Chapter 1

Automatic feature selection for high-dimensional biomedical data

1.1 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 [1, 2] which resulted in multiple algorithm proposals [3, 4, 5, 6] and tests on their performance [7, 8]. 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.

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 [17]. *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 [18]. Packages available by *Anaconda* and often used in the analyses are *NumPy* [19], *SciPy* [20] and *Scikit-learn* [21]. Aside from these widely used packages, the package *TPOT* [23] is used, as well.

Dimensionality TO BE ADDED
The dimensionality of a dataset in literature is the number of features of that dataset[9, 3, 5]. 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.
Dataset issues TO BE ADDED 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 [10, 11].

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 [10]. 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 [12], volume on its own becomes less of an issue. Volume does however become a challenge when it is combined with heterogeneity

Table 1.1: 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

and quality [13, 14].

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 [15]. 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 [16].

1.1.1 Data Symbols

TO BE DISCUSSED - ONLY TABLE?

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 1.1) 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.

1.1.2 Related work

A good overview of all the techniques used can be found in a survey article [24] explaining all the algorithms used, as well as several others. Similar research has been done prior to this one. Numerous algorithms have been proposed for feature selection [3, 4, 5, 6, 25, 26] and simple performance tests are done to do research on both these and other algorithms [1, 7, 8, 27, 28]. Most of these focus on a much smaller number of features than in this research however, not testing the algorithms for data with more than 1000 features. Other feature selection research

focus more on one type of the feature selection, being wrapper methods [29, 30, 31] or embedded methods [32]. A select number of articles focused on bigger feature set sizes. One research looked at bigger not necessarily biomedical datasets and tried to use support vector machines for feature selection, one of these datasets also being the Arcene dataset [33]. A similar research focused on feature selection for a microarray dataset of 7000 and showed that feature selection is beneficial [34]. Another research looked at 35 datasets with a range of features between 37 and 50000 and what type of feature selection works best [35]. They used five advanced algorithms and therefore harder to understand, whereas the ones used in our research are more intuitive. Another very similar research evaluated feature selection algorithms, too, but did not have the focus on the number of features. Much research is done on feature selection algorithms, as can be found in the related work. This work was made for using feature selection for biomedical data however, containing several benefits:

- The main goal was to create a feature subset as small as possible with a performance as high as possible, which only rarely was a goal in other research [36]. This resulted in both a score being made that incorporates both the feature subset size and the performance as the FS_score and the usage of this score to evaluate the usefulness of every feature. While not being a main goal, the computation time is also stored for additional investigations on that.
- Feature selection is done on very big biomedical datasets of more than 1000 features, therefore the feature selection has a big influence. This is not often done, whereas these datasets will become more available as well as investigated in the future.
- Only basic feature selection algorithms are used so other users can easily understand the basic principles and many different ones are investigated, making this work a very extensive search for well performing feature selection algorithms. Moreover, these algorithms are also very different in their feature selection, also showing the benefits and drawbacks of every type of algorithm.
- At last automation of feature selection techniques is added, too. Automated machine learning is a very interesting approach to extracting information out of a dataset. Properly implementing feature selection in automated machine learning opens it up for high dimensional datasets, as well.

1.2 Methods

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.

1.2.1 Datasets

Four datasets were used as a case study for the feature selection algorithms . Two sets are microarray datasets that are used for research on psoriasis [37, 38, 39, 40] and cancer [41]. Two other sets are mass spectrometry data sets, used for research on cancer [42] and micro organisms [43]. 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 1.2). 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.

Micro-array Datasets

TO BE ADDED, ONLY GENERAL MENTIONS?

Table 1.2: 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 [37, 38, 39, 40] –
Cancer	Micro-array	54675	383	9	- Used in a data mining challenge [41]
Cancer	Mass Spectrometry	10000	200	2	- Created for the NIPS conference [42] - Several probe features are present –
Micro-organisms	Mass Spectrometry	1300	571	20	- Originates from a micro organisms study [43]

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 [44]. 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 [45].

Micro-array data is known to present challenges in data quality and needs normalisation for the data to be useful [45]. 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 [37, 38, 39, 40]. 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 [41]. 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.

Mass Spectrometry Datasets

TO BE ADDED, ONLY GENERAL MENTIONS?

Mass spectrometry data contains information on proteins and peptides [47, 48]. Analysis of this information is used in studies about proteins, also known as proteomics [49]. 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 [50].

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 [42]. 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 [43]. 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 of a Gram stain test [52]. The groups differ in size varying from 11 to 60 samples, making a total of 571 samples.

1.2.2 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 [53]. Several techniques are available to choose a representation subset and the effectiveness of these techniques has been tested multiple times [27, 54]. These techniques can be grouped accordingly in three different categories: *filter methods*, *wrapper methods* and *embedded methods* [24]. 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.

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 [55, 24]. 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[56]. 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 [56].

- *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.

  Mutual information TO BE COMBINED
  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 1.1). In this equation the probability density function p is used to find the mutual information (MI) between variables x and y [57]. 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 [58].

$$MI(x,y) = \sum_x \sum_y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{1.1}$$

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 [25]. For the t-test, a p-value of 0.05 is often chosen as the threshold value [59, 60].

Algorithm 1 A basic top n filter algorithm [55]

```
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
```

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 [55, 24].

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 [61]. 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 [62]. 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 [61]. Evaluation functions can be based on conditional independence [61, 29], showing the difference in performance for a subset with and without a certain feature. Other evaluation functions are based on machine learning techniques [28, 24], rating the ability to classify the outcome. Several interesting approaches for evaluation functions are shown in maximum relevance, minimum redundancy algorithms [26, 30, 31] (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 2) for understanding [61].

  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 3) for understanding, as well [61].

  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 [61].

Algorithm 2 A forward selection sequential search algorithm [61]

```
1: procedure SequentialForwardSelection(X, y, F, J, α, l)
2:     F_selected ← ∅                          ▷ Start with empty feature set selection
3:     for f in F do                           ▷ For all features in F
4:         if J(X_{F_selected}, y, X_f) > α then   ▷ Check if evaluation is higher than α with feature f
5:             F_selected ← F_selected ∪ {f}       ▷ Add f to the feature set selection
6:         end if
7:         if length(F_selected) = l then      ▷ Stop if size of feature set selection has been reached
8:             break
9:         end if
10:    end for
11:    return F_selected
12: end procedure
```

Algorithm 3 A backward selection sequential search algorithm [61]

```
1: procedure SequentialBackwardSelection(X, y, F, J, α, r)
2:     F_selected ← F                          ▷ Start with complete feature set selection
3:     for f in F_selected do                  ▷ For all features in F_selected
4:         if J(X_{F_selected\{f}}, y, f) < α then   ▷ Check if evaluation is higher than α without f
5:             F_selected ← F_selected\{f}        ▷ Remove f from the feature set selection
6:         end if
7:         if length(F\F_selected) = r then    ▷ Stop if feature removal limit has been reached
8:             break
9:         end if
10:    end for
11:    return F_selected
12: 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). 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 [61].

  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, doing a forward and backward search iteratively. 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 [61].

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.

Also, these methods are dependent on the evaluation function and are known to be prone for over fitting [61, 24].

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 [63]. Usually this combination involves using the weights given to features by machine learning algorithms [64].

One approach to solve feature selection is based on contribution relaxation minimization. Since the theoretical approach of embedded methods can be very computationally intensive an approximation of the minimization problem is given, showing an example embedded method [63].

- *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 [63].

  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 4). 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 **??**). 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 [63].

Algorithm 4 An embedded forward selection algorithm [63]

```
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
```

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 [32, 36, 65, 66]. 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 [67, 68, 55]. 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 [69].

Tree-based Pipeline Optimization Tool

TO DO: Add automated machine learning and meta-learning paragraphs and simplification

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. *TPOT* consists of preprocessing and machine learning algorithms, that form the backbone of the pipelines. 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).

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 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.

1.3 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 **??**) were used as example datasets for the experiments.

1.3.1 Feature Selection Quality

To find out the quality of the feature selection, multiple machine learning algorithms are used [70]. 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 1.3.1). 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.

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

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 stud. 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 1.1). 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%, which meant for the example filter method that for 50 features an optimum was reached (Figure 1.1). Because of this empirically found desired trade-off, $\beta = 0.99$ will be chosen in this project when using *FS_score*.

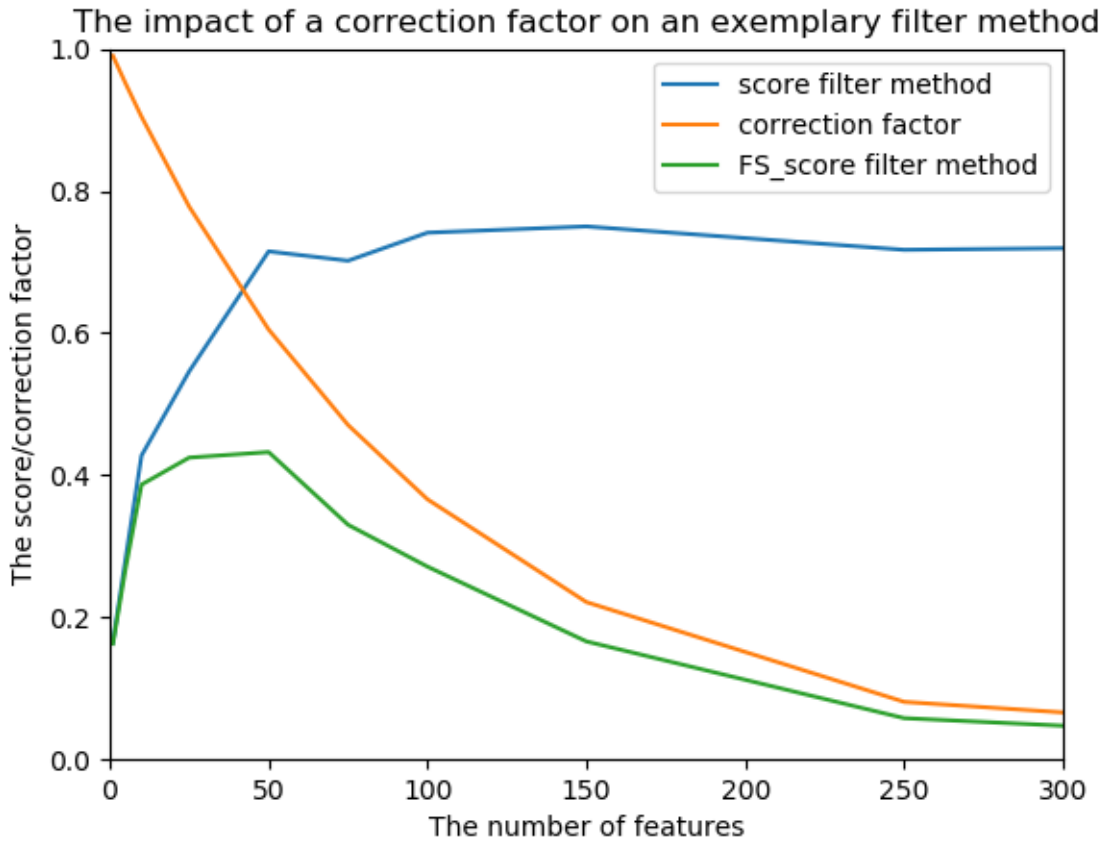

Figure 1.1: An example of the impact of the correction factor on the score, in this case accuracy. The shown correction factor uses $\beta = 0.99$. On the x-axis the number of features is shown and on the y-axis the value for the original accuracy, the correction factor and the *FS_score*.

1.3.2 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 1.3). 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.

1.3.3 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 1.3: The four meta-parameters with their possible values in the first experiment.

Variable	Description	Values
Dataset	The datasets used (subsection 1.2.1)	Psoriasis RSCTC Arcene Micro-Organisms
Ranking method	The method used for ranking the features (Subsection 1.2.2)	– 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
F1-score computation	The machine learning algorithms used to compute the F1-score (subsection 1.3.1)	– 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 1.4) and all separate meta-parameters are explained here. 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 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 1.2.2).

1.3.4 TPOT Feature selection integration

TPOT is an effective tool to find the best machine learning pipeline for a certain dataset. Two 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 1.2.2). 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

Table 1.4: 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 2)	- Order: Random, Mutual Information - Evaluation: Naive Bayes - Alpha: 0.01, 0.001
	PTA	– - Order: Random, Mutual Information - Evaluation: Naive Bayes - [l, r] = [20, 10], [5, 2] - Alpha: 0.01, 0.001
	Floating search	– - Order: Random, Mutual Information - Evaluation: Naive Bayes - Alpha: 0.01, 0.001
Embedded methods	Forward selection (Algorithm 4)	- Machine Learning: SVM, RandomForests - Threshold: 50, 100, 150 features

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 experiment consists of multiple runs of *TPOT* and all of these steps are also shown in an explanatory table (Table 1.5). All four datasets are tested (Subsection 1.2.1) and the accuracy is changed to the feature sensitive *FS_score* with $\beta = 0.99$ (Equation 1.3.1), as previously discussed (Subsection 1.3.1). 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.

1.4 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.

1.4.1 Feature Selection Exploration Results

A difference in score quality for the datasets was visible (Figure 1.2). 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.

Table 1.5: 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 1.2.1)
Performance measurement	One type: *FS_score* - $\beta = 0.99$	Addition of correction factor (Equation 1.3.1)
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

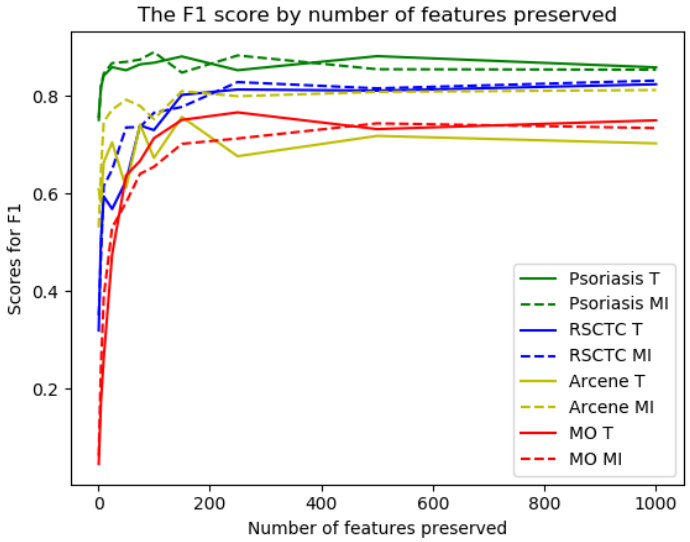

Figure 1.2: The average validation F1-scores shown per dataset and rank.

1.4.2 Feature Selection Algorithms Evaluation Results

The figure showed that all wrapper algorithms preserved on average less than 61 features for these settings, whereas the performance seems to average around 75%. 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.

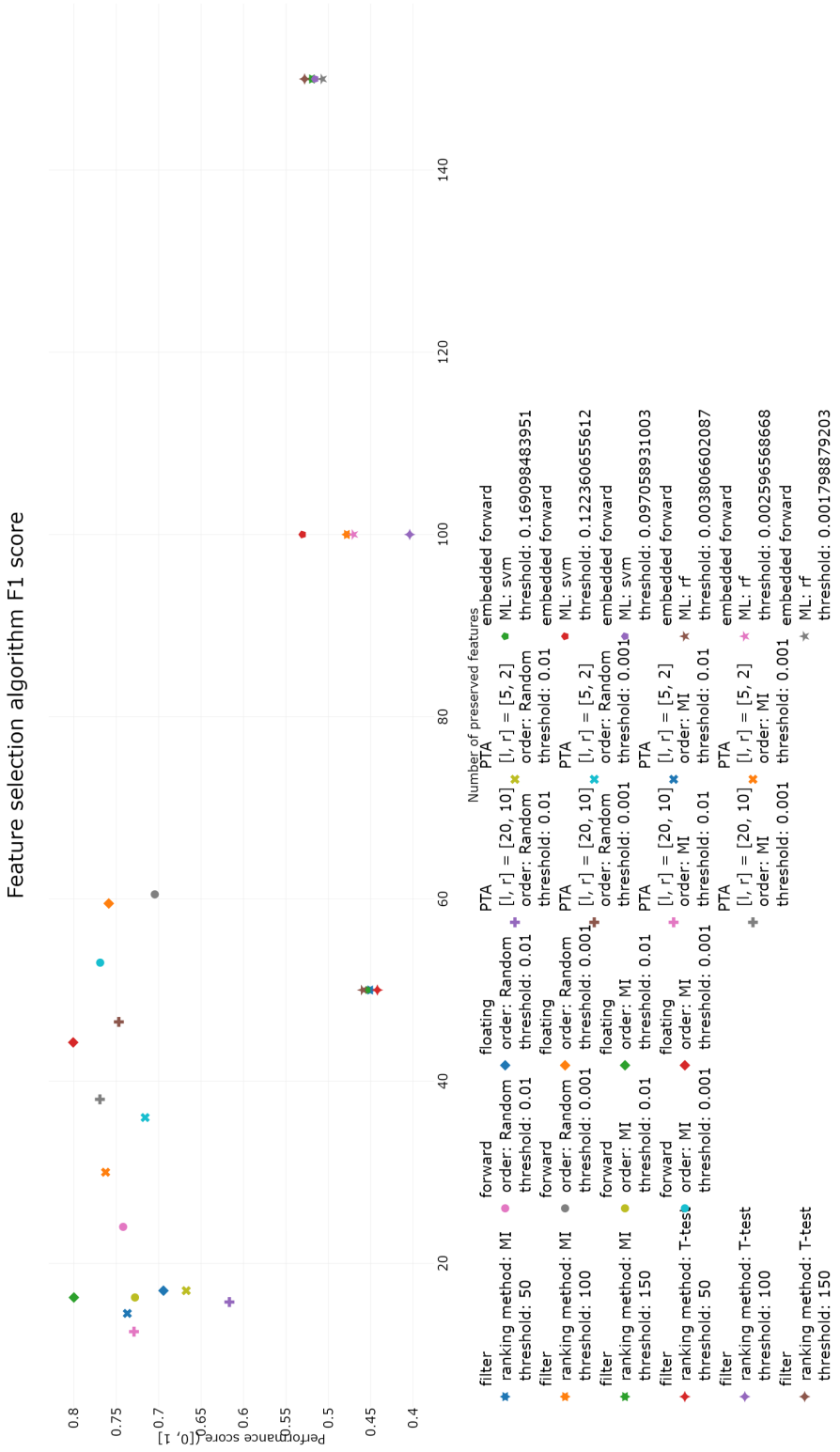

Figure 1.3: The F1 spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the F1 score of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

Table 1.6: 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

1.4.3 TPOT Feature Selection Integration Results

To visualize the results of the experiment, a table with the final results is made (Table 1.6). The optimization process is also recorded and shown in a plot for the four datasets (Figure 1.4). The five experiment reruns are averaged to one complete result.

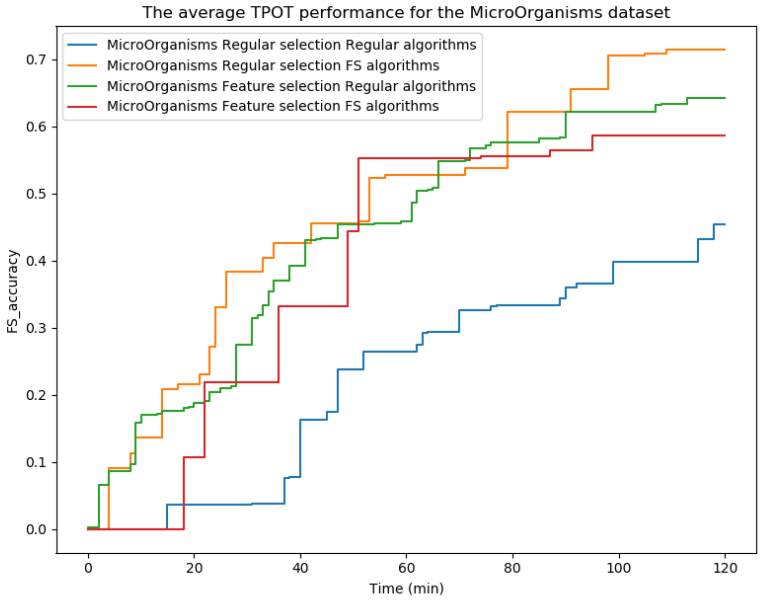

Figure 1.4: The optimization process for the different *TPOT* algorithms for the micro-organisms dataset. - TO DO: Make one for all 4 datasets

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.

1.5 Discussion

There are several works focused on the possibility of feature selection, both with a data analytical [7] as well as a biomedical [1, 2, 8] 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.

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.

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.

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.

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

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

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

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

[34] E. P. Xing, M. I. Jordan, R. M. Karp, *et al.*, "Feature selection for high-dimensional genomic microarray data," in *ICML*, vol. 1, pp. 601–608, Citeseer, 2001.

[35] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 1, pp. 1–14, 2013.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

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

[46] 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.

[47] 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.

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

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

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

[51] 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.

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

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

[54] 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.

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

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

[57] 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.

[58] 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.

[59] 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.

[60] 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.

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

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

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

[64] 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.

[65] 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.

[66] 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.

[67] 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.

[68] 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.

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

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