^2	continuous	Yes: predictor
surgery	Gastric sleeve or bypass	factor	No
hemoglobin	Hemoglobin in mmol/L	continuous	Yes: predictor
hematocrit	Hematocrit in L/L	continuous	Yes: predictor
erythrocytes	Erythrocytes in /pL	continuous	Yes: predictor
MCH	Mean corpuscular hemoglobin in fmol	continuous	Yes: predictor
MCV	Mean corpuscular volume in fL	continuous	Yes: predictor
thrombocytes	Thrombocytes in /nL	continuous	Yes: predictor
leukocytes	Leukocytes in /nL	continuous	Yes: predictor
glucose	Glucose in mmol/L	continuous	Yes: predictor
bilirubin	Bilirubin in μmol/L	continuous	Yes: predictor
ASAT	Aspartate aminotransferase in U/L	continuous	Yes: predictor
ALAT	Alanine aminotransferase in U/L	continuous	Yes: predictor
ASAT.ALATratio	Ratio between ASAT and ALAT	continuous	Yes: predictor
LD	Lactate dehydrogenase in U/L	continuous	Yes: predictor
ALP	Alkaline phosphatase in IU/L	continuous	Yes: predictor
GGT	Gamma Glutamyltransferase in U/L	continuous	Yes: predictor
urea	Urea in mmol/L	continuous	Yes: predictor
creatinine	Creatinine in μmol/L	continuous	Yes: predictor
potassium	Potassium in mmol/L	continuous	Yes: predictor
sodium	Sodium in mmol/L	continuous	Yes: predictor
calcium	Calcium in mmol/L	continuous	Yes: predictor
phosphate	Phosphate in mmol/L	continuous	Yes: predictor
albumin	Albumin in g/L	continuous	Yes: predictor
CRP	C-reactive protein <6 or ≥ 6 mg/L	factor	Yes: predictor
cholesterol	Total cholesterol in mmol/L	continuous	Yes: predictor
triglycerides	Triglycerides in mmol/L	continuous	Yes: predictor
HDL	High-density lipoprotein in mmol/L	continuous	Yes: predictor
cholHDLratio	Total cholesterol to hdl ratio	continuous	Yes: predictor
INR	International Normalized Ratio	continuous	Yes: predictor
HbA1c	Hemoglobin A1c in mmol/mol	continuous	Yes: predictor
PTH	Parathyroid hormone in pmol/L	continuous	Yes: predictor
iron	Iron in μmol/L	continuous	No
ferritin	Ferritin in μg/L	continuous	Yes: predictor
folate	Folate in nmol/L	continuous	Yes: predictor
vitB1	Vitamin B1 in nmol/L	continuous	No
vitB6	Vitamin B6 in nmol/L	continuous	No
vitD25OH	Vitamin D 25-hydroxy in nmol/L	continuous	No
vitB12	Vitamin B12 in pmol/L	continuous	No
CKD.EPI	CKD-EPI eGFR in $ml/min/1.73m^2$	continuous	Yes: predictor
dia.full	Diabetes label as recorded in DATO	factor	No
hyp.full	Hypertension label as recorded in DATO	factor	No
dys.full	Dyslipidemia label as recorded in DATO	factor	No
dia.bin	Diabetes label converted to binary	logical	Yes: outcome
hyp.bin	Hypertension label converted to binary	logical	Yes: outcome
dys.bin	Dyslipidemia label converted to binary	logical	Yes: outcome
no.of.comorbs	Number of co-morbidities (out of 3)	continuous	No
noneVSrest	Binary label indicating if there are no co-morbidities	logical	Yes: outcome
multipleVSrest	Binary label indicating if there are multiple co-morbidities	logical	Yes: outcome
oneVSrest	Binary label indicating if there is one co-morbidity	logical	Yes: outcome
comorb	Ordinal co-morbidity label	factor	Yes: outcome
classlabel	Label indicating which combination of co-morbidities are present	factor	No

Figure C.1: All variables present after pre-processing [24]

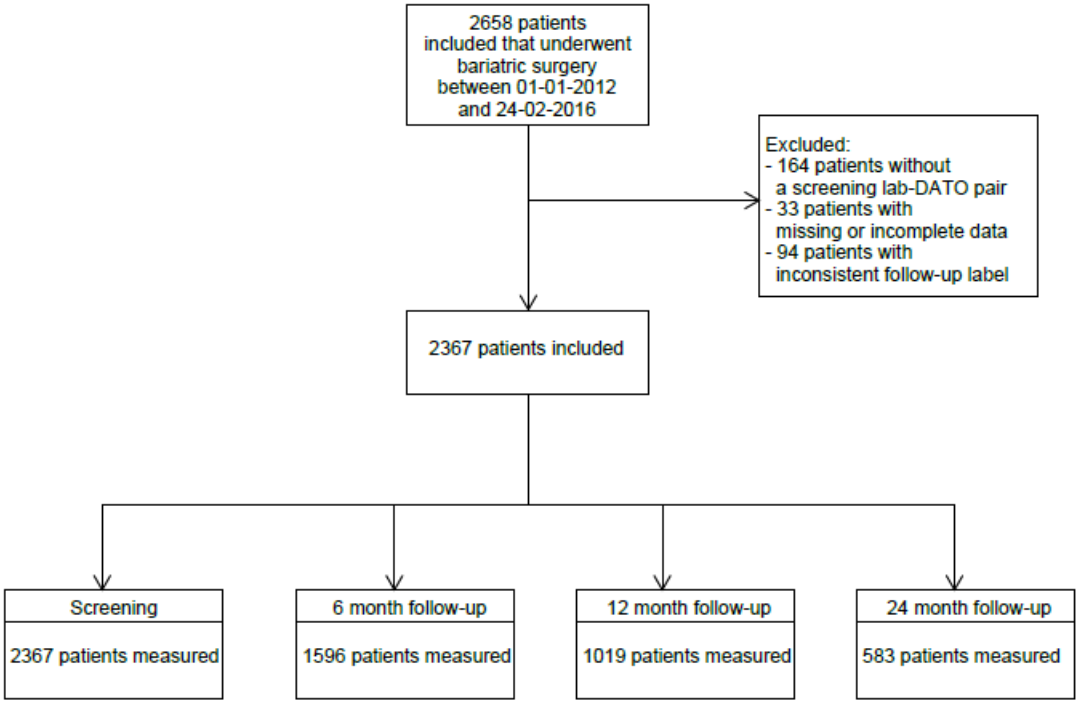


Figure C.2: All patients present after pre-processing [24]

for discrimination, calibration and overall performance. Lastly the model was visualized with a nomogram for further use.

C.4 Proposal Comparisons

A proposal was created after reading the introduction, background and material. The original research was also discussed and a summary was given. The new proposal should now be compared with the original research to find out when they match and when they do not. Several topics should be discussed that differ greatly between the original and new proposal. These differences should be addressed to create a final proposal for a new research. The first difference was about data omission. Many aspects of the original data set were debatable whether they should be removed or not. A second difference was the final scoring system. The newly proposed one was vastly different from the original one.

C.4.1 Data Omission

In preprocessing (subsection C.3.1) a big number of data omissions was done with various reasons. While many of them were regarded as valid, a number of them could be seen as being a bit too rigorous. These questionable ones were discussed whether they should be left out.

1. **Lab set without a close DATO set** Several lab sets were removed because no DATO set was close enough. This seemed logical as they should match for complete results. However the DATO set had many variables that are not subject to much change over time. If a lab set was squeezed in between DATO sets that were very similar, an estimate for the intermediate DATO set at that time could easily be made.

2. **Predefined measuring moments** The DATO and lab sets were all measured on predefined moments. There were other times as well, though, were measurements took place. On those other times data was omitted. These omissions seemed too rigorous. These data sets were maybe not useful enough to show differences on those measuring moments, however these sets could be very helpful to create a model. These questionable omissions included pairs that were double at one moment and moments at 36 or 48 months.

3. **Markers only in pre-surgery measurements** Five of the biomarkers were only measured before surgery. The reason for this was not stated, however these markers were removed as about half of the data sets had these. When almost 50% of the data sets had these markers, there was enough available to check these variables on their relation with the co-morbidities. If all of them showed no significant difference in presence or absence of the co-morbidities, then they were not useful. If they would show a significant difference, they might be a good marker for further co-morbidity detection. Disregarding them only because half of the sets did not have the value seemed questionable.

4. **LDL-cholesterol** This marker from the lipid panel was removed due to its possible derivation from the same panel markers. The omission was discussed quite heavily in the thesis. Even though it was derived from other markers, the derivation was done in a very specific way. Maybe a description for one of the co-morbidities would benefit greatly of this specific derivation. Omitting LDL-cholesterol was less questionable than the other cases, however at least some tests could be done to check the significance.

C.4.2 Scoring System

At the project start a goal was made about creating a severity scoring system for co-morbidities. To achieve that from categorical data, as little data reduction as possible should be done to keep the score quality as high as possible. The proposed ordinal system however actually reduces information. First there is no distinction between two or three co-morbidities. The type of co-morbidity is removed, too, as well as its progress after surgery. This is a very counter intuitive way to correctly model a scoring system when these three decisions all could contribute to a more efficient scale.

A second remark about the scoring system is about the final product. The linear predictor score was not really an answer to the original goal. The goal was to create a severity score, whereas the linear predictor score in the nomogram is originated from its probability counterpart and not really showing severity. As discussed earlier, it was hard to create a severity score from categorical labelling, so maybe that goal might be too ambitious.

For this research it seemed better to keep the three co-morbidities separate as long as possible. For every co-morbidity separately a score could be made which in the end could be joined together. This score would be best based on a probability at the start, so how much chance there was to have a certain co-morbidity. After creating probability models for all of them, they could be combined in a joint probability model. While doing this, obviously collinearity needed to be taken into account. As recently discussed, a probability model was not according to the goal. A score was desired and not a probability. Since the data was not gathered in a way to find a score with the measured tasks, this was much harder. After possibly finding good results for a probability model, a next step could be to translate that model in a scoring system.

C.4.3 Modelling Tools

About every part of the original research was done in R. As discussed previously, especially the variable significance could be done efficiently in R, as it was one of the more advanced tools to do statistical analysis with. For the modelling part however there were more advanced programs or frameworks that could be used. One of the reasons to choose R was to create a "white-box" answer, that every step of the process could be explained. The disadvantage in choosing a white-box model, however, was that the result could be worse than for a more traditional machine learning approach. Those approaches could lead to a better result and were therefore definitely worth to investigate. In R however, machine learning options were extremely limited if even available. Other programs or frameworks could facilitate that better than R, such as *Python* with the Scikit-learn package, Theano or TensorFlow This could also facilitate a better final visualization result for the scoring system than the nomogram, which was not very user-friendly and straightforward.

C.5 Final Proposal

The final proposal combined the first one and the original research for optimization. Questions from the first proposal were re-used, with some modifications and additions. The hypotheses were changed accordingly. The methods for answering the research questions were similar to the ones in the first proposal, however, they were extended after more information from the original research was added.

C.5.1 Research Questions

The main research question stayed the same as the one from the first proposal. This was the main goal of the research and therefore was a good all-encompassing question for it.

> *Can a severity score for co-morbidities present in bariatric patients based on biomarkers be developed?*

The first sub-question also stayed mostly the same. It was a bit unclear that there was relevant data in the DATO set. Therefore instead of saying biomarkers, the term "available variables" would be used. The word available was added due to not immediately omitting some of the variables, so their relation to co-morbidities would still investigated.

1. *Which relations are present between co-morbidities and the available variables?*

One thing the original research correctly investigated that was not thought of, was collinearity. This was already investigated for most variables, however the ones then removed might still have collinearity characteristics. Therefore a new sub-question was added to find those.

2. *Which collinearity relations are present within the available variables?*

The other sub-question was about adding a score to the co-morbidities originated from variables. This question was quite vague and with the help of the original research it was split up in three separate questions. The first one was about modelling the co-morbidities with variables.

3. *How are the co-morbidities best modelled with the available variables?*

The second part of the split sub-question was about the scoring part. Eventually a score should be given to the co-morbidities, which must be investigated thoroughly.

4. *How can scores best be added to the available co-morbidity models?*

The last part was about combining the co-morbidities together. Until now they were regarded separately, but the eventual result should have these scores combined. Therefore the third one of the split sub-question was about combining those.

5. *How can the co-morbidity models best be combined?*

This main question and the five sub-question should cover all aspects in this research. If all questions were answered one by one, a scoring model for the co-morbidities should be available for bariatric patients.

C.5.2 Hypotheses

Since three sub-questions are added. The new hypotheses should be added, as well. They will be addressed in order of appearance and afterwards a general hypothesis will be given for the main research question.

The first sub-question has the same hypotheses as in the earlier proposal. Strong relations should be found in the markers that the co-morbidities are diagnosed from and weak relations

should be found in the ones that are linked in literature. On top of that, since the original research answered this question partially, too, the following variables should show strong relations: triglycerides, HbA1c, CKD.EPI, age.at.surgery, HDL, cholesterol0-HDL-ratio, albumin, period, potassium, period * BMI, cholesterol and urea. All of these showed the best correlation.

The second sub-question has also been partially answered in the original research. Therefore strong relations should be found between: hematocrit and hemoglobin, MCV and MCH, ALAT and ASAT, HbA1c and glucose, hemoglobine and gender, erythrocytes and hematocrit. Aside from these mostly the omitted variables should show relations within their own panel. Insulin for example is expected to show a relation with glucose and C-peptide.

The third sub-question is partially answered in a specific direction by the original research. Logistic regression was extensively investigated, however other machine learning aspects were not used. The hypothesis is that most likely several machine learning methods should be sufficient to answer this question. With those machine learning methods a model can be derived for the co-morbidities to be found out of the available variables.

The fourth question is the also hardest. How can a score be added? The original research started first with an ordinal score, which is debatable for the information thrown away. When thinking ahead, the best idea for now is to start with a probability score. These probabilities usually also show the severity, as having a higher probability for a co-morbidity, the severity of it usually is bigger, too. After having a probability score further insights can be used to create a better score than probability. An example of such a score can be this:

> **Score proposal** At the start a score interval of 0% to 100% will be chosen, with 0%-50% being not having the disease and 50%-100% having it, with different severities. Binary cases will then be given 25% and 75% as absent and present co-morbidity respectively, knowing those as the values they should be closest to. The ordinal cases that showed improvement or deterioration will be given a score of 50%.

The last sub-question is bigger than only adding them together. Since the co-morbidities are also related to each other, these correlations have to be taken into account when creating a score. On the other side, when two co-morbidities are both present in a patient, they also have a bigger chance to be more severe, as they strengthen each other. Different additions or combinations can be tested to combine them.

The hypotheses of most sub-questions are fairly straightforward and with the original research are known to give results already. The aim is however to create better answers than the already available ones. For the first two questions this is definitely possible, the last three depend whether the results will be improving the known ones already, mainly because it is a new area for researching. Still, though, as there is already one present, the main research question can be answered with a yes.

C.5.3 Methods

This time the methods is changed a bit from the first proposal, as it would be better to line them up with the original research and show differences between those. So first preprocessing is started as well as everything related to that. Secondly the modelling part will be discussed, together with the sub-questions that are answered in here.

In preprocessing almost every thing will be done the same as in the original research. Most differences will be in not removing certain parts of the data set. This means that more than one data set can be made for the different instances. All differences will be discussed.

1. **All paired DATO and lab set** All data that is available after pairing the DATO and lab set will be put in a data set. There will not be any distinction between data on predefined moments and not, because for modelling those do not matter at all. In case more results should be generated only on those predefined moments, a separate attribute set will be made to extract those.

2. **Variables omission** Not all variables will be omitted when a significant number of them is available. Relations between those variables and with the co-morbidities will be tested. When finding out that there are significant relations between them, possible further research will be done on both a set with and without them. Variables that fall into this category are the five that are only measured pre-surgery and LDL.

Several steps should be taken to achieve the final goal. First the variables needs to be compared with the co-morbidities (sub-question 1) and themselves (sub-question 2). Next a model has to be found (sub-question 3) for all co-morbidities as well as a score added to the model (sub-question 4). Lastly the three models need to be combined (sub-question 5) to create a final answer to the main research question.

To answer the first two sub-questions, statistical analysis must be done. Correlation matrices must be made to see whether two variables or one and a co-morbidity correlate significantly. Also the variables must be checked on their own population, whether they have a high variance and follow a statistical distribution. In case of high correlations between variables, a joint variable should be created if possible. At last, also combinations of variables should be created and tested for relations with co-morbidities. This all can efficiently be done in *Python* with the *SciPy* package.

A co-morbidity can be modelled using different approaches in machine learning. Several classification techniques are available such as support vector machines (SVM), nearest neighbour (NN), stochastic gradient descent (SGD). Research must be done which is most applicable and testing must be done which gives the best results. This is repeated three times for every co-morbidity. Within *Python* the Scikit-learn package has the basics to create proper results and if needed, programs such as Theano or TensorFlow can be used within *Python* for more extensive research. For testing, a similar approach as in the original result can be used, called 10 fold cross-validation. The final model and the visualisation depends on the type of machine learning used, however graphs that show clusters of morbidities for several variables seems a good way to approach that.

Scoring addition in machine learning approaches is a challenge. At first tests can be done with changing the methods from classification approach to their regression counterpart, for example from SVC (Support Vector Classification) to SVR (-Regression). This can have major for now unknown implications. As discussed earlier (subsection C.5.3) setting up a score interval should be thought of before and also able to be implemented. The proposed score can be used as a base and be changed if advantageous.

The last challenge is to create a model that combines all three of the co-morbidities. To start this, all three separate models can be used to create the combination. If this does not give the desired results and makes scores much worse than initially thought, a similar ordinal system as used in the original approach can be used. Then sub-questions three and four are repeated only with co-morbidities combined, removing details from each different one.

Appendix D

Case Study 2 - Skin Diseases

D.1 Introduction

To find out what techniques and guidelines are useful to put in a framework for biomedical engineers, a case study is done with an example biomedical dataset. This example dataset is a based on gene expression of skin diseases [25, 26, 27, 28, 29, 30, 31, 32] and is a microarray dataset. Two skin diseases were intended to be examined with this dataset, psoriasis and atopic dermatitis. The expression for a total of 54675 genes were measured for skin disease patients on skin affected by the disease (lesional skin) and skin not affected by the disease (non-lesional skin). Further data from healthy subjects was also acquired. Nine datasets were available, six for psoriasis [25, 26, 27, 28, 29] and three for atopic dermatitis [30, 31, 32]. The number of tested skin biopies ranged from 28 to 180 per dataset whereas the number of measured genes is the same for every set, namely 54675.

In this project, a basic analysis was done on the available skin disease datasets, as a case study for creating a framework for biomedical engineers. First, a background is given on the dataset. Secondly, methods to extract information from the data are explained, followed by their results. Finally, the results are discussed and useful aspects for a biomedical engineering framework about the basic analysis are concluded from the case study.

D.2 Skin Diseases Datasets

Skin diseases can have a major impact in someone's life. Whereas skin diseases are not as life threatening as diseases such as cancer, Alzheimer and AIDS, they can lower quality of life significantly. When looking at the health-related quality of life (HRQL), patients with psoriasis show the same problems as patients with other major chronic health conditions [190]. Patients with psoriasis or atopic dermatitis suffer from severe itching and pains. Further insights into the skin diseases can help alleviate their unwanted side-effects and help improve the patients' quality of life [191].

Information on both of these skin diseases can be found in nine datasets stored on the NCBI database [192]. The datasets comprise microarray data extracted biopsies of psoriasis patients, both from their lesional and non-lesional skin. In several experiments this skin is taken from the same patient. Also some skin is taken from patients not suffering from any of these two diseases. six datasets gathered data from Psoriasis patients and three from atopic dermatitis patients. These datasets consist of a total number of 54675 genes, the features of the dataset. The range of acquired samples varies among datasets from 28 tot 180. Also, since every dataset was created by different people, some minor differences can be present in them as well (Table D.1), due to different measurement equipment.

The nine datasets are rich with information. The dimensionality is very high and if combined the datasets also have a decent number of samples, from a data mining perspective. Whereas

Table D.1: Details of the nine skin disease datasets. The number of samples and genes has been given, as well as remarks of the skin types.

Disease	Dataset name	Number of samples	Genes	Sample size remarks
Psoriasis	**GSE13355** [25]	180	54675	Three skin types: - NN (normal, 64 samples) - PN (non-lesional, 58 samples) - PP (lesional, 58 samples)
	GSE30999 [26]	170	54675	- No normal patients - Non-lesional (85 samples) - Lesional (85 samples)
	GSE34248 [27]	28	54675	- No normal patients - Non-lesional (14 samples) - Lesional (14 samples)
	GSE41662 [27]	48	54675	- No normal patients - Non-lesional (24 samples) - Lesional (24 samples)
	GSE78097 [28]	33	54675	Different types of skin samples: - Normal (6 samples) - Mild Psoriasis (14 samples) - Severe Psoriasis (13 samples)
	GSE14905 [29]	82	54675	- Normal skin (21 samples), - Non-lesional skin (28 samples) - Lesional skin (33 samples)
Atopic Dermatitis	**GSE32924** [30]	33	54675	- Normal skin (8 samples) - Non-lesional skin (12 samples) - Lesional skin (13 samples)
	GSE27887 [31]	35	54675	Different type of skin samples, pre and post treatment of skin: - Pre non-lesional (8 samples) - Post non-lesional (9 samples) - Pre lesional (9 samples) - Post lesional (9 samples)
	GSE36842 [32]	39	54675	Also difference between acute and chronic dermatitis. - Normal (15 samples) - Non-lesional (8 samples) - Acute lesional (8 samples) - Chronic lesional (8 samples)

biomedical datasets often contain several processing challenges, this data also has some of them. Here, three of these challenges are discussed.

- At first the challenge of handling nine different datasets is essential. Even though the genes were chosen according to the Affymetrix Human Genome [193], the layouts are not identical. These difference originate from the intended research goals and the data availability. It is not possible to just concatenate samples without some form of preprocessing. Only the parts that are the same all over the datasets must be taken and all other parts omitted.
- A second challenge can be found in the high number of genes. There were 54675 genes measured. The number of genes that are significantly involved with the skin diseases is however expected to be much lower than 54675. Many genes most likely are redundant and can be removed during preprocessing, a valuable and complex step in biomedical data mining.

- The third challenge is about data volume. The number of samples differs from 28 to 180, all of them being a very low number compared with the number of genes. From a data mining perspective, this number of samples is hard to gather information from. This can create problems, mainly during machine learning, with such a low training and test set. Several cases will arise where all training and test set agree with the algorithm, whereas other samples from the sample space would not.

D.2.1 Additional Data

The features all correspond to the same genes for all of these nine different datasets. The NCBI database [192] also provides separate data containing substantial information for every gene. This data can be used to find links between several processes and their corresponding genes. This information includes for example:

- gene ID
- commonly known gene name
- commonly known gene abbreviation
- originated gene database
- processes the gene is involved with (ranges from possibly zero to many processes)
- cellular locations the gene has a relation to (ranges from possibly zero to many locations)
- molecular reactions the gene is involved with (ranges from possibly zero to many reactions)

D.3 Methods

Three different aspects were investigated with the dataset. These three aspects were discussed in three different sections. The three sections are:

- *Feature reduction* At first several techniques were used to reduce the high number of genes.
- *Clustering* Secondly, after the feature reduction was done, the genes were clustered,.
- *Psoriasis vs atopic dermatitis.* Thirdly comparisons were made between the datasets of psoriasis and atopic dermatitis. This also was done after feature reduction and done to find out whether genes had been over- or under expressed in both of them.

A previous project[1] found out that healthy and non-lesional skin do not show many differences and after some testing, the same conclusion was reached in the current study. Aside from this, the difference between lesional and non-lesional skin was the most important for skin diseases to compute, as that difference showed which genes are over- or under expressed in lesional skin. Therefore the main focus of the project was showing the difference between non-lesional and lesional skin in terms of genes.

For the computation of feature reduction, the largest possible dataset was created with only non-lesional and lesional skin samples. The biggest dataset that could be created was with the Psoriasis sets that had both non-lesional and lesional skin (table D.1), good for a total of 423 samples. When comparing psoriasis and atopic dermatitis, all suitable samples of Atopic Dermatitis were collected for a total of 58 samples. In gene reduction and clustering only the psoriasis dataset was used, due to its bigger sample size. When comparing psoriasis and atopic dermatitis, both were used.

[1] BEP Project - *Manouk Groels*

D.3.1 Feature Reduction

Since the number of features in the data was 54675 (the number of genes), a significant feature reduction was needed before any meaningful computations could be done. Therefore two different ways of feature reduction were explored. The first one was doing a simple t-test to find all genes that were significantly different. The second one was testing correlation between all genes, also known as multicollinearity testing. The complete layout of the feature reduction was visualized for understanding (Figure D.1).

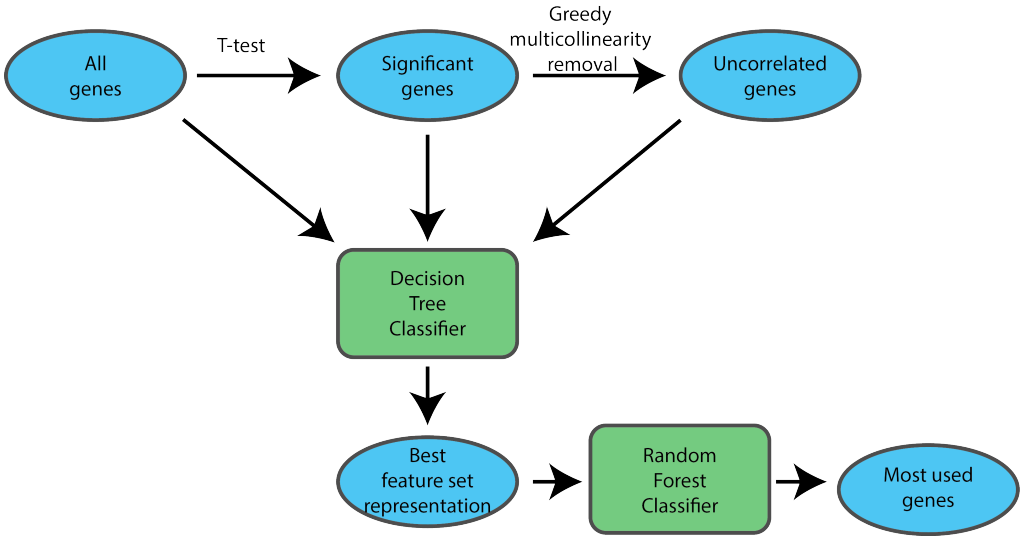


Figure D.1: A flowchart layout of the feature reduction. The complete data set first was reduced by using a t-test and next by the greedy clustering algorithm. The three feature set representations (all genes, significant genes and uncorrelated genes) were all three tested by a decision tree classifier and the one with the best results was chosen. At last the best feature set representation was classified with a random forest classifier and the genes that were most used in the final random forest were selected.

For using a t-test, the *SciPy* package was used. Three different t-test possibilities are available, one for paired data, one for data with equal unknown variance and one for data with unequal unknown variance[2]. Two of these t-tests were used, the paired t-test and the t-test for unequal unknown variance. Almost all datasets (GSE13355, GSE30999, GSE34248 and GSE41662) were paired, in those cases the paired t-test was used (Equation D.1) and in the remaining set (GSE14905) plus the combined set (all datasets together) the unequal variance t-test were used (Equation D.2). In these equations $\bar{y}$ was the mean of a distribution, s the standard deviation, n the number of samples, $\bar{d}$ the average value for difference in paired samples and t was the normalized version of the gene to be checked ($t = \frac{y-\bar{y}}{s/\sqrt{n}}$).

$$\mathbb{P}(t > |t_{calc}|) \text{ with } t_{calc} = \frac{\bar{d}}{s_{\bar{d}}} \tag{D.1}$$

$$\mathbb{P}(t > |t_{calc}|) \text{ with } t_{calc} = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \tag{D.2}$$

A further reduction could be done by removing multicollinearity. Multicollinearity means that multiple genes are highly correlated with each other. Usually this would be computed by calculating the correlation coefficient between all genes. This coefficient however was computationally quite heavy to compute and therefore an alternative approach was used (Algorithm 19

[2]Biomedical Data Analysis - *Tim Beishuizen*

and Figure D.2). A greedy hands on approach computed correlation between genes and removed multicollinearity. For this greedy hands on approach a list of genes was used, ordered on the significant difference of gene expression between non-lesional and lesional skin data, computed by the t-test.

Algorithm 19 The greedy multicollinearity removal algorithm

```
1:  procedure GreedyMulticollinearityRemoval(SortedGenes)
2:      UncorrelatedGenes ← emptyList
3:      correlationValue ← 0
4:      for newGene in SortedGenes do
5:          for oldGene in UncorrelatedGenes do
6:              if correlation(oldGene, newGene) > correlationValue then
7:                  correlationValue ← correlation(oldGene, newGene)
8:              end if
9:          end for
10:         if correlationValue < 0.7 then
11:             UncorrelatedGenes ← UncorrelatedGenes ∪ newGene
12:         end if
13:         correlation ← 0
14:     end for
15:     return UncorrelatedGenes
16: end procedure
```

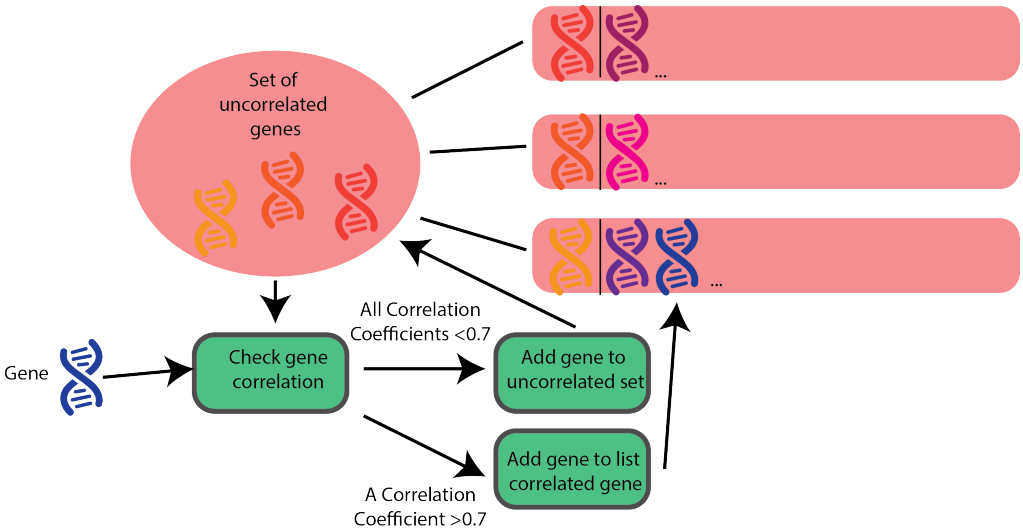
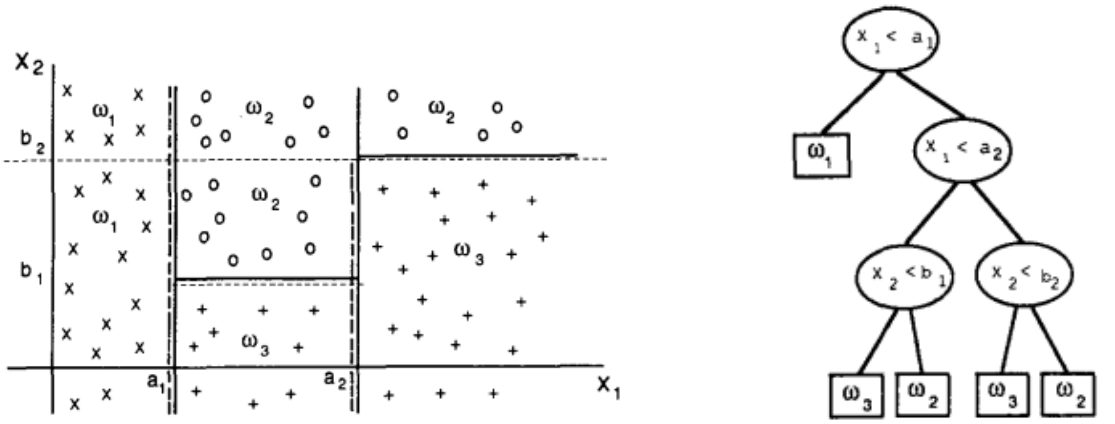


Figure D.2: A flowchart layout how the gene multicollinearity removal algorithm worked. The correlation coefficient is iteratively computed between a new gene and the uncorrelated set of genes. If all correlation coefficients are lower than 0.7, the new gene did not correlate enough with the set of genes and therefore is added to the uncorrelated set of genes. If it had a correlation of higher than 0.7 for another gene, it was added to that the list for the gene it is correlated with.

After removing the insignificant genes and multicollinearity, a decision tree classifier (Figure D.3) from the *scikit-learn* package was used to find out whether removing unusable genes and multicollinearity actually improved the possibility to better classify lesional skin. All three gene sets (all genes, significant genes and greedy clustered genes) were used for decision tree classification and split in 80% training and 20% test set. A cross validation was done with 100 different subsets for this classification, dividing the 80% training set into 100 subsets that all were used as a

validation set once. With the cross validation, both a validation score and a test score were made to evaluate the performance with the three gene datasets.



(a) A two dimensional sample space divided in three regions.

(b) The corresponding decision tree

Figure D.3: An example of how a decision tree divides a sample space in different regions. A sample space (Figure D.3a) with features (x_1 and x_2) is divided in regions (ω_1, ω_1 and ω_1). A decision tree (Figure D.3b) shows how this division is done using thresholds for the features. Every node in the tree corresponds to dividing the sample space in different regions using a threshold for a feature and every leaf corresponds to a region. The classifier tries to divide the sample space in regions finding the best feature threshold combinations, until boundary conditions are met.

After testing with a decision tree classifier a random forest classifier was used. A random forest classifier creates not one, but multiple decision tree classifiers on subsets of the complete sample space and averages the outcome, for this case 1000 decision trees were created. The input of the random forest classifier was the best gene dataset of the three tested gene sets (all genes, significant genes, uncorrelated genes), according to the decision tree classifier. Aside from the input, the decision trees were only allowed to have a depth of two, to avoid over fitting Genes that were used multiple times as a splitting criterium in the created decision trees were extracted from the random forest classifiers. These genes occurred most frequently as the best splitting criterium and therefore had the best relation with the skin disease. The six genes that occurred most were selected.

D.3.2 Clustering

Given the high number of genes even after feature reduction, clustering is a good way to find whether genes show similar behaviour. Two different ways of clustering were explored. The first way is based on the values of the genes, clustering genes that showed the same behaviour. A second way is clustering them based on biomedical relations between the genes. After exploring these two ways, the interesting results were combined. A complete layout of every thing done during this clustering was visualized (Figure D.4).

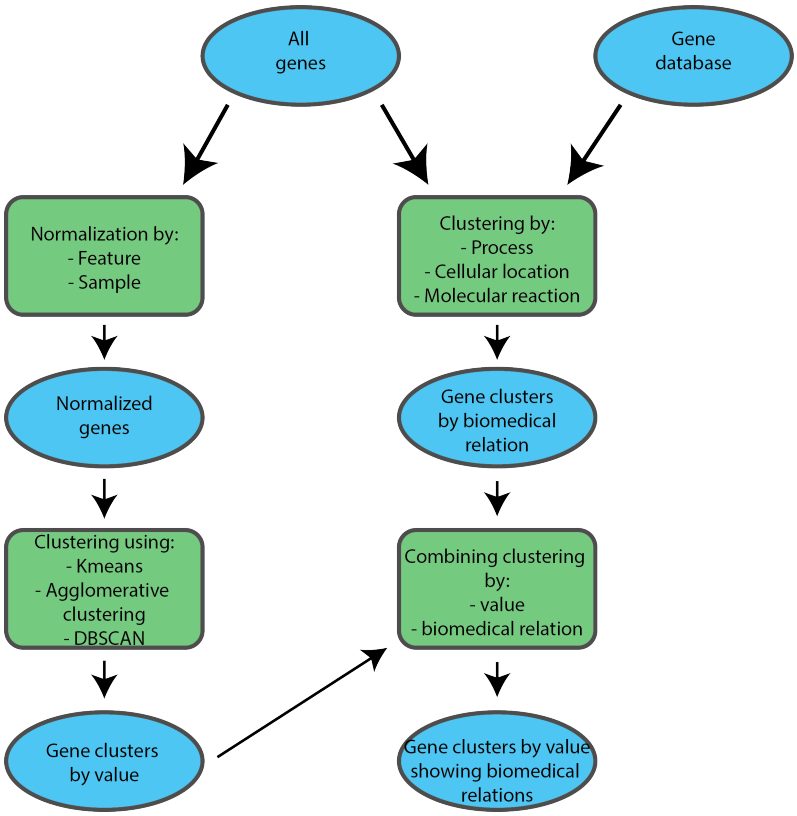
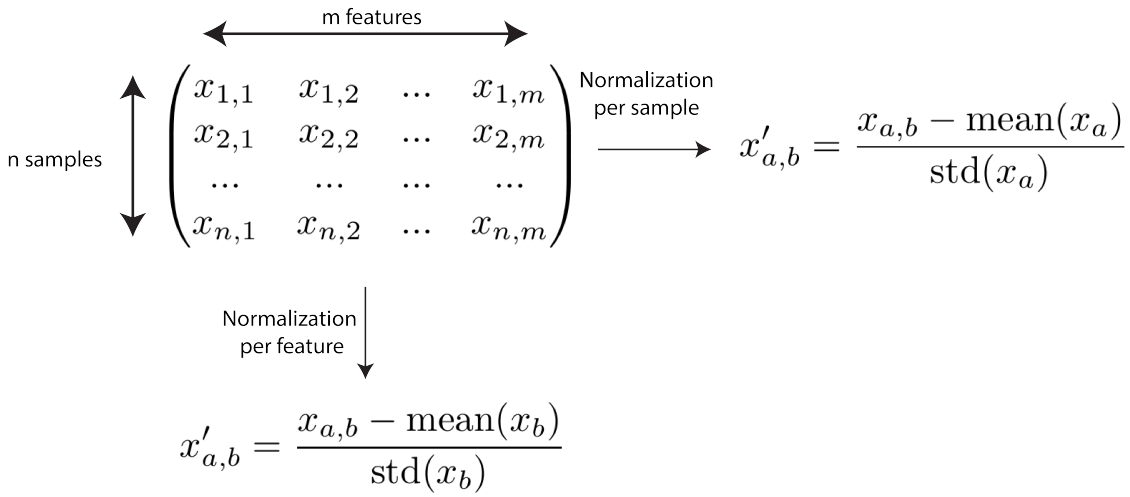


Figure D.4: A flowchart layout how clustering was done. Two types of clustering were used, by biomedical relation and by value. For the clustering by value, first a normalization was done. After both types of clustering were done, they were combined and biomedical relations within the clusters were shown.

Before clustering was done, the values were normalized. This normalization was done to remove high differences in especially variation. Two types of normalization were done (Figure D.5). At first normalization per genes was done, so every gene was treated equally. Relatively big differences in gene expression were observed that way. The normalization per sample compared the gene expression between genes within a sample. This gave skin cell specific genes higher values than non-skin cell specific genes, because their expression should be higher overall. This way genes known to be more prevalent in skin cells had a higher chance to be noticed for the difference between lesional and non-lesional skin.



$$\begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & \dots & x_{2,m} \\ \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & \dots & x_{n,m} \end{pmatrix}$$

$$x'_{a,b} = \frac{x_{a,b} - \text{mean}(x_a)}{\text{std}(x_a)}$$

$$x'_{a,b} = \frac{x_{a,b} - \text{mean}(x_b)}{\text{std}(x_b)}$$

Figure D.5: An explanation of the difference between normalization per feature and normalization per sample. $x_{a,b}$ is value x for sample a and feature b, mean() gives the mean of the values given (either a sample row or feature column) and std() gives the standard deviation for the values given. For both the formula of how to compute the normalized value $x'_{a,b}$ is given.

The first type of clustering was done by using basic clustering methods. Three different types of clustering were used (Table D.2): K-means, Agglomerative and DBSCAN, all three of them methods in *scikit-learn*. For both K-means and agglomerative clustering a variation of cluster numbers is chosen to find the best possible selection.

Table D.2: The three different clustering types for clustering by value.

Clustering type	**Explanation and parameters (if applicable)**
K-means	Assigns genes to clusters closest to cluster centres Changes cluster centres after new genes are assigned Tested number of clusters: 5, 8, 9 10, 11, 12, 15, 20.
Agglomerative (Hierarchical)	Adds genes closest to each other together in a cluster until only the desired number of clusters are left. Tested number of clusters: 5, 8, 9 10, 11, 12, 15, 20
DBSCAN	Clusters gene groups with high density together. Good with data with high number of features and low number of samples

Aside from clustering the genes by values, another way was clustering by biomedical relations. The additional dataset (Subsection D.2.1) links genes to three different biomedical components (Table D.3): processes, cellular locations and molecular reactions. Genes linked to the same biomedical relation means the genes are involved in that particular process and therefore can be clustered together. Since genes can be linked to multiple processes, locations and reactions, they were put in multiple clusters for all three.

Table D.3: The three biomedical relation types available in the database (subsection D.2.1) with a description and several examples.

Biomedical relation	**Explanation**	**Examples for gene:** *discoidin domain receptor tyrosine kinase 1*
Process	Biomedical processes the gene product is involved with	- regulation of cell growth - peptidyl-tyrosine autophosphorylation - lactation
Cellular location	The gene product is located in or part of certain cellular locations	- extracellular region - plasma membrane - extracellular vesicular exosome
Molecular reaction	A molecular reaction the gene product is involved with	- nucleotide binding - collagen binding - protein tyrosine kinase collagen receptor activity

Both types of clustering should give different results, as one of them only took into account the values and while the other focused on biomedical relations. A good way to find results was to combine both clustering types, to find out which value clusters were also involved in similar biomedical processes and vice versa. This was done in two different ways. The first way was done by ordering the biomedical relation dependent clusters by their difference between lesional and non-lesional gene expression. More difference between these expressions means it is regarded as showing a bigger difference between lesional and non-lesional skin. This difference was quantified by first taking the mean of all sample values for both lesional and non lesional skin for every gene. The process then averaged this mean of all difference values of the genes to a general difference between non-lesional and lesional skin.

The second cluster combination one was by selecting all value dependent clusters, so all clusters created by a clustering algorithm. For all of these clusters biomedical relations were investigated. To do that, for every gene the processes it is related to was computed. If clusters showed that multiple genes in that cluster were related to an interesting process, cellular location or molecular reaction, a higher coverage would be given (equation D.3).

$$\text{coverage}_{\text{process}} = \sum_{\text{gene} \in \text{cluster}} \mathbf{1}\{\text{gene process}\} \tag{D.3}$$

At last it is important how to visualize the clusters. Every gene has 423 different values, one for every sample. this however is hard to visualize. Therefore, the gene values were averaged after normalization in two different values, one value corresponding to lesional skin and one to non-lesional skin. Gene plots were made with every gene being a data point with the x-value being the non-lesional skin gene expression and for the y-value the average lesional skin gene expression.

D.3.3 Psoriasis Versus Atopic Dermatitis

A brief search for the difference between Psoriasis and Atopic Dermatitis is done. Since up- and down regulations for both of these diseases are more interesting, first feature reduction is done by only choosing the ones that showed a significant difference. The genes for both diseases are then compared and for all genes that are up- or down regulated for both diseases, all biomedical processes, cellular locations and molecular reactions are extracted and counted how many times they were present for both diseases. The five most occurring processes, cellular locations and molecular reactions were given, if interesting enough.

D.4 Results

The results are presented in the same order as the methods. First, the results for feature reduction are shown for the t-test, multicollinearity reduction as well as for the machine learning improvements. Then, the results for clustering are shown. Emphasis is put on combining clustering by value and clustering by biomedical relations. Finally, the similarities between Psoriasis and Atopic Dermatitis are shown to see if genes, processes, cellular locations and molecular reactions are involved in both diseases.

D.4.1 Feature Reduction

The t-test is used between non-lesional and lesional skin for every dataset separately and combined (Table D.4). A low p-value of $p = 0.001$ is chosen by trial and error. Interestingly enough when all datasets are combined, the number of significant genes is lower than when looking at the datasets separately. This can indicate that dataset specific noise is present in the datasets and heterogeneity actually improves noise reduction. After this feature reduction, further computation is done with the 1768 genes left.

Table D.4: The results of using the t-test for genes in all relevant psoriasis datasets separately and combined. A paired t-test is done when they are paired, otherwise an unknown variance unpaired t-test was done. The number of samples for both lesional and non-lesional skin are shown additionally.

Dataset	Samples Lesional Skin	Samples Non-Lesional Skin	Paired	All Genes	Significant Genes
GSE13355	58	58	Yes	54675	22106
GSE30999	85	85	Yes	54675	20836
GSE34248	14	14	Yes	54675	7824
GSE41662	24	24	Yes	54675	15672
GSE14905	33	28	No	54675	13355
Combined	214	209	No	54675	1768

Computing multicollinearity was done next. Using greedy multicollinearity removal yields 335 uncorrelated genes for the combined dataset, so for 335 genes the correlation coefficient is lower than 0.7 between all of them. The ability to find the difference between lesional and non lesional skin with a decision tree classifier shows that only using significant genes give the best results, whereas using the uncorrelated genes make both the validation and test score worse. (Table D.5).

Table D.5: The results of the decision tree classifier for different sets of genes: all genes, only the significant genes and the uncorrelated genes. The decision tree is cross validated by division in 100 different subsets and afterwards tested by a separate test set.

Gene set	Genes	Validation score	Test score
All genes	54675	0.584	0.575
Significant genes	1768	0.959	0.943
Uncorrelated genes	335	0.928	0.915

With a validation and test score of around 0.95, the significant gene set is used for a random forest classification consisting of 1000 decision trees. The genes that are used as splitting criteria for these decision trees are collected to find out whether the same genes are used multiple times as the best splitting criteria (Table D.6). While the number of multiple occurrences is not as high as expected, the six most occurring genes already show to have a relation with skin diseases.

Table D.6: The six most used genes as splitting criteria for the random forest classifier. The genes are also compared with literature.

Gene ID	Gene Title	Times used in random forest	Link with Psoriasis
NM_004262	transmembrane protease, serine 11D	45	Previously discovered relation with Psoriasis [26]
AI186548	keratin 77	33	Keratin is up-regulated in uncontrollable growth skin cells [26]
BF032500	MACRO domain containing 2	33	Previously discovered relation with Psoriasis [26]
NM_001062	transcobalamin I (vitamin B12 binding protein, R binder family)	31	Previously discovered relation with Psoriasis [26]
NM_005621	S100 calcium binding protein A12	31	Previously discovered relation with Psoriasis [26]
U19557	serpin peptidase inhibitor, clade B (ovalbumin), member 3 & 4	31	Previously discovered relation with Psoriasis [26]

D.4.2 Clustering

Only the 1768 significant genes found from feature reduction are used for clustering. The genes are clustered per process, cellular location and molecular reaction they were linked to. For example, if two genes are both related to a certain process, they are clustered together. Since a gene can be linked to multiple processes, it can also be present in multiple clusters. The sixteen highest scoring processes (Appendix D.7 and Figure D.6), cellular locations (Appendix D.7) and molecular reactions (Appendix D.7) are shown for normalization per gene and normalization per sample. The findings on the cellular locations and molecular reactions of the genes do not show any interesting results. The interesting results are found in several interesting processes, related to inflammatory response. All of these show up regulation in the sample and are plotted to show their gene locations (Figure D.6)

- acute inflammatory response
- response to interferon gamma
- chronic inflammatory response
- oxidative stress.

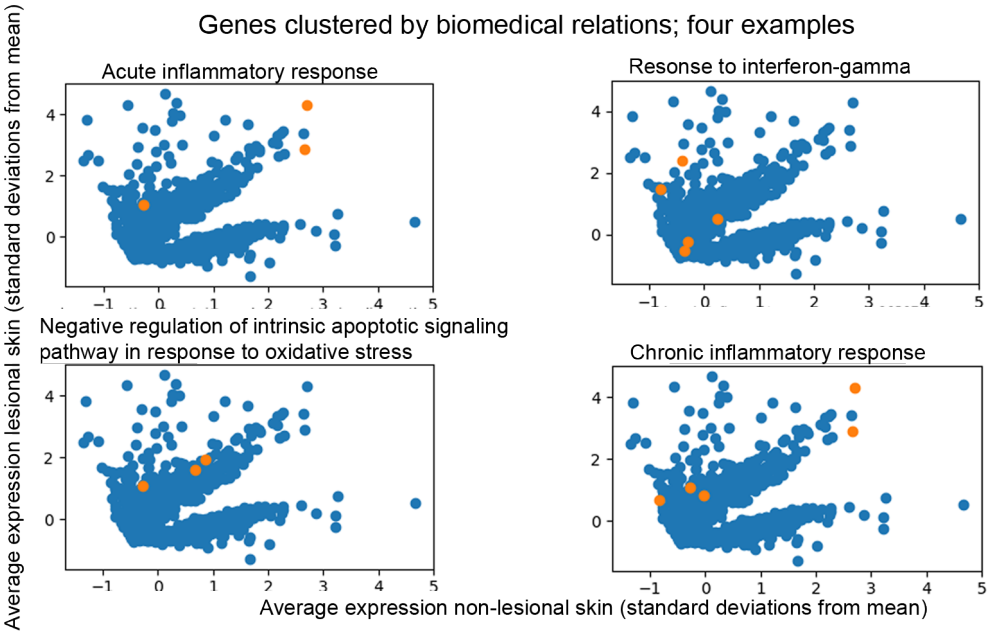


Figure D.6: Four examples of processes that show a high difference in gene expression between lesional and non-lesional data. The data is normalized per gene and orange points are genes that are related to the process, blue points are genes unrelated to the process

Comparing the value-based clusters and the biomedical relation clusters, agglomerative clustering with ten clusters gives the best results (Figure D.7). After linking the genes with biochemical processes, cluster 5 and cluster 8 show interesting links in results. Cluster 5 is linked to multiple processes typically occurring in skin cells, for example involving keratin and the epidermis. Cluster 8 is linked to multiple processes related to negative regulation of several enzymes, for example endopeptidase, peptidase and proteolysis. Other clusters are linked to less specific processes, e.g. transport and protein binding.

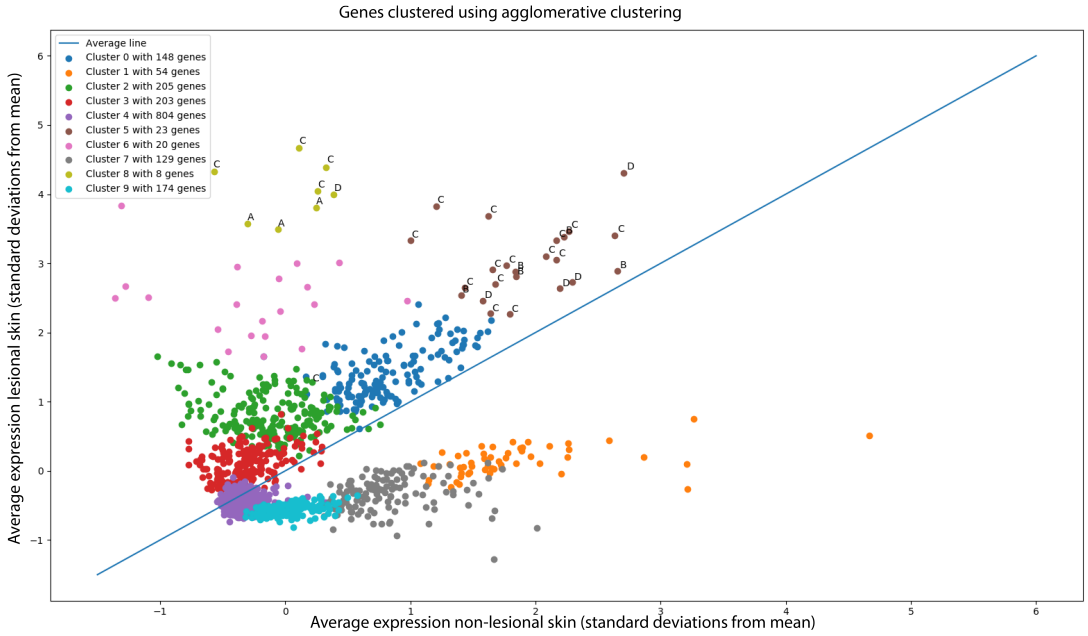


Figure D.7: 10 different clusters found by agglomerative clustering for the Psoriasis dataset. The data points are genes that show a significant difference in expression between lesional (y-axis) and non-lesional (x-axis) skin and normalized per sample. Cluster 5 and 8 show the most useful genes and its genes are also marked for Psoriasis relevance (Appendix D.8): Known Psoriasis marker (A), Known to be up-regulated for Psoriasis (B), Known to be up-regulated for uncontrollable growth (C) and unknown relation (D)

Since cluster 5 and 8 gave interesting results on a process level, literature was used to link them to psoriasis as much as possible (Appendix D.8). Most of the genes in these clusters were linked to psoriasis directly or indirectly (Figure D.7).

D.4.3 Psoriasis Versus Atopic Dermatitis

The significant genes for atopic dermatitis were compared with the significant genes of psoriasis (Table D.7). After matching both datasets to find out whether any genes were significant in both datasets, 96 genes were left. These genes were linked to processes, cellular locations and molecular relations to find biomedical relations that showed differences between lesional and non-lesional skin in both psoriasis and atopic dermatitis. Whereas the results from cellular locations and molecular reactions were inconclusive, the processes gave some more results. Five of these processes were related to biggest number of genes. All of these processes, however, seemed fundamental and therefore not useful for further investigation:

- metabolic process
- small molecule metabolic process
- regulation of transcription, DNA-templated
- signal transduction
- transcription, DNA-templated

Table D.7: The initial conditions of the Psoriasis and Atopic Dermatitis datasets

Dataset	Samples Lesional Skin	Samples Non-Lesional Skin	All Genes	Significant Genes
Psoriasis	214	209	54675	1768
Atopic Dermatitis	30	28	54675	516

D.5 Discussion

Three different aspects have been researched: feature reduction, gene clustering and comparing psoriasis and atopic Dermatitis. Some results do not give any additional insights, however others actually show possible future research topics.

Using a t-test seemed to be an efficient way to reduce the number of genes significantly without any loss in information. Especially for multiple and bigger datasets this method was effective for Psoriasis. The loss of information was present however when using the greedy correlation method, which was not used in further computations because of that information loss. After computing which genes would be used as splitting criteria, the six most occurring genes all but one either can be found in literature having a relation with Psoriasis. The remaining gene would be logical to be related to Psoriasis as well and therefore the significant gene set seems a good set to continue with.

Normalization is something that needed to be discussed, as well. Several mistakes were made by clustering which should be avoided in the future. At first the datasets were not normalized before joining them together. Some genes might behave differently per experiment which would be removed by this normalization. Future research should definitely do that. Also normalization per sample is not conventional, but gave good results for this project. This could have been a fortunate case in which this would work, but most likely next dataset it would not.

Initially, clustering by biomedical attribute and clustering by value was inconclusive. No genes seemed interesting enough by using only one type of clustering. However when combining both of them, the genes that were clustered together by both values and biomedical process gave two clusters of genes that showed to have a relation with both each other as well as with psoriasis. After labelling these genes multiple are already linked with psoriasis directly or indirectly, however some of the genes are worth investigating further.

D.6 Conclusion for Framework

The skin disease datasets were examined as a case study for which aspects are important for a biomedical framework. This case study lead to three different insights that would be helpful to add to a framework for general research: Global analysis, feature dimensionality reduction and database integration

Initially, a global analysis of the data set would be beneficial at the start of a project. The goal of this global analysis would mainly be to understand the how the data looks like. Means and variances should be shown for both samples as features and irregularities such as missing values and outliers should be made visible, so the one using the dataset, can understand how it works. Another important part for global analysis would be the multicollinearity, whether features or samples are very similar and therefore could cause problems for the user if not knowing these similarities. A global analysis would help understanding the data quicker and therefore create higher efficiency early on.

On the subject of feature dimensionality reduction some simple approaches have been used. Using a t-test showed quite a good result, whereas multicollinearity testing lacked a possibility to do that for high number of features. Some initial insights were obtained what to add and in which directions more research should be done to add more strategies for coping with feature dimensionality.

A last topic that would be useful for the framework is about integration of biomedical knowledge. Information about known biomedical relations between genes, metabolites or other biomedical substances is very valuable. This information would be very helpful if it would be possible to automatically use it during the research. An extension that could do that is therefore something to consider during the framework creation.

D.7 Appendix: Biomedical relation graphs

In this appendix the graphs for biomedical relations are shown. These graphs did not show any significant results, however do show how the method worked.

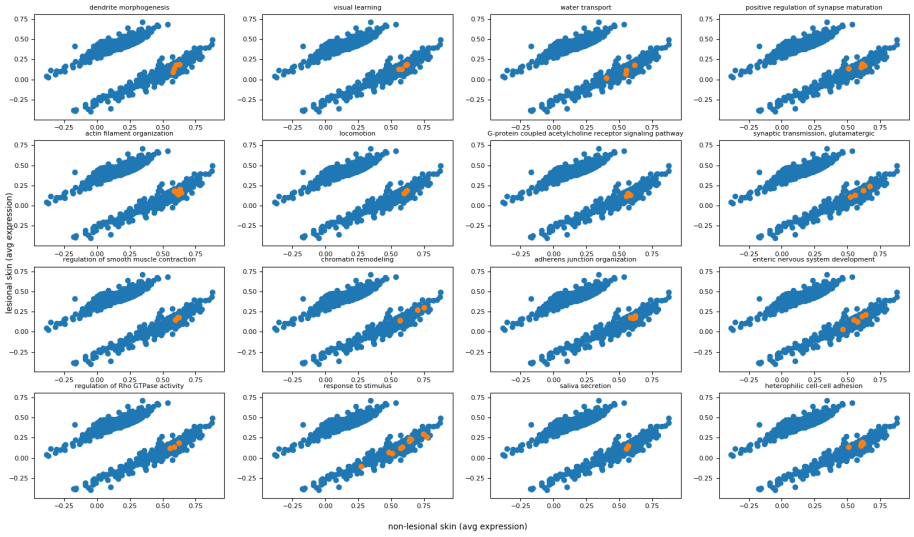


Figure D.8: The 16 processes that showed the highest difference in gene expression between lesional and non-lesional data. The data is normalized per feature and orange points are genes that are related to the process, Blue dots are genes unrelated to the process. The band between the two separate distributes is created by removal of the insignificant genes.

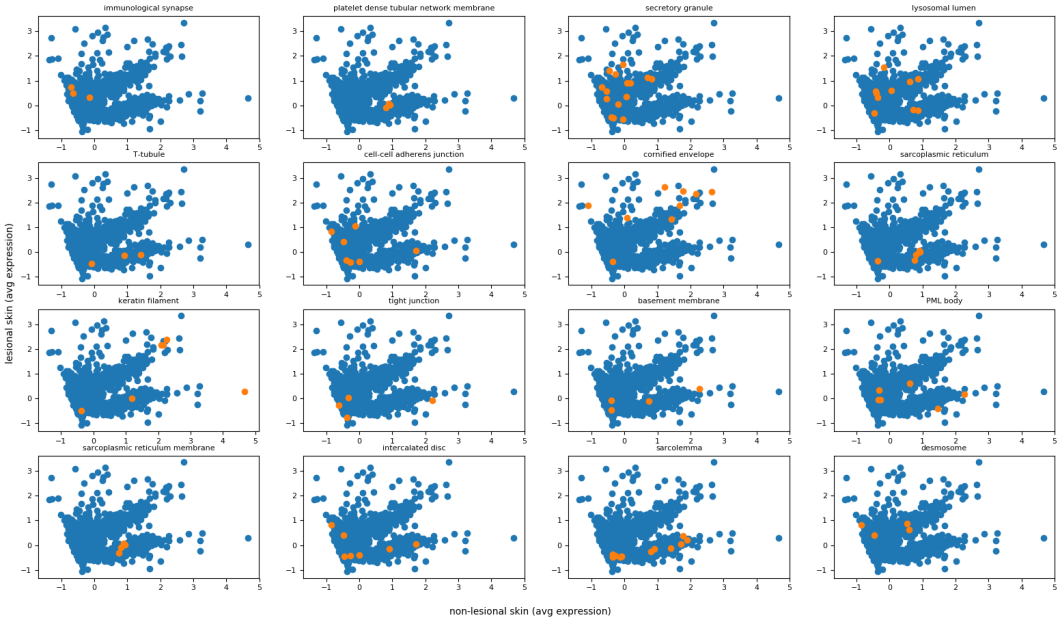


Figure D.9: The 16 cellular locations that showed the highest difference in gene expression between lesional and non-lesional data. The data is normalized per sample and orange points are genes that are related to the cellular location, Blue dots are genes unrelated to the cellular location.

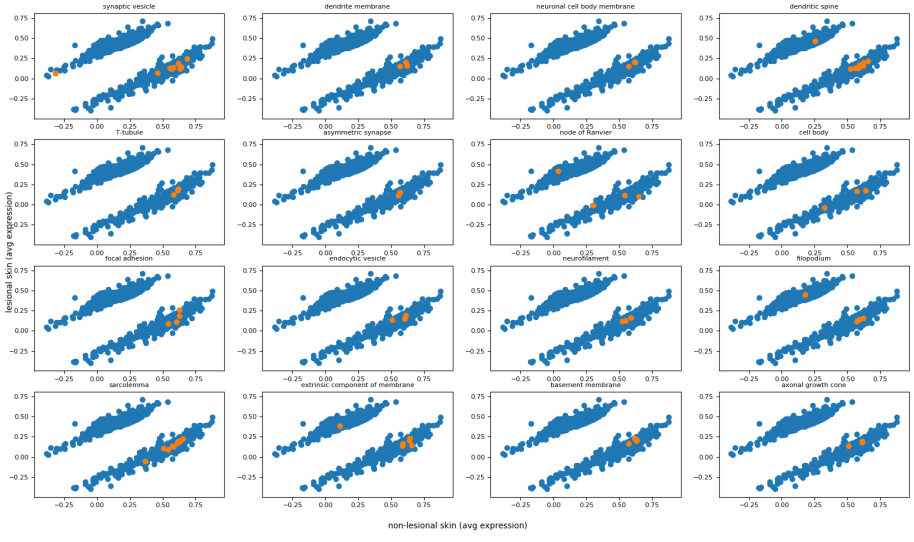


Figure D.10: The 16 cellular locations that showed the highest difference in gene expression between lesional and non-lesional data. The data is normalized per feature and orange points are genes that are related to the cellular location, Blue dots are genes unrelated to the cellular location. The band between the two separate distributes is created by removal of the insignificant genes.

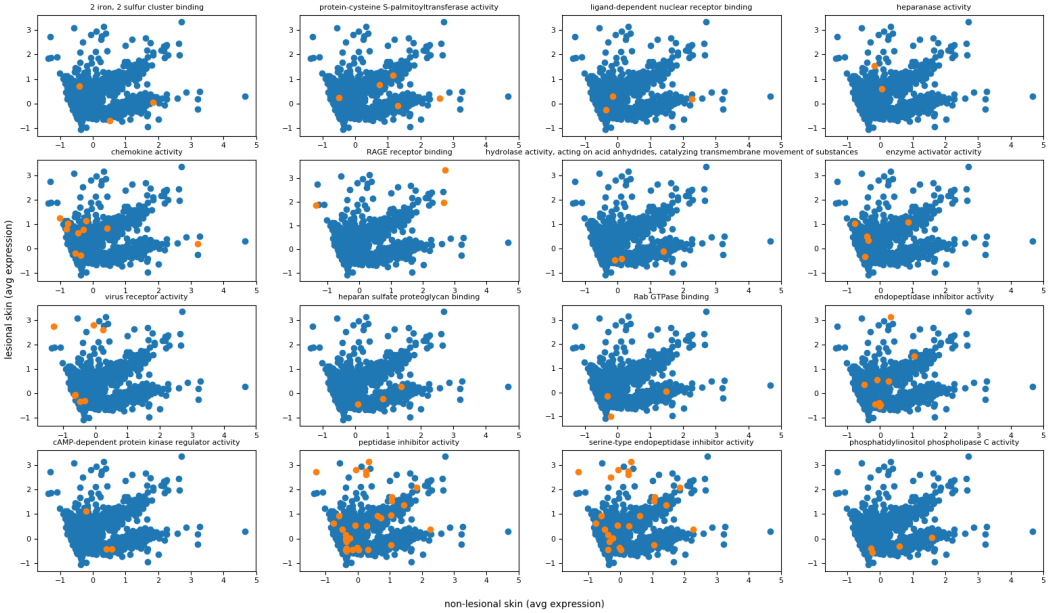


Figure D.11: The 16 molecular relations that showed the highest difference in gene expression between lesional and non-lesional data. The data is normalized per sample and orange points are genes that are related to the molecular relation, Blue dots are genes unrelated to the molecular relation.

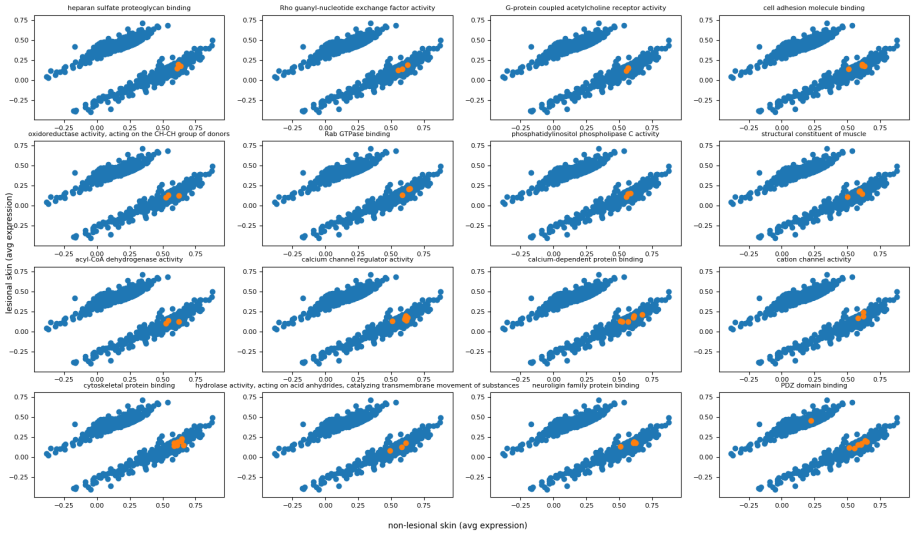


Figure D.12: The 16 molecular reactions that showed the highest difference in gene expression between lesional and non-lesional data. The data is normalized per feature and orange points are genes that are related to the molecular relation, Blue dots are genes unrelated to the molecular relation. The band between the two separate distributes is created by removal of the insignificant genes.

D.8 Appendix: Cluster Gene Specifications

In this appendix the genes from cluster 5 (Table D.8) and cluster 8 (Table D.9) are listed. Their commonly known name and a their possible link to psoriasis both are explained, if found in literature.

Table D.8: All genes in the highlighted cluster 5 with their link to psoriasis

Gene name	Gene description	Role in psoriasis
NM_006945	small proline-rich protein 2B	skin barrier development [194]
AF061812	keratin 16	markers of keratinocyte hyper proliferation [195]
BG327863	CD24 molecule	differentiation marker of epithelial cells [196]
BF575466	small proline-rich protein 3	skin barrier development [197]
AI923984	small proline-rich protein 1A	skin barrier development [198]
AB049591	cornifelin	Related to cornified envelope [199]
AF216693	interleukin 36 receptor antagonist	Antinflammatory cytokine//Aberrant interleukin-36Ra structure and function lead to unregulated secretion of inflammatory cytokines and generalized pustular psoriasis [200]
AB048288	late cornified envelope 3D	Late cornified envelope (LCE) genes, located in the epidermal differentiation complex on chromosome 1, encode a family of 18 proteins of unknown function, whose expression is largely restricted to epidermis [201]
AW238654	S100 calcium binding protein A8	proteins that attract leukocytes, up-regulated in psoriatic keratinocytes, also synthesized by neutrophils or myeloid dendritic cells that are recruited into psoriatic epidermis [194]
J00269	keratin 6A	hyper proliferation marker [202]
NM_002964	S100 calcium binding protein A8	proteins that attract leukocytes, up-regulated in psoriatic keratinocytes, also synthesized by neutrophils or myeloid dendritic cells that are recruited into psoriatic epidermis [194]
NM_005987	small proline-rich protein 1A	skin barrier development [198]
L42612	keratin 6B	hyper proliferation marker [202]
AL569511	keratin 6A /// keratin 6B /// keratin 6C	hyper proliferation marker [202]
AJ001698	serpin peptidase inhibitor clade B (ovalbumin) member 13	The encoded protein inhibits the activity of cathepsin K and is itself transcriptionally repressed by RUNX1. Known to be up-regulated in lesional skin. [26]
AI286239	No name	No connection found with psoriasis
M86849	gap junction protein beta 2 26kDa	Associated with psoriasis, regulates epidermal barrier and wound remodelling [203, 204]
NM_001878	cellular retinoic acid binding protein 2	Altered epidermal gene expression of CRABP II is not disease-specific and may reflect instead an altered state of epidermal differentiation and/or may be linked to the inflammation and cellular infiltration common various conditions. [205]
NM_000045	arginase 1	Participates in the regulation of iNOS activity by competing for the common substrate l-arginine, is highly overexpressed in the hyper proliferative psoriatic epidermis and is co-expressed with iNOS [206]
NM_005532	interferon, alpha-inducible protein 27	drives the development of psoriasis, marker of epithelial proliferation [207, 208]
NM_003125	small proline-rich protein 1B	skin barrier development [198]
NM_005130	fibroblast growth factor binding protein 1	FGFBP2 is down regulated, FGFBP1 not mentioned [26]
NM_002108	histidine ammonia-lyase	No connection found with psoriasis

Table D.9: All genes in the highlighted cluster 8 with their link to psoriasis

Gene name	**Gene description**	**Role in psoriasis**
AJ243672	S100 calcium binding protein A7A	IL-17 regulated inflammatory gene [209]
U19557	serpin peptidase inhibitor clade B (ovalbumin) member 3 /// serpin peptidase inhibitor, clade B (ovalbumin), member 4	Immune related gene and biomarker of psoriasis [210, 202]
NM_002965	S100 calcium binding protein A9	innate immune mediators, IL-17 regulated gene [209]
BC005224	serpin peptidase inhibitor clade B (ovalbumin) member 3	Immune related gene and biomarker of psoriasis [210, 202]
U19556	serpin peptidase inhibitor clade B (ovalbumin) member 3	Immune related gene and biomarker of psoriasis [210, 202]
NM_004942	defensin beta 4A /// defensin beta 4B	antibiotic peptide which is locally regulated by inflammation [211]
L10343	peptidase inhibitor 3 skin-derived	Upregulated in psoriasis, his gene encodes an elastase-specific inhibitor that functions as an antimicrobial peptide against Gram-positive and Gram-negative bacteria, and fungal pathogens. [212, 213]
NM_002638	peptidase inhibitor 3 skin-derived	Upregulated in psoriasis, his gene encodes an elastase-specific inhibitor that functions as an antimicrobial peptide against Gram-positive and Gram-negative bacteria, and fungal pathogens. [212, 213]

Appendix E

Feature Selection Results

E.1 Feature Selection Exploration Plots

The validation and test score for all combinations of dataset, ranking method and machine learning quality measure (Table 4.3) in figures (Figures E.1, E.2, E.3, E.4, E.5, E.6, E.7 and E.8).

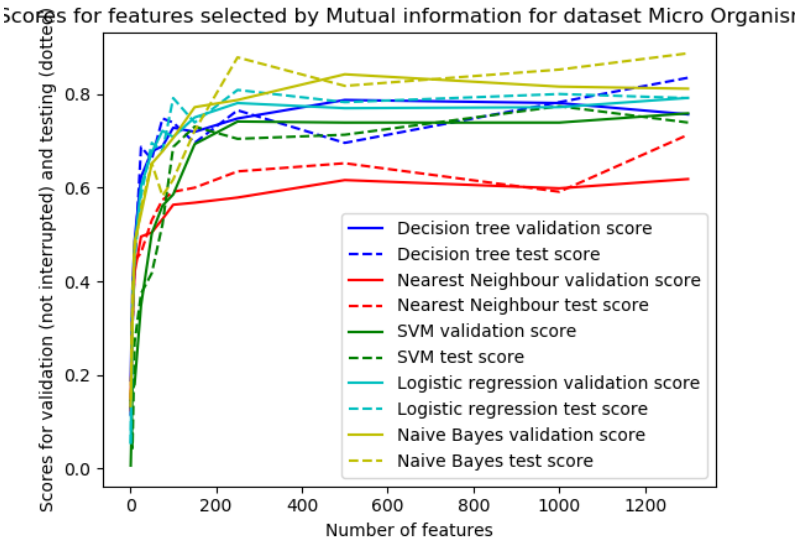


Figure E.1: The validation and test score per feature preserved for the Micro-organisms data set with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

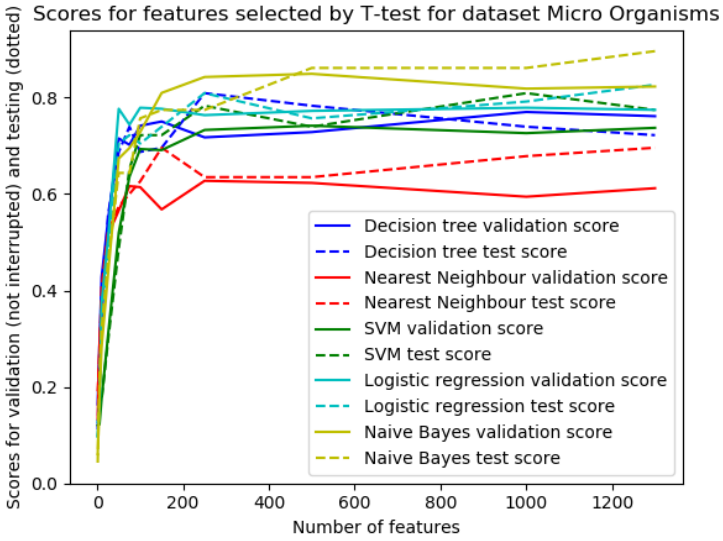


Figure E.2: The validation and test score per feature preserved for the Micro-organisms dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

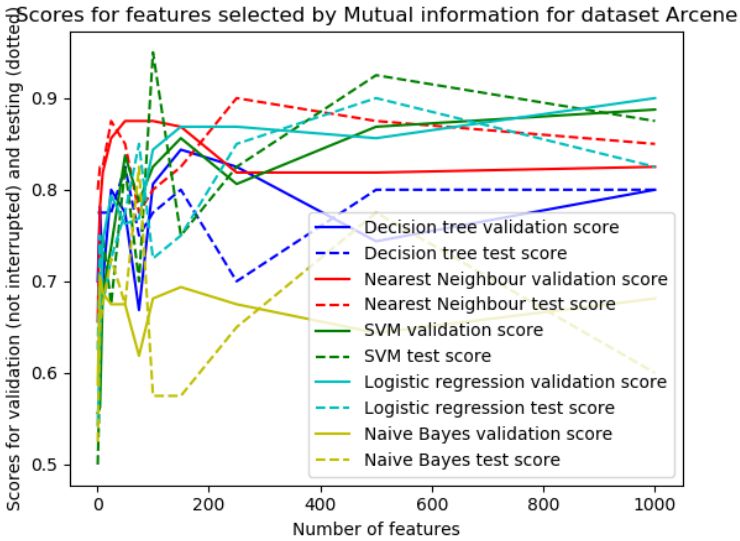


Figure E.3: The validation and test score per feature preserved for the Arcene dataset with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

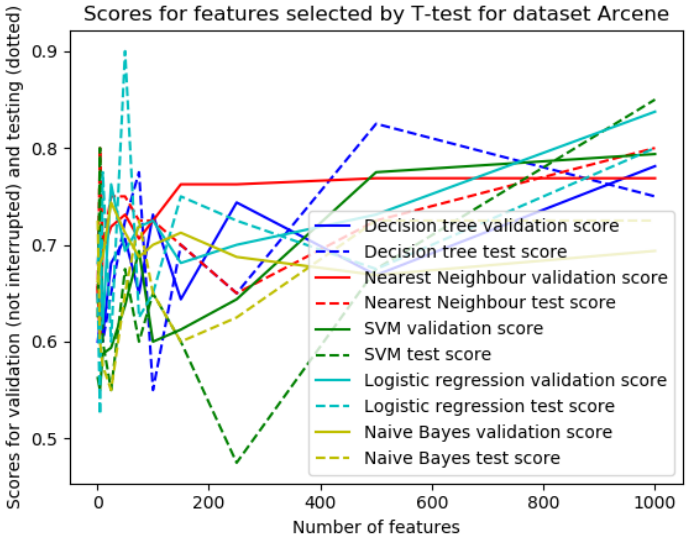


Figure E.4: The validation and test score per feature preserved for the Arcene dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

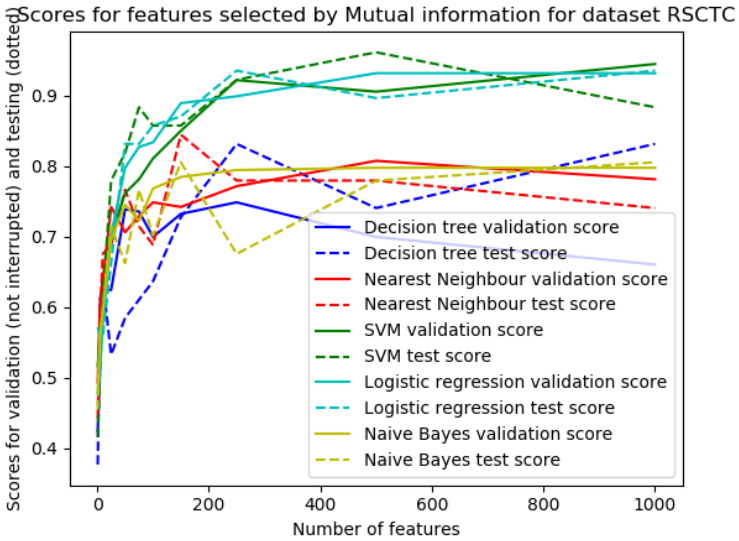


Figure E.5: The validation and test score per feature preserved for the RSCTC dataset with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

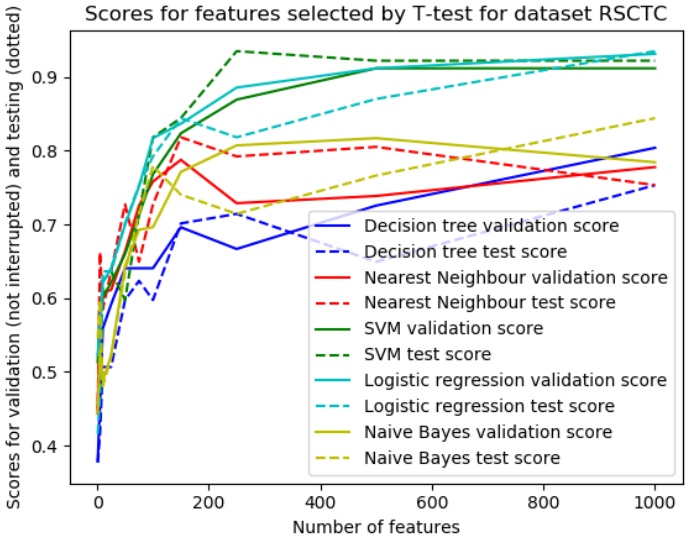


Figure E.6: The validation and test score per feature preserved for the RSCTC dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

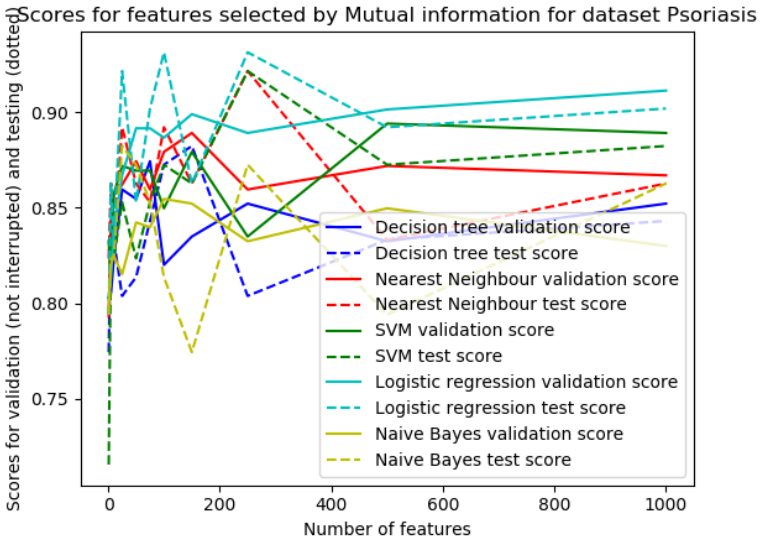


Figure E.7: The validation and test score per feature preserved for the Psoriasis dataset with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

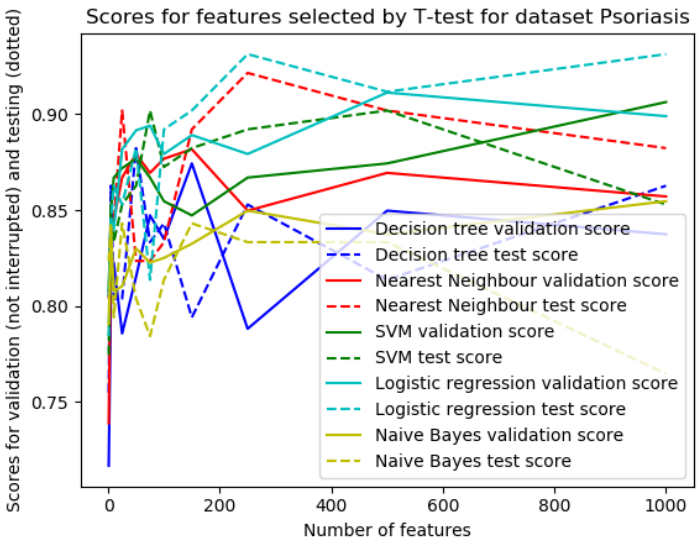


Figure E.8: The validation and test score per feature preserved for the Psoriasis dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

E.2 Feature Selection Exploration Precision And Recall

The average precision (Figure E.9) and recall (Figure E.10). These are shown per data set and per ranking method.

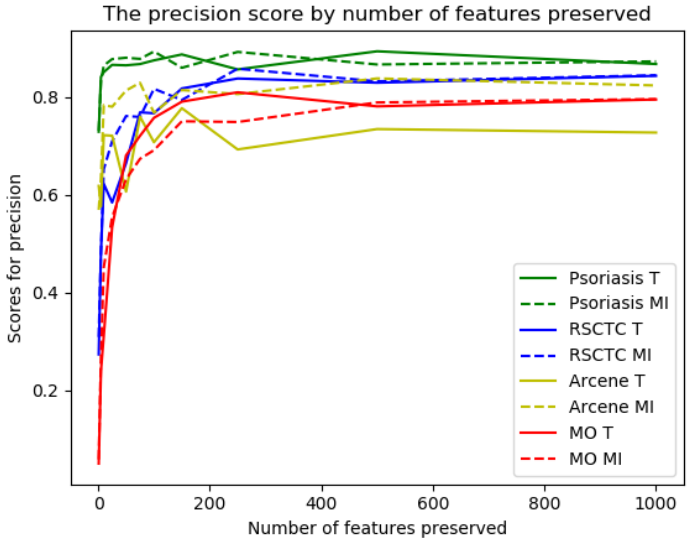


Figure E.9: The average precision shown per dataset and rank.

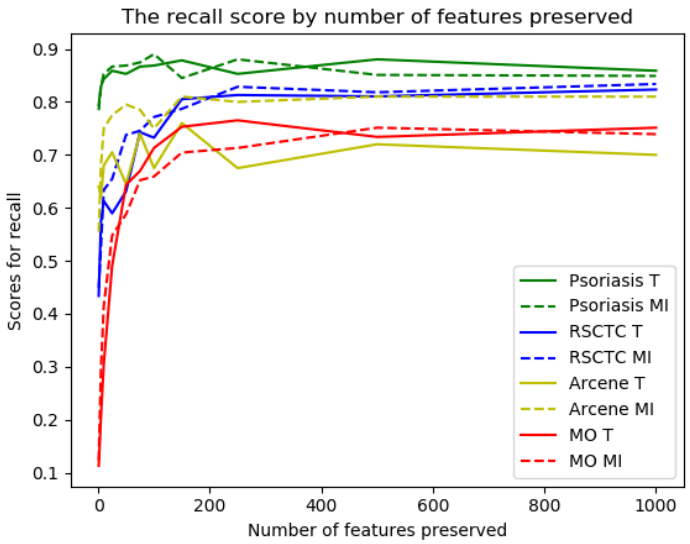


Figure E.10: The average recall shown per dataset and rank.

E.3 Feature Selection Evaluation Plots

The spectra for the feature selection algorithms for the average dataset are shown, using the precision and recall.

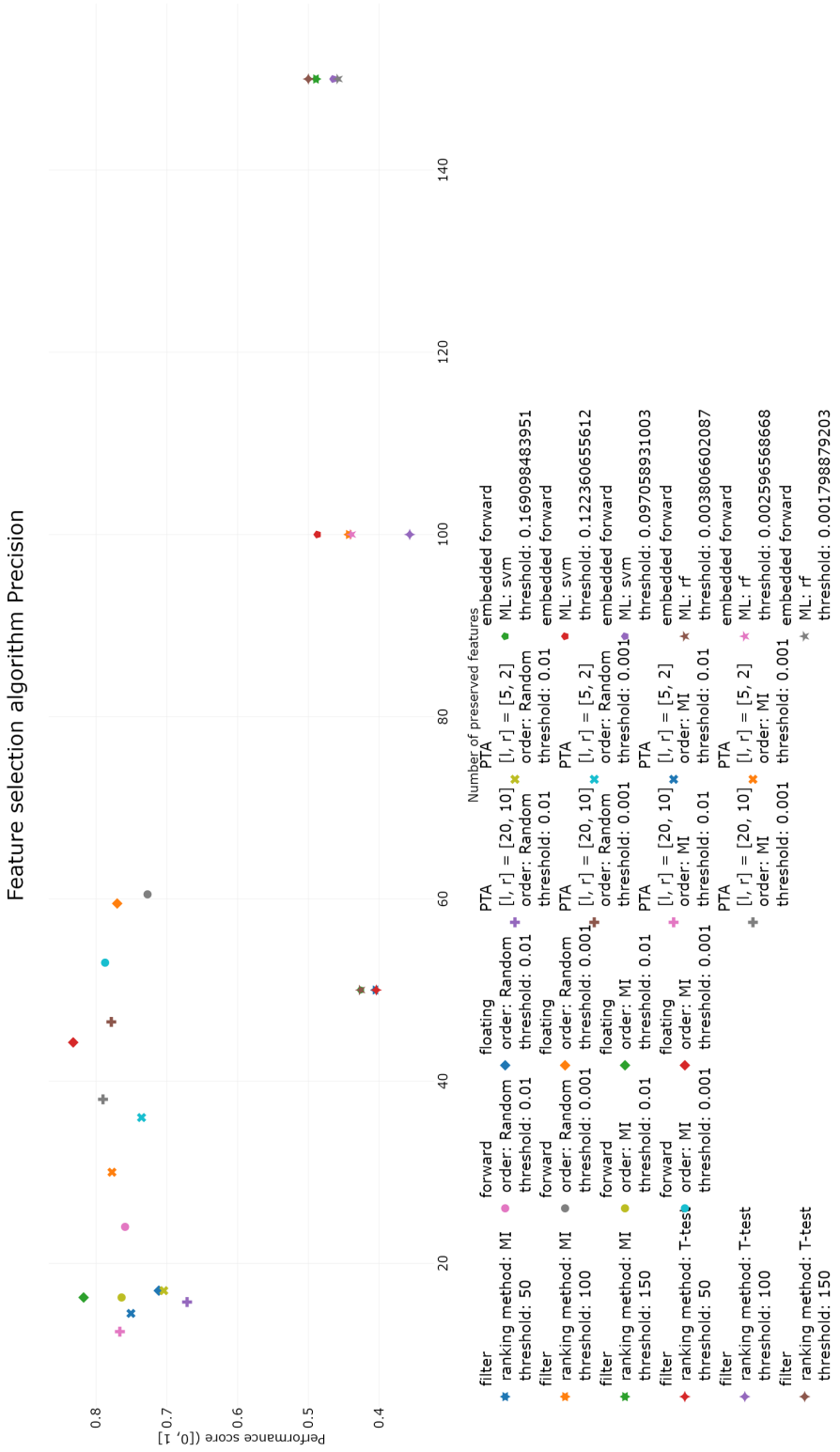


Figure E.11: The precision spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the precision of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

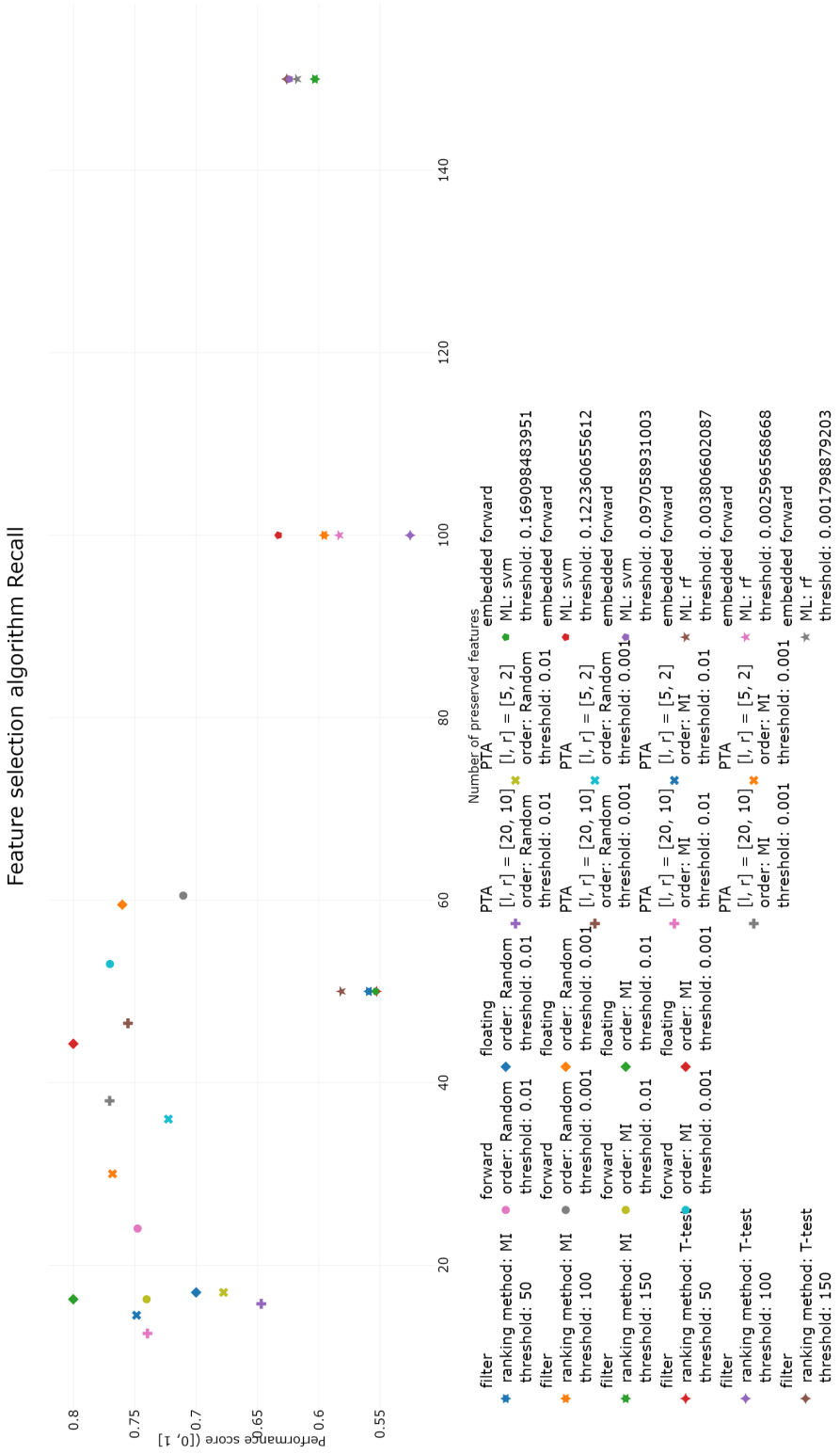


Figure E.12: The recall spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the recall of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

Appendix F

Missing Value Handling Results

F.1 Feature Distribution Tables

The distribution tables for the hepatitis (Tables **??** and **??**), cirrhosis (Tables F.3 and F.4) and cervical cancer (Table **??**) datasets. For the hepatitis and cirrhosis datasets the tables are split in list deletion (Tables **??** and F.3) and imputation (Tables **??** and F.4). For the cervical cancer dataset both tables were combined due to not being any significant difference between the imputation technique outcomes.

Table F.3: Testing for the cirrhosis dataset whether the type of missing values can be represented by the remaining values. This is done by comparing distributions between all sample values and remaining sample values after CCA and WCA. For numeric values, two tests were used, an independent t-test for equality of mean and a Levene's test with the median in brackets to test equality of variance. For ordinal and categorical features the medians and modes where checked respectively to be similar as well as a chi squared test in brackets fro equality of distribution. P-values lower than $p < 0.05$ are marked red for failure of representation, p-values higher than $p > 0.05$ are marked green for correctly being represented. If at least one feature is not represented after CCA, the missing values cannot be MCAR. If at least one feature is not represented after WCA, the pseudo-randomness cannot be corrected by only using weights for other values.

Feature name	**case number**	**number of days**	**status**	**drug**	**age**	**sex**	**day**	**pres. of ascites**	**pres. of hep.**
Value type	Num	Num	Ord	Cat	Num	Cat	Num	Cat	Cat
Missing	0%	0%	0%	0%	0%	0%	0%	3.08%	3.13%
p-values CCA	<0.01 (0.13)	0.69 (<0.01)	False (<0.01)	True (0.99)	0.65 (0.82)	True (0.97)	<0.01 (<0.01)	True (0.99)	True (1.00)
p-values WCA	0.40 (0.62)	0.75 (0.16)	False (0.74)	True (1.00)	0.86 (0.26)	True (0.26)	0.48 (0.02)	True (0.02)	False (0.97)

Feature name	**pres. of spiders**	**pres. of edema**	**serum bili-rubin**	**serum choles-terol**	**albu-min**	**alkaline phos-phatase**	**SGOT**	**plate-lets**	**pro-throm-bin time**
Value type	Cat	Ord	Num	Num	Num	Num	Num	Num	Num
Missing	3.23%	0%	0%	42.2%	0%	3.08%	0%	3.75%	0%
p-values CCA	True (0.99)	True (0.71)	0.08 (0.07)	0.91 0.97	0.03 (<0.01)	0.47 (0.83)	0.01 (0.10)	0.25 (0.34)	0.79 (0.90)
p-values WCA	True (0.99)	False (0.58)	0.29 (0.22)	0.23 (0.21)	0.78 (<0.01)	0.43 (0.23)	0.31 (0.17)	0.21 (0.08)	0.44 (0.21)

Table F.4: Testing for the cirrhosis data set if certain types of imputation create a vastly different distribution for features with missing values. The imputation values are generated with mean imputation, hot deck imputation, k-Nearest Neighbour imputation ($k = 3$), regression imputation and MICE (number of cycles is $s = 5$). For numeric values, two tests were used, an independent t-test for equality of mean and a Levene's test with the median in brackets to test equality of variance. For ordinal and categorical features the medians and modes where checked respectively to be similar as well as a chi squared test in brackets fro equality of distribution. P-values lower than $p < 0.05$ are marked red for failure of representation, p-values close to $p > 0.05$ are marked green for correctly being represented.

Feature name	**pres. of ascites**	**pres. of hep.**	**pres. of spiders**	**serum choles-terol**	**alkaline phos-phatase**	**plate-lets**
Value type	Cat	Cat	Cat	Num	Num	Num
Missing	3.08%	3.13%	3.23%	42.2%	3.08%	3.75%
Mean Imputation	True (0.99)	True (0.98)	True (0.98)	1.00 (<0.01)	1.00 (0.73)	1.00 (0.18)
Hot Deck Imputation	True (0.99)	True (1.00)	True (1.00)	0.95 (0.79)	0.97 (0.95)	0.93 (0.84)
kNN Imputation	True (0.98)	True (0.99)	True (0.98)	0.69 (0.40)	0.89 (0.82)	0.50 (0.86)
Regression Imputation	True (0.97)	True (0.98)	True (0.97)	0.59 (0.62)	0.64 (0.85)	0.32 (0.89)
MICE (5 cycles)	True (0.96)	True (0.98)	True (0.97)	0.31 (0.11)	0.77 (0.90)	0.56 (0.72)

F.2 Classification Plots

The classification results for each dataset separately. A figure for the Hepatitis (Figure F.1), Cirrhosis (Figure F.2) and the Cervical Cancer dataset (Figure F.3) are shown.

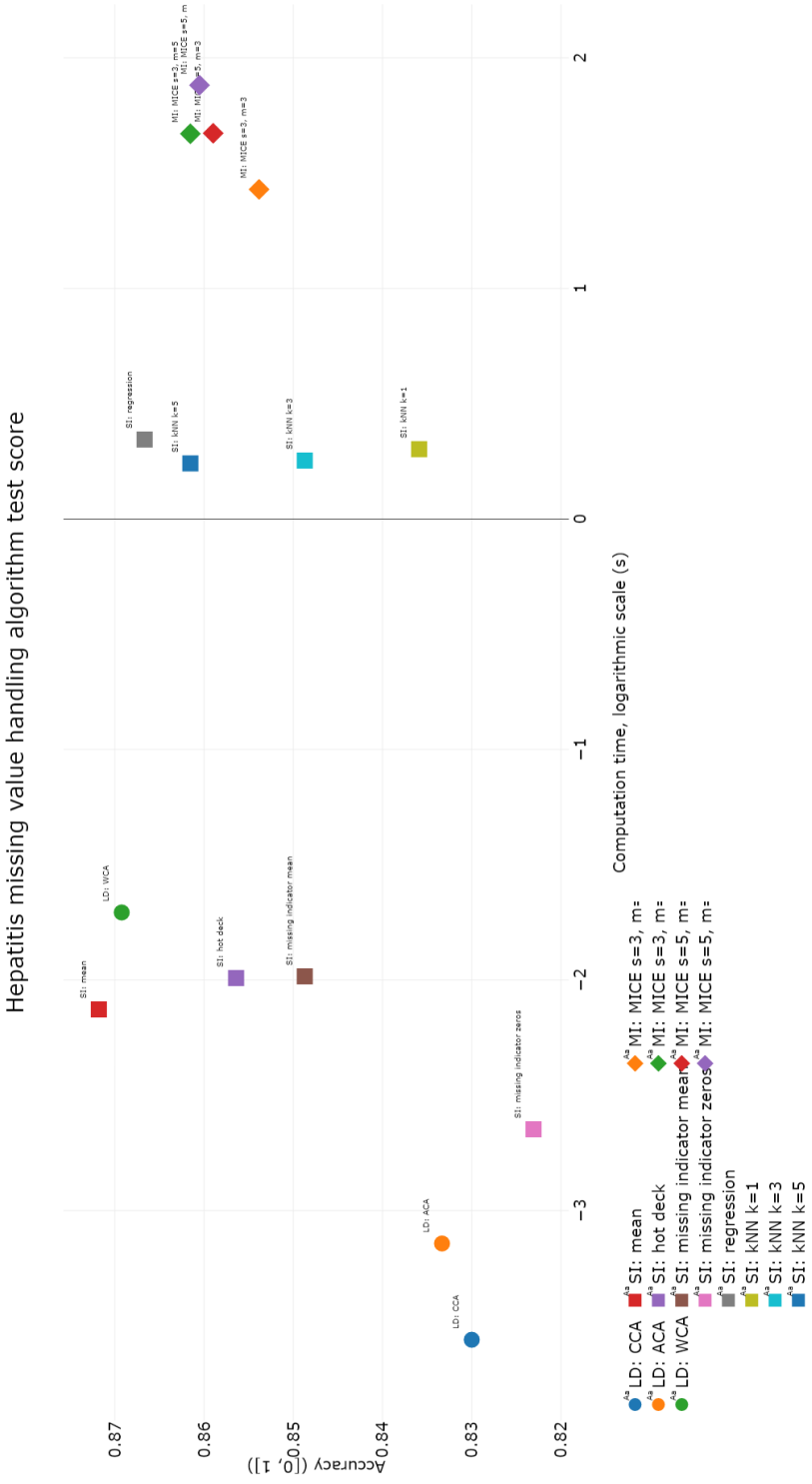


Figure F.1: A plot showing the accuracy for the Hepatitis dataset after using a missing value handling algorithm. The x-axis shows the computation time for the missing values, done on a 10-logarithmic scale due to the big differences between them. The y-axis shows the accuracy on a scale of 0 to 1. The explanation of every data point can be found both in the text on the top right of the data point as well as in the legend.

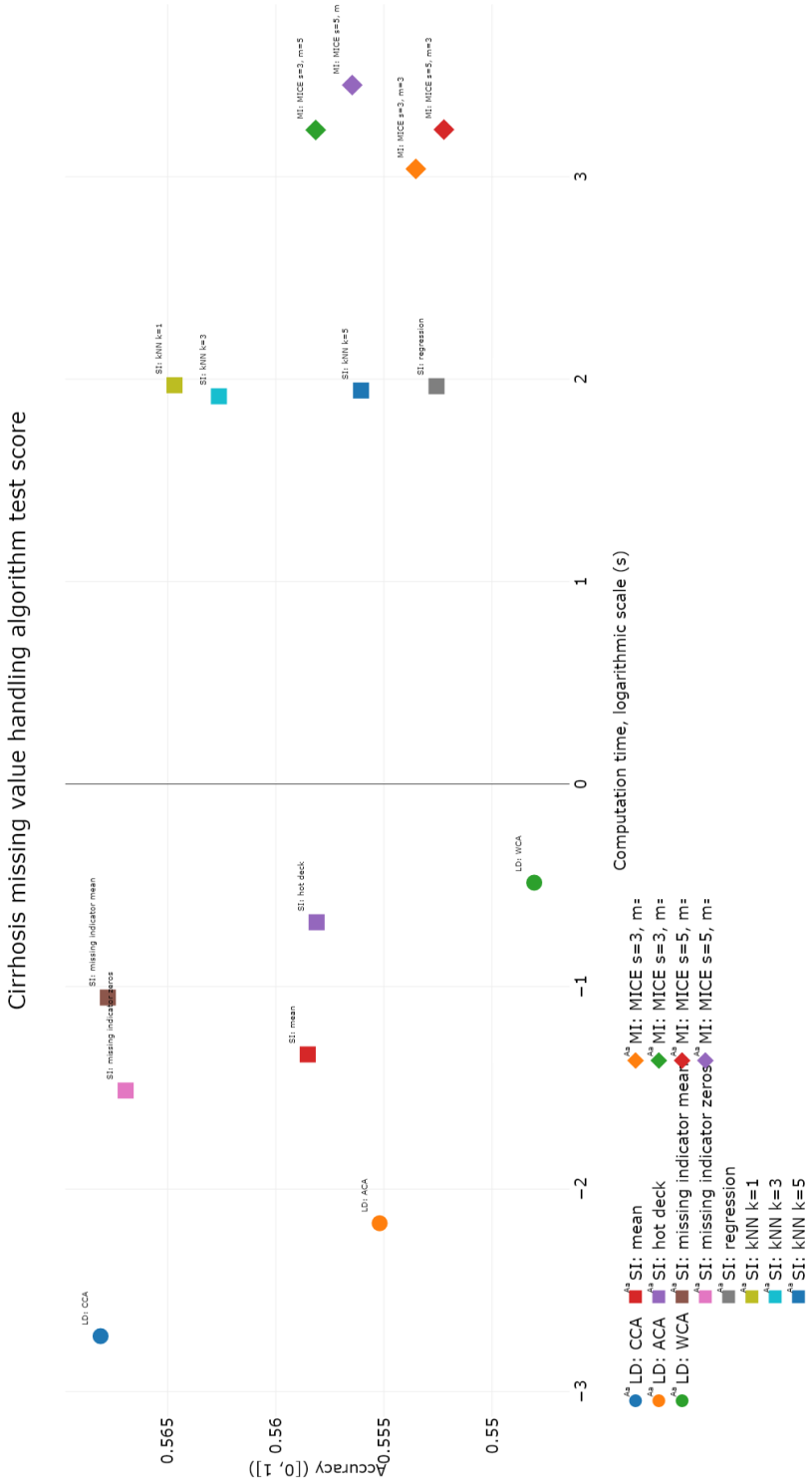


Figure F.2: A plot showing the accuracy for the Cirrhosis dataset after using a missing value handling algorithm. The x-axis shows the computation time for the missing values, done on a 10-logarithmic scale due to the big differences between them. The y-axis shows the accuracy on a scale of 0 to 1. The explanation of every data point can be found both in the text on the top right of the data point as well as in the legend.

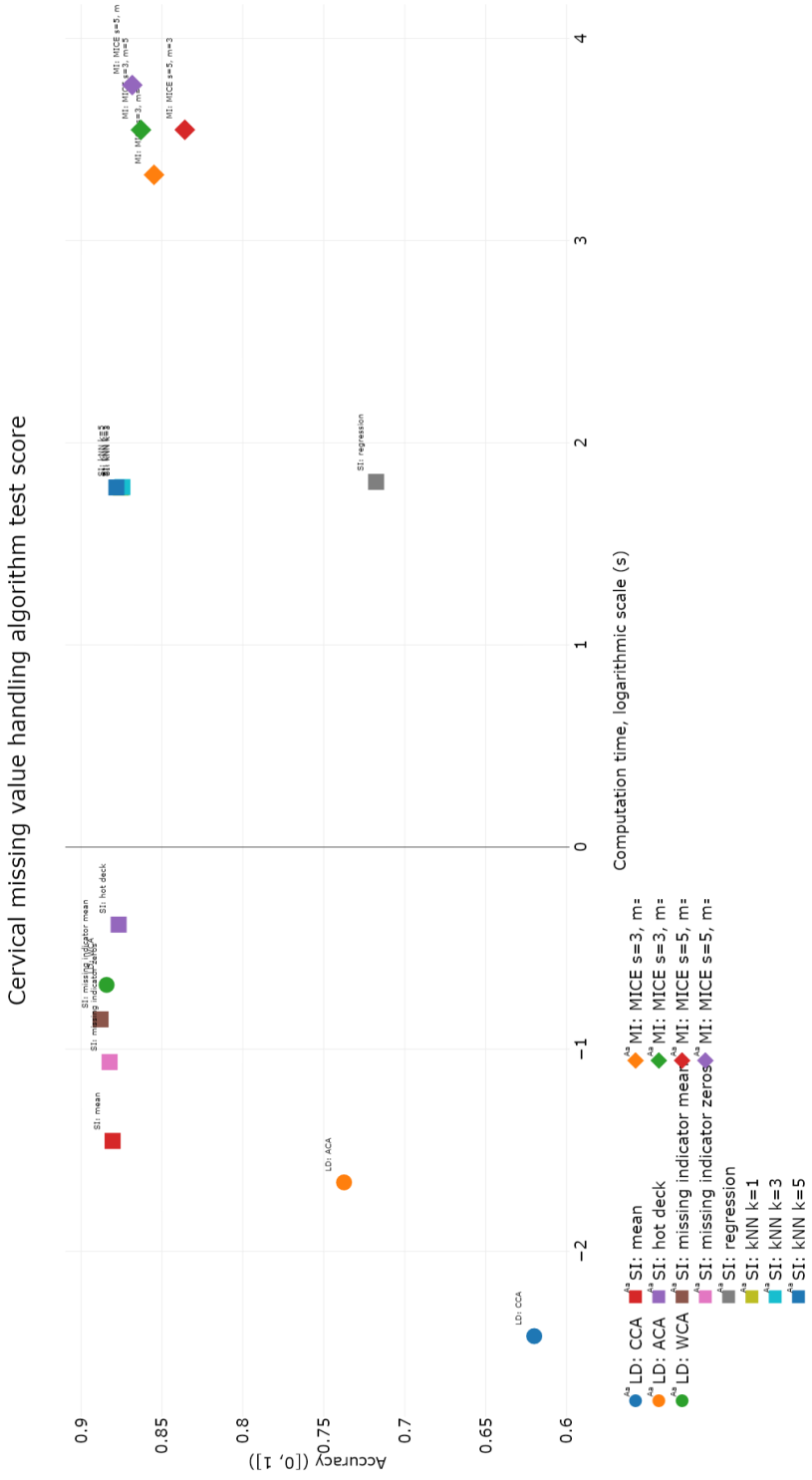


Figure F.3: A plot showing the accuracy for the Cervical cancer dataset after using a missing value handling algorithm. The x-axis shows the computation time for the missing values, done on a 10-logarithmic scale due to the big differences between them. The y-axis shows the accuracy on a scale of 0 to 1. The explanation of every data point can be found both in the text on the top right of the data point as well as in the legend.

Appendix G

Dataset Exploration Meta-features

G.1 Meta-features

Meta-features are values gathered out of the dataset. These values can be computed easily with counting aspects or can be found by doing a more extensive analysis. These meta-features show valuable information that can be used for better insights in the dataset and can be used further on in analyses.

G.1.1 Basic Meta-features

The available dataset at the start of a project can have several different important aspects that need to be taken into account. Examples are the size of the dataset and the distribution of the values in it. These dataset aspects are called meta-features [152]. These meta-features are used in multiple analysis techniques and are an important aspect in meta-learning [152, 214, 215, 216]. These meta-features are also usable without meta-learning however, as they can help by a global exploration of the data. With this exploration directions for preprocessing and analysis can be found, aiding the data analyst in his research. Five different basic meta-feature types are discussed:

- **Data size**
  Finding out the size of the data shows very quickly whether possible limitations or challenges are present. The data size is split into the number of features and the number of samples.
- **Feature types**
  There are two main types of features that can be used in a dataset, categorical features and numeric features. Further sub-categorization of the feature types is possible, but is harder to detect. An example of this would be ordinal. Ordinal is categorical data with an order in the categories. In practice, ordinal features either have categorical values with given ordering or numeric values losing the names and are treated as the type they are given.
- **Missing values**
  Missing values may be present in the dataset and need some type of processing. This processing can be done in multiple ways which depend on the ratio of missing values as well as the distribution of them over features and samples.
- **Output type**
  The output of the dataset can be both categorical and numeric. The type of output changes the choice between using classification and regression respectively. For categorical data, the distribution of the separate categorical classes and whether there are majority and minority classes.
- **Feature cardinalities**
  Cardinalities shows the number of unique options available, as there is a significant difference

between features with only 2 and with as much as 100 different categories. The same holds for numeric data as with low cardinality the data may be less spread out.

G.1.2 Statistical Meta-features

The dataset can also be approached in a statistical manner. The feature values usually follow a certain distribution. Also similarities between features are worth being looked into. Three statistical meta-feature types are discussed:

- **Feature distributions**
  The distribution of numeric features is usually shown by using four values, the mean, standard deviation, skewness and kurtosis. The combination of these four indicate distributions that afterwards can be compared for finding difference between distributions
- **Principal component analysis**
  An effective way of reducing the number of features by changing them is principal component analysis (PCA). PCA collects the data of all possibly correlated features and creates new uncorrelated features called principal components. To reduce the number of features as much as possible, these principal components are created one by one, giving as much variability to a new principal component as much as possible. Therefore the ratio of total explained variability of the first few principal components can show the presence of correlation within the features and whether feature reduction can be done to lower the dimensionality. In research for which no direct link is needed between input and output, PCA is used to significantly reduce the number of features [217].
- **Correlation**
  Whereas PCA creates new uncorrelated features to test variability between them, also a direct look at the correlation can be done by comparing all features with each other [77]. For a high number of features, this can be computationally very exhaustive, however it is a fairly easy way to spot multicollinearity in a dataset [78].

G.1.3 Information-theoretic Meta-features

Information-theoretic meta-features are focused on the information present in the dataset, how useful this information is and if this information is relevant when linking it to the output. These meta-features are focused on the entropy and mutual information and on the combination of these:

- **Entropy**
  The entropy shows the potential information present in features and the output. Using entropy best works with categorical features, however binning numeric features sometimes can show information gain, too. This information can be seen in the distribution of values over separate categories in categorical data and if the values are more spread out over the categories, the potential information in the feature is higher. This entropy is called attribute entropy is computed by using spreading ratios r_c for every category c (Equation G.1). This spreading ratio r_c is the ratio r of instances that are category c.

$$\text{entropy} = -\sum_{c} r_c \log r_c \tag{G.1}$$

  The higher the entropy, the more potential information is present. The use of the potential information gain can computed by combining the feature categories and the output classes into new categories and then compute the entropy. If the new so-called joint entropy is very close to the attribute entropy, the potential information can be used to predict output classes and therefore the feature is a good predictor for the output [81].

- **Mutual information**
  Mutual information is another way of matching features with the results. The relevance of using a feature to predict the output is used in the equation to compute mutual information [218].

- **Entropy and mutual information**
  The outcome values of entropy and mutual information can be combined for measuring ratios between signal and noise. One way to do that is by dividing the class entropy by the averaged mutual information (Equation G.2). The class entropy needs a multitude of features with a certain mutual information for perfect prediction. Since mutual information is lower than the class entropy an estimation can be done for the number of features needed to explain the complete class entropy.

$$\text{equivalent number features} = \frac{\text{class entropy}}{\frac{1}{n}\sum_{1}^{m}\text{MI(x)}} \tag{G.2}$$

  A second way of combining attribute entropy and mutual information is by finding out the noise to signal ratio. The attribute entropy of a feature is only partly interesting as not all entropy can be used to predict the output. Mutual information is the part of entropy that can be used in prediction. To find out how much the ratio between noise and information is present, these two can be combined (Equation G.3).

$$\text{noise to signal ratio} = \frac{\text{attribute entropy} - \text{MI}}{\text{MI}} \tag{G.3}$$

G.1.4 Landmarking Meta-features

The quality of analysis can also be tested by performing small analyses. These small analyses can show how well it would perform on the dataset and some small conclusions can be made on them. This is called landmarking, creating very preliminary results in order to reach an outcome [219].

Useful landmarks are based around well known machine learning algorithms. Decision trees are an example of such an algorithm, as it is a well known and easy to understand approach in machine learning. Other examples are Naive Bayes, k-nearest neighbour, linear discriminant analysis and logistic regression. Quick analyses can show results that can be used as outcome.

These results and therefore the landmarks are based on the quality of the machine learning algorithm. This quality can be measured in multiple ways, such as the accuracy, F1-score and Cohen's kappa.

G.2 Meta-feature Values Hepatitis

All values for the first analysis for the dataset hepatitis are given here.

- **Basic meta-features**

1. *Data size*
   Number of instances: 155
   Number of features: 19
   Number of classes: 2
   Dimensionality: 0.12

2. *Data types*
   Number of numeric features: 6
   Number of categorical features: 13

Ratio of numeric features: 0.32
Ratio of categorical features: 0.68

3. *Missing values*
Number of missing values: 167
Ratio of missing values: 0.06
Number of instances with missing values: 75
Ratio of instances with missing values: 0.48
Number of features with missing values: 15
Ratio of features with missing values: 0.79

4. *Output type*
Mean of class probability: 0.5
Standard deviation of class probability: 0.42
Minimum of class probability: 0.21
Maximum of class probability: 0.79
Minority class size: 32
Majority class size: 123

5. *Feature cardinalities*
(Table G.1)

Table G.1: The cardinality meta-features for the Hepatitis dataset

Cardinality	**Categorical features**	**Numeric features**
Mean	2	54.67
Standard deviation	0	24.09
Minimum	2	30
Maximum	2	85

- **Statistical Meta-features**

1. *Feature distributions*
(Table G.2)

Table G.2: The numeric feature distribution of the hepatitis dataset

Value distributions	**Means**	**Standard deviations**	**Skewnesses**	**Kurtoses**
Mean	49.92	29.74	1.28	4.49
Standard deviation	42.57	29.73	1.46	6.11
Minimum	1.43	0.652	-0.12	-0.53
Maximum	105.33	89.65	3.18	14.02
First quartile	13.16	89.65	0.11	0.25
Second quartile	51.53	17.72	0.86	1.66
Third quartile	79.88	44.35	2.51	8.18

2. *Principal component analysis*
Explained variation component 1: 0.69

Explained variation component 2: 0.26
Explained variation component 3: 0.04
Eigenvalue component 1: 7919.97
Eigenvalue component 2: 2924.60
Eigenvalue component 3: 412.14
Determinant value: 3.48

- **Information-theoretic meta-features**

1. *Entropy*
   Class entropy: 0.51
   (Table G.3)
2. *Mutual information*
   (Table G.3)

Table G.3: The entropies and mutual information values of the hepatitis dataset

Distributions	**Categorical features**			**Numeric features**		
	Attribute entropy	**Joint entropy**	**Mutual information**	**Attribute entropy**	**Joint entropy**	**Mutual information**
Mean	0.54	1.01	0.04	1.02	1.44	0.08
Minimum	0.33	0.81	0.00	0.60	1.05	0.01
Maximum	0.69	1.19	0.09	1.32	1.80	0.17
First quartile	0.43	0.93	0.01	0.76	1.19	0.03
Second quartile	0.51	1.01	0.03	1.16	1.50	0.6
Third quartile	0.67	1.12	0.06	1.25	1.68	0.13

3. *Entropy and mutual information*
   Equivalent number of categorical features: 13.8
   Equivalent number of numeric features: 6.42
   Categorical noise to signal ratio: 13.60
   Numeric noise to signal ratio: 11.90

- **Landmarking meta-features**
  (Table G.4)

G.3 Meta-feature Values Micro-Organisms

All values for the first analysis for the Micro-organisms are given here.

- **Basic meta-features**

1. *Data size*
   Number of instances: 571
   Number of features: 1300
   Number of classes: 20
   Dimensionality: 2.28

2. *Data types*
   Number of numeric features: 1300

Table G.4: The landmarking meta-features of the hepatitis dataset

Machine learning algorithm	**Error rate**	**Cohen's kappa**
Naive bayes	0.22	0.48
k-Nearest neighbour (k = 1)	0.32	0.15
Decision stump	0.21	0.13
Random tree (depth = 1)	0.19	0.21
Random tree (depth = 2)	0.23	0.15
Random tree (depth = 3)	0.27	0.21
Linear Discriminant	0.17	0.46

Number of categorical features: 0
Ratio of numeric features: 1
Ratio of categorical features: 0.0

3. *Missing values*
   Number of missing values: 0
   Ratio of missing values: 0.0
   Number of instances with missing values: 0
   Ratio of instances with missing values: 0.0
   Number of features with missing values: 0
   Ratio of features with missing values: 0.0

4. *Output type*
   Mean of class probability: 0.05
   Standard deviation of class probability: 0.02
   Minimum of class probability: 0.02
   Maximum of class probability: 0.1
   Minority class size: 11
   Majority class size: 60

5. *Feature cardinalities*
   (Table G.5)

- **Statistical Meta-features**

1. *Feature distributions*
   (Table G.6)
2. *Principal component analysis*
   Explained variation component 1: 0.14
   Explained variation component 2: 0.10
   Explained variation component 3: 0.08

Table G.5: The cardinality meta-features for the Micro-organisms dataset

Cardinality	Categorical features	Numeric features
Mean	NaN	45.29
Standard deviation	NaN	44.62
Minimum	NaN	1.0
Maximum	NaN	296

Table G.6: The numeric feature distribution of the Micro-organisms dataset

Value distributions	Means	Standard deviations	Skewnesses	Kurtoses
Mean	67437.29	343176.13	10.67	177.27
Standard deviation	191740.68	700237.17	6.95	168.76
Minimum	0.0	0.0	0.0	0.0
Maximum	2882892.51	7792765.65	23.90	571.00
First quartile	696.92	9641.40	6.58	50.64
Second quartile	13117.02	94709.23	10.15	122.68
Third quartile	50446.57	356651.90	15.51	274.29

Eigenvalue component 1: 112301000896299.64
Eigenvalue component 2: 83271983089506.69
Eigenvalue component 3: 63192806579339.62
Determinant value: ∞

- **Information-theoretic meta-features**

1. *Entropy*
   Class entropy: 2.91
   (Table G.7)
2. *Mutual information*
   (Table G.7)

Table G.7: The entropies and mutual information values of the Micro-organisms dataset

Distributions	Categorical features			Numeric features		
	Attribute entropy	Joint entropy	Mutual information	Attribute entropy	Joint entropy	Mutual information
Mean	NaN	NaN	NaN	0.11	2.98	0.04
Minimum	NaN	NaN	NaN	0.0	2.91	0.00
Maximum	NaN	NaN	NaN	0.83	3.49	0.27
First quartile	NaN	NaN	NaN	0.03	2.93	0.01
Second quartile	NaN	NaN	NaN	0.08	2.96	0.03
Third quartile	NaN	NaN	NaN	0.16	3.01	0.06

3. *Entropy and mutual information*
   Equivalent number of categorical features: NaN
   Equivalent number of numeric features: 70.27
   Categorical noise to signal ratio: NaN

Numeric noise to signal ratio: 1.64

- **Landmarking meta-features**
  (Table G.8)

Table G.8: The landmarking meta-features of the Micro-organisms dataset

Machine learning algorithm	**Error rate**	**Cohen's kappa**
Naive bayes	0.20	0.79
k-Nearest neighbour (k = 1)	0.34	0.64
Decision stump	0.83	0.08
Random tree (depth = 1)	0.87	0.05
Random tree (depth = 2)	0.78	0.14
Random tree (depth = 3)	0.71	0.22
Linear Discriminant	0.31	0.67