

# Feature Selection Analysis

*A case study for a computational biology framework*

T.P.A. BEISHUIZEN (0791613)  
Biomedical Engineering - Computational Biology  
Computer Science - Data Mining  
Eindhoven, University of Technology  
Email: `t.p.a.beishuizen@student.tue.nl`

September 10, 2018

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Background</b>	<b>3</b>
2.1	Datasets . . . . .	3
2.2	Feature Selection . . . . .	3
2.2.1	Filter Methods . . . . .	5
2.2.2	Wrapper Methods . . . . .	7
2.2.3	Embedded Methods . . . . .	11
2.3	Automated Machine Learning . . . . .	13
2.3.1	Meta-Learning . . . . .	14
2.3.2	Tree-based Pipeline Optimization Tool . . . . .	15
<b>3</b>	<b>Hypotheses</b>	<b>18</b>
<b>4</b>	<b>Methods</b>	<b>19</b>
4.1	Feature Selection Quality . . . . .	19
4.2	Feature Selection Exploration . . . . .	21
4.3	Feature Selection Algorithms Evaluation . . . . .	21
4.4	TPOT Feature selection integration . . . . .	23
<b>5</b>	<b>Results</b>	<b>26</b>
5.1	Feature Selection Exploration Results . . . . .	26
5.2	Feature Selection Algorithms Evaluation Results . . . . .	28
5.3	TPOT Feature Selection Integration Results . . . . .	37
<b>6</b>	<b>Discussion</b>	<b>38</b>
<b>7</b>	<b>Conclusions</b>	<b>38</b>
<b>A</b>	<b>Feature Selection Exploration Plots</b>	<b>44</b>
<b>B</b>	<b>Feature Selection Exploration Precision And Recall</b>	<b>48</b>

## 1 Introduction

Many biomedical datasets are created to use for expansion of biomedical knowledge and improvement of healthcare. Biomedical data is a generalizing term that describes multiple data types[1]. Examples of biomedical data are micro-array data[2], mass spectrometry data[3, 4] and nuclear magnetic resonance data[5], but also clinically derived data[6, 7]. From a bio-informatics perspective these biomedical data types vary significantly[1] and therefore extracting information out of biomedical data is not a trivial task. A framework for biomedical data analysis can help guiding biomedical engineers in their process of information extraction from their biomedical 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 focus on 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]. A framework that combines database integration, multiple research areas and multiple types of data analysis would be very beneficial for biomedical engineers, guiding them through their biomedical data analysis projects.

For such a framework dimensionality reduction is important. Some biomedical datasets contain a high number of features, whereas only a selection of those features are interesting. 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 for future research[21, 22, 23, 24] and tests on their performance[25, 26]. Basic feature selection algorithms should be present in the framework and are therefore tested on the available data sets. Therefore 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 we therefore present several of those algorithms and test their quality.

Four data sets were used as a case study for the feature selection algorithms. Two sets are micro-array datasets that are used for research on psoriasis[27, 28, 29, 30] and cancer[31]. Two other sets are mass spectrometry data sets, used for research on cancer[32] and micro organisms[33]. These four datasets all have a high number of features varying from 1000 to 54675 features with a number of samples varying from 200 to 580 samples. 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 on classification. Many features are expected to be irrelevant in this classification and therefore could be removed with feature selection.

## 2 Background

Before testing several feature selection methods, first the background of the study is explained. Firstly, the datasets used as a case study and their characteristics are briefly given. 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.

### 2.1 Datasets

Four datasets were used to test feature selection algorithms. The micro-array cancer dataset and both mass spectrometry datasets were found with the help of OpenML[34], whereas the psoriasis dataset was proposed by a project based on psoriasis[35]. All four of these datasets can be used for classification, as all of them were tested on two or multiple test subject groups. Also, all of them have a high number of features from which most are expected to be irrelevant, so classification would benefit from feature reduction. A schematic overview of the datasets is made (Table 1).

- *Psoriasis micro-array dataset*

This dataset is comprised of five different data sets[27, 28, 29, 30]. 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 (167 samples). Combining these three samples types gives 580 samples. Since the data comes from five different experiments, the data is normalized for every experiment.

- *Cancer micro-array dataset*

This dataset is used in a challenge focussing on classification problems with a low number of samples, but a high number of features.[31]. 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.

- *Cancer mass spectrometry dataset*

This dataset was created as a classification problem to distinguish cancer patterns from normal patterns[32]. 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[33]. 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[36]. The groups differ in size varying from 11 to 60 samples, making a total of 571 samples.

### 2.2 Feature Selection

Feature selection is a way to perform dimensionality reduction. In feature selection a subset of features is chosen to represent the complete sample space[37]. Several techniques are available to choose a representation subset and the effectiveness of these techniques is tested multiple

Table 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[27, 28, 29, 30] —
Cancer	Micro-array	54675	383	9	- Used in a data mining challenge[31]
Cancer	Mass Spectrometry	10000	200	2	- Created for the NIPS conference[32] - Several probe features are present —
Micro Organisms	Mass Spectrometry	1300	571	20	- Originates from a micro organisms study[33]

times[38, 39, 40]. These techniques can be grouped in accordingly in three different categories: filter methods, wrapper methods and embedded methods[41]. Each of these methods is explained in more detail.

Before actually performing feature selection, usually some kind of preparation or preparation is done. This preparation consists of making the data able for feature selection to be possible as well as preprocessing so all features are processed evenly. Common preprocessing steps include hot encoding categorical values and normalization of numerical values:

- *Hot encoding:*

Hot encoding is used when numerical, ordinal and categorical values are combined in the dataset. Whereas numerical and ordinal features have numbers as values, categorical features usually correspond to text with no clear order. Hot encoding is used to give numerical values to the 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 made possible between categories and they are treated evenly.

- *Normalization:*

The difference in value size between numerical features can differ greatly. For example, if both the length of a patient's body and tongue are measured, the value of body length is much bigger than the value for tongue 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 normalization is usually done in two steps for a feature with mean  $\mu_0$  and standard deviation  $\sigma_0$ : Step one is changing to a mean value  $\mu_1$  with value  $\mu_1 = 0$  by subtracting all values by the mean  $\mu_0$ . Step two is to divide all features by the standard deviation  $\sigma_0$ , so the new standard deviation  $\sigma_1$  becomes  $\sigma_1 = 1$ .

Since these filter methods can usually be best explained by showing example algorithms, several pseudo-algorithms were created for visual clarity. Variables used in these algorithms include:

- $F$ : A list that contains all the features. It is a list of size  $m$  with  $m$  being the number of features. A subset  $x$  of  $F$  is written as  $F_x$ .
- $X$ : 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.

- $y$ : A vector that contains all class labels for every sample. It is a vector of size  $n$  with  $n$  being the number of samples.
- $\alpha$ : A threshold value used for ranking and evaluation methods to find out whether a feature should be selected or not in feature selection.
- $W$ : Weights given to a feature after training a machine learning algorithm. The weight of feature  $f$  or set of features  $F_x$  is called  $W_f$  and  $W_{F_x}$  respectively.

### 2.2.1 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[42, 41]. 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 ( $X_{f_1}, X_{f_2} \dots X_{f_n}$ ) for computation. Secondly the columns are split by classes  $c_1, c_2 \dots c_n$  ( $X_f^{c_1}, X_f^{c_2} \dots X_f^{c_p}$ ) to distinguish the predictive power between feature for the 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*

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[43] (Table 2), based on computations on their mean and variance. This t-test gives a probability value (p-value) on interval  $[0, 1]$ , representing the chance the two groups originate from the same distribution. The p-value for the datasets is computed by creating a distribution for every class. Every feature is assigned a different p-value using the distributions for the groups. The t-test focuses on the difference between two groups and has different formulas to measure it: A paired t-test if the data can be paired between the two classes, an equal variance t-test and an unequal variance t-test for which it is assumed the variances of the two groups are equal or not equal respectively[43] (Table 3). For sample classifications, the groups would consist of samples with the same label. For example, the t-test can be done between group 1 with classification label "patient" and group 2 with label "normal". If the values of a certain feature from these groups are highly unlikely to follow the same distribution, the p-value will be very low and therefore the rank will be higher.

Table 2: T-test 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 3)[43]. This is calculated with the values for the mean  $\mu$ , significance level  $\alpha$ , average  $\bar{y}$ .

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

If there are more than two groups that must be checked whether they are from the same distribution, 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 Kruskal Wallis test

Table 3: The values of  $s_{\Delta\bar{y}}$  and  $t_{calc}$  for data sets with common unknown variance, uncommon unknown variance and paired data. [43]. This is calculated with the values for the average  $\bar{y}$ , variance  $s$ , population size  $n$  and average paired data difference  $\hat{d}$ .

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 $t_{calc} = \frac{\bar{d}}{s_{\bar{d}}}$

can be used if this reliability is needed. Another disadvantage is that t-test and ANOVA assumes the groups follow a certain distribution, which might not be the case[43]. Most analysis tools have a built-in package for statistics that include T-test and ANOVA. An example for this would be the SciPy[44] package for Python.

- *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 (Equation 1). In this equation the probability density function  $p$  is used to find the mutual information  $MI$  between variables  $x$  and  $y$ [45]. Mutual information can be used for both classification and regression. It was initially meant for using in communication channels, however its statistical decision making capabilities makes it a good filter method[46]. For mutual information packages are commonly available as well. An example would be the mutual information methods for discrete and continuous data in Scikit-Learn[47] for Python.

$$MI(x, y) = \int \int p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy \quad (1)$$

- *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. 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's correlation[48] (Equation 2). The correlation filter methods seem to be a good alternative for continuous data, but not for discrete data. With continuous data, the correlation between feature  $x$  and the output  $y$  is used to compute the ranks[49]. Also for correlation methods packages are commonly available. An example is Scikit-Learn[47] again for Python.

$$r(x, y) = \frac{\sum_i (x_i - \bar{x}_i)(y_i - \bar{y}_i)}{\sum_i (x_i - \bar{x}_i) \sum_i (y_i - \bar{y}_i)} \quad (2)$$

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[50] (Algorithm 2). For the t-test, a p-value of 0.05 is often chosen as an example[51, 52].

**Algorithm 1** A basic top  $n$  filter algorithm[42]

---

```

1: procedure FILTERSELECTION( $X, y, F, R, n$ )
2:    $F_{selected} \leftarrow \emptyset$  ▷ Start with empty feature set
3:    $Z \leftarrow \emptyset$  ▷ Start with empty set of ranking values
4:   for  $f$  in  $F$  do ▷ For all features in  $F$ 
5:      $Z_f \leftarrow R(X_f, y)$  ▷ Compute the ranking value between feature and output
6:      $Z \leftarrow Z \cup \{Z_f\}$  ▷ Add the ranking value to the set of ranking values
7:   end for
8:   Sort  $F$  by  $Z$  ▷ Sort the features by their ranking value
9:    $F_{selected} \leftarrow F_{[1,n]}$  ▷ Select the top  $n$  features from the sorted feature set
10:  return  $F_{selected}$ 
11: end procedure

```

---

**Algorithm 2** A basic filter algorithm[42]

---

```

1: procedure FILTERSELECTION( $X, y, F, R, \alpha$ )
2:    $F_{selected} \leftarrow \emptyset$  ▷ Start with empty feature set
3:   for  $f$  in  $F$  do ▷ For all features in  $F$ 
4:     if  $R(X_f, y) > \alpha$  then ▷ Check if ranking value is higher than threshold  $\alpha$ 
5:        $F_{selected} \leftarrow F_{selected} \cup \{f\}$  ▷ 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[42, 41].

### 2.2.2 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[53]. 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[54]. 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[53]. Evaluation functions can be based on conditional independence[53, 55], showing the difference in performance for a subset with and without a certain feature. Other evaluation functions are based on machine learning techniques[56, 41], rating the ability to classify the outcome. A good machine learning algorithm that can be used is Naive Bayes, as it is not very affected by conditional dependency[57]. Several interesting approaches for evaluation functions are shown in maximum relevance, minimum redundancy algorithms[58, 59, 60] (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).



- *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[53].

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. A layout of backward selection is shown as a pseudo algorithm (Algorithm 4) for understanding, as well[53].

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[53].

---

**Algorithm 3** A forward selection sequential search algorithm[53]

---

```

1: procedure SEQUENTIALFORWARDSELECTION( $X, y, F, J, \alpha, l$ )
2:    $F_{selected} \leftarrow \emptyset$  ▷ Start with empty feature set selection
3:   for  $f$  in  $F$  do ▷ For all features in  $F$ 
4:     if  $J(X_{F_{selected}}, y, X_f) > \alpha$  then ▷ Check if evaluation is higher than  $\alpha$  with feature  $f$ 
5:        $F_{selected} \leftarrow F_{selected} \cup \{f\}$  ▷ Add  $f$  to the feature set selection
6:     end if
7:     if  $\text{length}(F_{selected}) = l$  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 4** A backward selection sequential search algorithm[53]

---

```

1: procedure SEQUENTIALBACKWARDSELECTION( $X, y, F, J, \alpha, r$ )
2:    $F_{selected} \leftarrow F$  ▷ Start with complete feature set selection
3:   for  $f$  in  $F_{selected}$  do ▷ For all features in  $F_{selected}$ 
4:     if  $J(X_{F_{selected} \setminus \{f\}}, y, f) < \alpha$  then ▷ Check if evaluation is higher than  $\alpha$  without  $f$ 
5:        $F_{selected} \leftarrow F_{selected} \setminus \{f\}$  ▷ Remove  $f$  from the feature set selection
6:     end if
7:     if  $\text{length}(F \setminus F_{selected}) = r$  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, 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[53].

---

**Algorithm 5** A plus l-take away r sequential search algorithm[53]

---

```

1: procedure PTA( $X, y, F, J, \alpha, l, r$ )
2:    $F_{selected} \leftarrow \emptyset$  ▷ Start with empty feature set selection
3:    $i \leftarrow 0$  ▷ Initialize feature index
4:   while  $i < \text{length}(F)$  do ▷ Continue while not all features are evaluated
5:      $size \leftarrow \text{length}(F_{selected})$  ▷ Update size of feature set selection
6:     for  $k$  in  $[i, \text{length}(F)]$  do ▷ For features in  $F$  with index higher than  $i$ 
7:       if  $J(X_{F_{selected}}, y, X_{F_k}) > \alpha$  then ▷ Check if evaluation is higher than  $\alpha$  with  $f$ 
8:          $F_{selected} \leftarrow F_{selected} \cup F_k$  ▷ Add  $f$  to the feature set selection
9:       end if
10:       $i \leftarrow k$  ▷ Update feature index
11:      if  $size + \text{length}(F_{selected}) = l$  then ▷ Stop if feature addition limit has been reached
12:        break
13:      end if
14:    end for
15:     $size \leftarrow \text{length}(F_{selected})$  ▷ Update size of feature set selection
16:    for  $f$  in  $F_{selected}$  do ▷ For features in  $F_{selected}$ 
17:      if  $J(X_{F_{selected} \setminus \{f\}}, y, X_f) < \alpha$  then ▷ Check if evaluation is higher than  $\alpha$  without  $f$ 
18:         $F_{selected} \leftarrow F_{selected} \setminus \{f\}$  ▷ Remove  $f$  from the feature set selection
19:      end if
20:      if  $size - \text{length}(F_{selected}) = r$  then ▷ Stop if feature removal limit has been reached
21:        break
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[53].

- *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[61]. 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

---

**Algorithm 6** A floating search algorithm[53]

---

```

1: procedure FLOATINGSEARCHSELECTION( $X, y, F, J, \alpha$ )
2:    $F_{selected} \leftarrow \emptyset$  ▷ Start with empty feature set
3:   while  $F_{selected}$  changes do ▷ Continue while the feature set selection changes
4:     for  $f$  in  $F$  do ▷ For features in  $F$ 
5:       if  $f \notin F_{selected}$  and  $J(X_{F_{selected}}, y, X_f) > \alpha$  then ▷ Check if evaluation is higher
        than  $\alpha$  with feature  $f$ 
6:          $F_{selected} \leftarrow F_{selected} \cup \{f\}$  ▷ Add  $f$  to the feature set selection
7:       end if
8:     end for
9:     for  $f$  in  $F_{selected}$  do ▷ For features in  $F_{selected}$ 
10:      if  $J(X_{F_{selected} \setminus \{f\}}, y, X_f) < \alpha$  then ▷ Check if evaluation is higher than  $\alpha$  without
        feature  $f$ 
11:         $F_{selected} \leftarrow F_{selected} \setminus f$  ▷ Remove  $f$  from the feature set selection
12:      end if
13:    end for
14:  end while
15:  return  $F_{selected}$ 
16: end procedure

```

---

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.[53] A second example of a stochastic search algorithm is a genetic algorithm, that collects multiple subsets and mutates them in an evolutionary way[62].

**Algorithm 7** Simulated Annealing search algorithm[53]

---

```

1: procedure SA( $X, y, F, J, T_0, T_1, m, v$ )
2:    $F_{selected} \leftarrow \text{randomSubset}(F)$  ▷ Start with random feature subset
3:    $T \leftarrow T_0$  ▷ Initialize simulation temperature  $T$  with  $T_0$ 
4:   while  $T_0 \geq T_1$  do ▷ While temperature is not cold enough ( $T_1$ ), yet
5:      $i \leftarrow 0$  ▷ Initialize counter for not improvements
6:     while  $i < m$  do ▷ While not improvements is lower than maximum  $m$ 
7:        $f \leftarrow \text{randomFeature}(F)$  ▷ Select random feature  $f$ 
8:       if  $f \in F_{selected}$  then ▷ Check if  $f$  is in feature set selection
9:          $F_{candidate} \leftarrow F_{selected} \setminus \{f\}$  ▷ create candidate feature subset selection
10:         $\Delta = J(X_{F_{selected} \setminus \{f\}}, y, X_f)$  ▷ Calculate evaluation difference without feature  $f$ 
11:      else ▷ Check if  $f$  is not in feature set selection
12:         $F_{candidate} \leftarrow F_{selected} \cup \{f\}$  ▷ create candidate feature subset selection
13:         $\Delta = J(X_{F_{selected} \cup \{f\}}, y, X_f)$  ▷ Calculate evaluation difference with feature  $f$ 
14:      end if
15:      if  $\Delta > 0$  then ▷ Check if change is positive
16:         $F_{selected} \leftarrow F_{candidate}$  ▷ Use candidate as feature subset selection
17:      else
18:         $r \leftarrow \text{randomReal}(0, 1)$  ▷ Get a random value  $r \in [0, 1]$ 
19:        if  $r < \exp(-\Delta/T)$  then ▷ Check if a random change should occur
20:           $F_{selected} \leftarrow F_{candidate}$  ▷ Use candidate as feature subset selection
21:        end if
22:         $i \leftarrow i + 1$  ▷ Increment counter for no improvement
23:      end if
24:    end while
25:     $T \leftarrow T \times v$  ▷ Lower the temperature with  $v \in [0, 1]$ 
26:  end while
27:  return  $F_{selected}$ 
28: end procedure

```

---

The major advantage of using wrapper methods is that it also takes 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 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 overfitting[53, 41].

### 2.2.3 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 the performance of feature subsets. Machine learning however has multiple attributes that can be used directly to select or eliminate features from the ideal feature set selection. Embedded methods combine feature selection with a machine learning algorithm  $M$ , in contrast of wrapper methods in which these are separated[63]. Usually this combination involves using the weights given to features by machine learning algorithms[64].

Embedded methods usually solve feature selection by using one of two possible solution. The first solution is based on contribution relaxation minimization and the second solution 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 exemplary embedded methods[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 a feature selection 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. 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 take into account the dependency between features[63].

---

**Algorithm 8** An embedded forward selection algorithm[63]

---

```

1: procedure EMBEDDEDFORWARDSELECTION( $X, y, F, M, \alpha$ )
2:    $F_{selected} \leftarrow \emptyset$  ▷ Start with empty feature set
3:    $W \leftarrow M(X, y)$  ▷ Extract the weights for every feature
4:   for  $f$  in  $F$  do ▷ For all features in  $F$ 
5:     if  $W_f > \alpha$  then ▷ Check if weight is higher than threshold  $\alpha$ 
6:        $F_{selected} \leftarrow F_{selected} \cup \{f\}$  ▷ 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 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[63].

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 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[63].

**Algorithm 9** An embedded backward elimination algorithm[63]

---

```

1: procedure RECURSIVEBACKWARDELIMINATION( $X, y, F, M, \alpha$ )
2:    $F_{selected} \leftarrow F$  ▷ Start with complete feature set selection
3:   while  $F_{selected}$  changes do ▷ While the feature set selection changes
4:      $W_{F_{selected}} \leftarrow M(X_{F_{selected}}, y)$  ▷ Extract the weights for every feature
5:      $f_{min} \leftarrow f \in F_{selected}$  with  $W_f = \min(W_{F_{selected}})$  ▷ Find the feature with the lowest
weight
6:     if  $W_{f_{min}} < \alpha$  then ▷ Check if weight is lower than threshold  $\alpha$ 
7:        $F_{selected} \leftarrow F_{selected} \setminus \{f\}$  ▷ Remove  $f$  from selected features
8:     end if
9:   end while
10:  return  $F_{selected}$ 
11: 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 support vector machines (SVM). The weights given by SVM give the features a contribution factor in the outcome[65, 66, 67, 68]. 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[69, 70, 42]. To overcome an overfitting problem, commonly occurring in decision trees, random forests can be used instead and the best splitting features can be used[71].

### 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)[72]. After some time, when preprocessing was added in the mix as well, the name autoML was being used[73]. 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[72]. An upgrade was published that added regression and parallelism[74].

As explained shortly before, to go from data and 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[73].

AutoML tries to find the best machine learning pipelines to compute which algorithms must be selected, combined with tuning the hyper-parameters and preprocessing. This algorithm selection usually is done in a meta-learning approach, which focuses on finding how the machine learning algorithms perform for a task interval. 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 (Subsection 2.3.1). Aside from that, a tool that makes use of AutoML, TPOT, is also discussed (Subsection 2.3.2).

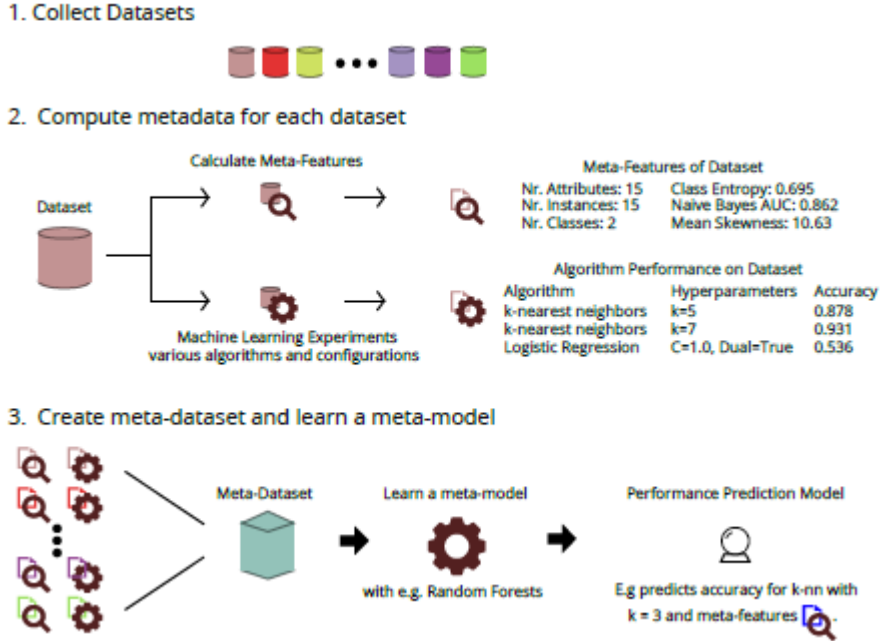


Figure 1: A layout of how meta-learning works. 1. Data sets are collected. 2. Meta-data is computed for each dataset. 3. A meta-dataset is created and a meta-model is learned[73].

### 2.3.1 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 ??). It tries to link the datasets with a suitable algorithm for high performance and tries to improve this performance by finetuning 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 types of these meta-features. The first type is simple, statistical and information-theoretic. 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 type 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[75, 76].

For using those meta-features in picking the best machine learning algorithm meta-learners can be used. Meta-learners are algorithms that choose between the possible choices. 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[77].

Table 4: All TPOT Algorithms used for the TPOT classifier function. Not only the algorithms differ, but also the hyper-parameters within these algorithms.

Algorithm type	Specification	Algorithms
<b>Classifier</b>	Naïve Bayes	GaussianNB, BernoulliNB, MultinomialNB
	Decision Tree	DecisionTree, ExtraTrees, RandomForest, GradientBoosting
	Nearest Neighbor	KNeighbors
	Support Vector Machines	LinearSVC
	Logistic Regression	Logistic Regression
<b>Preprocessors</b>	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

### 2.3.2 Tree-based Pipeline Optimization Tool

A tool that implements autoML is tree-based pipeline optimization tool (TPOT). It uses the machine learning pipelines and evolutionary optimization to find the best solution for every data set. This evolutionary optimization is done by genetic programming. Genetic programming evolves 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.

TPOT makes use of this genetic programming with using the machine learning pipelines in a tree (Figure 2). TPOT consists of preprocessing and machine learning algorithms, that form the backbone of the pipelines (Table 4). Their hyper-parameters are the variables. TPOT makes mostly use of the machine learning and preprocessing algorithms of skikit-learn from Python in which it is also written, but also several additional algorithms are present (e.g. a hot encoding algorithm).

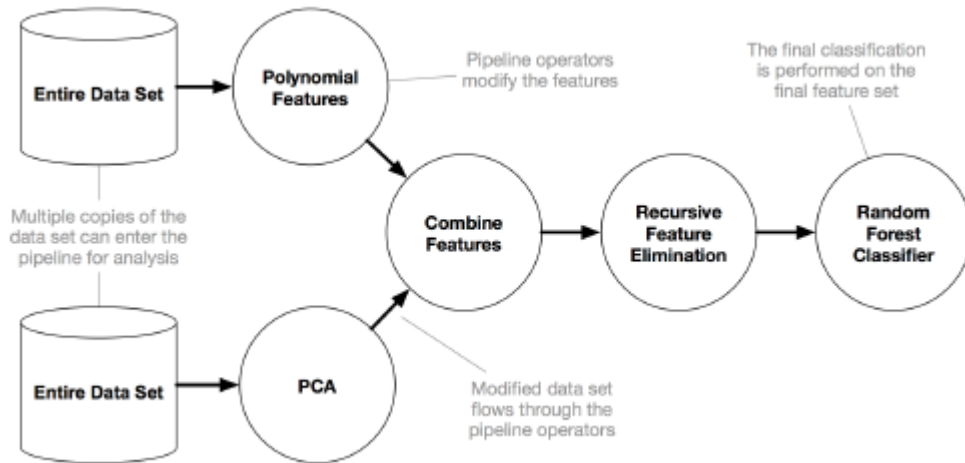


Figure 2: An example of a machine learning pipeline in TPOT. It only shows the primitive algorithms and not hyperparameter terminals. At the root is the machine learning algorithm[73].



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 3)

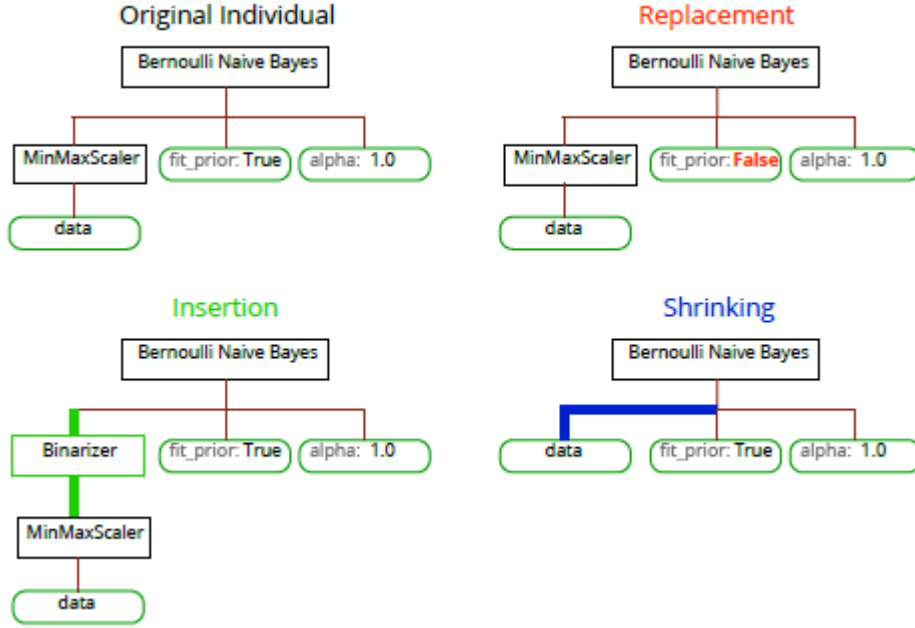


Figure 3: Examples of the three mutations in the TPOT algorithm: insertion, replacement and shrinking[73].

TPOT also focuses on mutations between two pipelines through the means of crossovers. Between two pipelines, sub-trees and primitives can be changed, given that the both pipelines remain valid (Figure 4). Every time a crossover is performed, two separate pipelines are used and changed, creating two new ones.

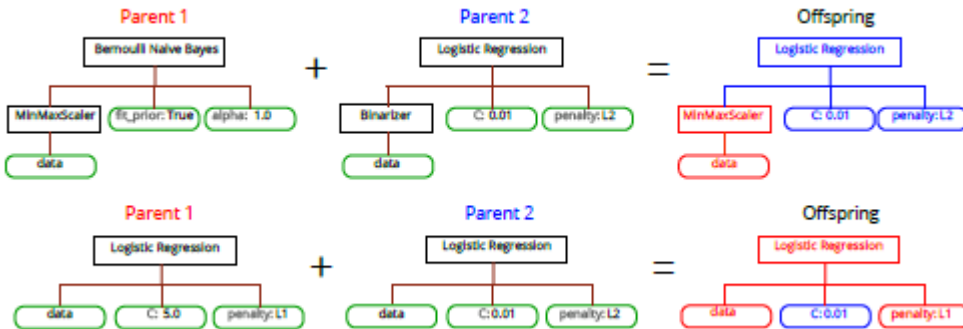


Figure 4: An example of a TPOT crossover[73].

Comparing the possibilities from TPOT and the challenges in biomedical data, it seems that TPOT has some implementations to tackle them. It has several different scalers (StandardScaler, RobustScaler, MinMaxScaler) to tackle feature heterogeneity between different data sets and er-

rors. It also has some feature selection operators to tackle errors (VarianceThreshold, SelectKBest, SelectPercentile). For missing values, the algorithm imputes the median as estimation for the missing value.

### 3 Hypotheses

The main goal for this project 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 are discussed here:

#### 1. Optimum in number of selected features

An optimum should be found in the number of 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[78], 500 to 13 and for even bigger datasets from 5000-100000 features to between 200 and 900 relevant features[79]. Interestingly, the number of relevant features was not related to the original feature set size, indicating the optimum to be an absolute number. The value of this absolute optimum is estimated to be lower than 1000, as no study before ever used a lower number of relevant features.

#### 2. Best feature selection methods

Three types of feature selection methods are discussed, all of them having a 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 more close to trying to find the optimal solution the exhaustive way and therefore are expected to perform the best feature selection at a 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 to find the results and those machine learning methods somewhat 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.

#### 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 an 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.

## 4 Methods

For empirical evaluation the collection of feature selection methods experiments are done. The quality of the feature selection is tested by creating 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 tests are done in Python, using the Anaconda distribution, as with Anaconda external packages can be used quickly. Methods that can be used for feature selection are from the packages scikit-learn[47], SciPy[44] and NumPy[80]. The four example datasets (subsection 2.1) were used as exemplary datasets for the experiments.

### 4.1 Feature Selection Quality

To find out the quality of the feature selection, multiple machine learning algorithms are used[81]. 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 overfitting 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 leave one out (scikit-learn) 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 accuracy in those cases is not sufficient enough to evaluate the result. To incorporate the quality of feature selection a new modified quality term FS\_accuracy is created (Equation 4.1). In FS\_accuracy a modified version of the original accuracy is given by first multiplying it by a factor dependent on the number of features. This factor consists of a constant  $\alpha$ , a value in range  $[0, 1]$  which can be chosen for the influence of the value that represent the number of features. In practice  $\alpha > 0.95$  so the reduction does not have too much influence on the accuracy. An example plot of how the correction factor changes is given (Figure 5).

$$\text{FS\_accuracy} = \text{accuracy} \times \alpha^{\# \text{features}}$$

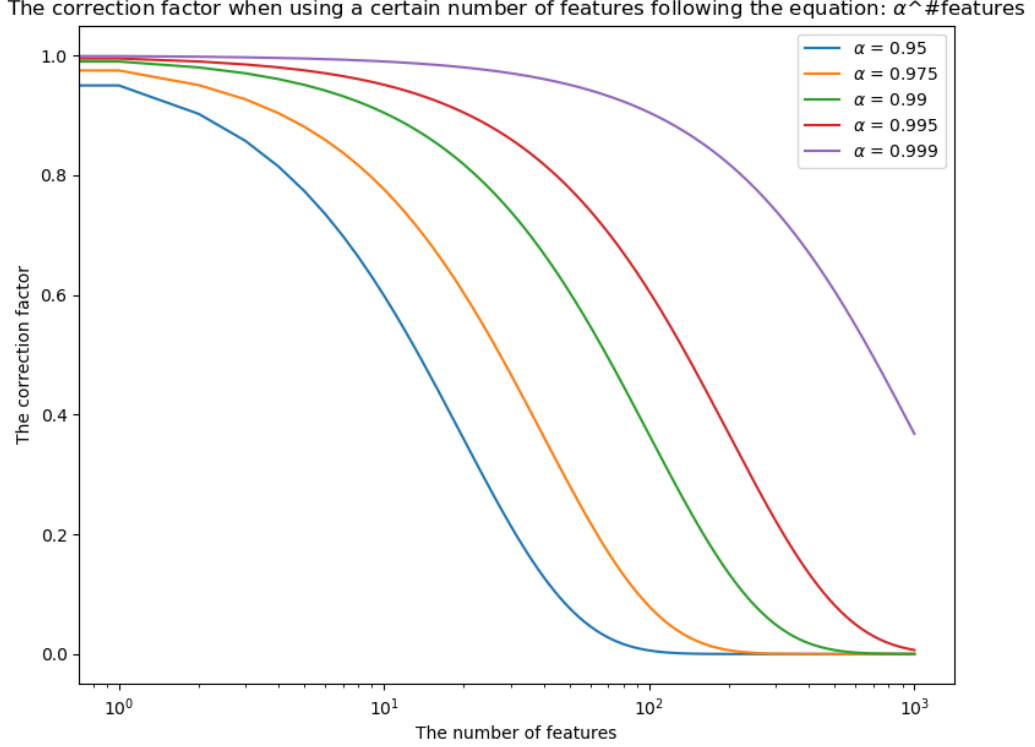


Figure 5: The correction factor for the FS\_accuracy from Equation 4.1. Every graph corresponds to a different value of  $\alpha$ .

The factor  $\#features$  is an absolute count, instead of a relative count (for example a percentage of features). The goal of data analysis is the main reason for using an absolute count. 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 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 lets you more actively choose for the input size.

To put the impact of the FS\_accuracy in perspective, an exemplary figure is made how the outcome is computed (Figure 6). This figure shows the impact on the performance of a method when using a certain number of features. After using a correction factor, a new optimum is created for which the number of features is important. The example (with  $\alpha = 0.99$ ) seems to give a good indication of the desired outcome. For every 10 features, the FS\_accuracy is reduced by about 10% (Figure 5), which meant for the exemplary filter method that for 50 features an optimum was reached (Figure 6). Because of this empirically found desired trade-off,  $\alpha = 0.99$  will be chosen in this project when using FS\_accuracy.

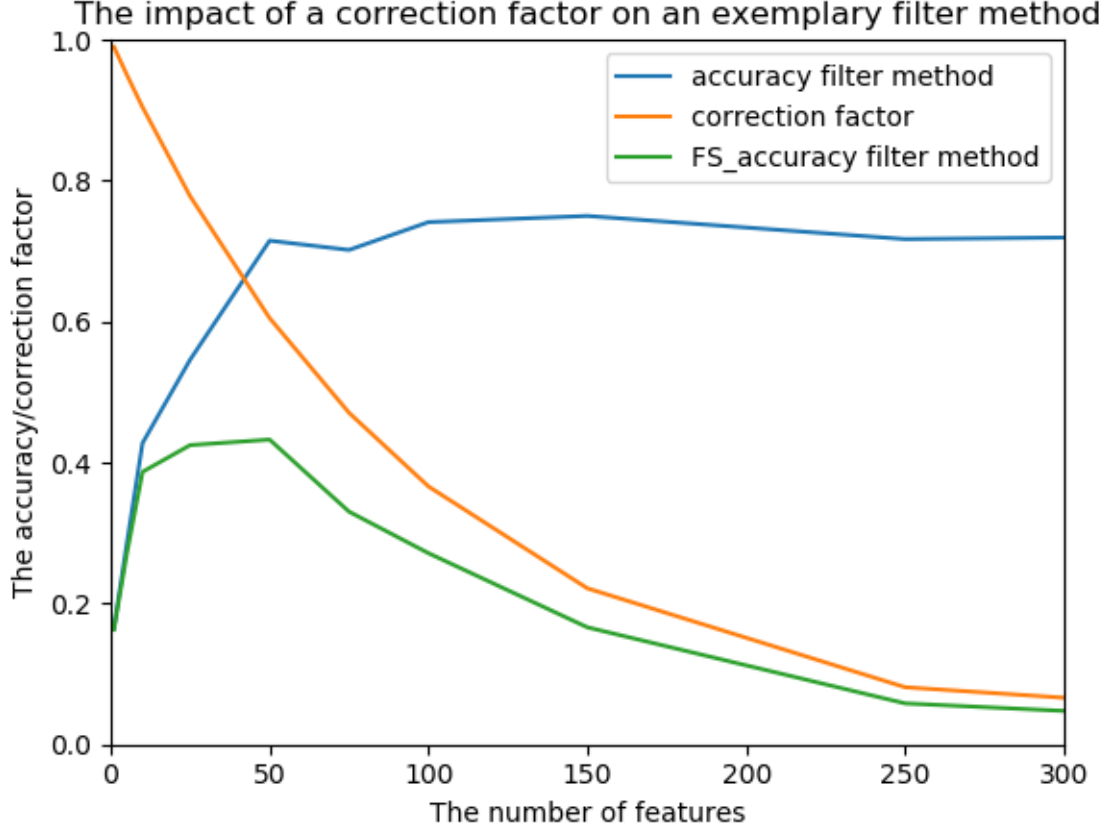


Figure 6: An example of the impact of the correction factor on the accuracy. The shown correction factor uses  $\alpha = 0.99$ . On the x-axis the number of features is shown and on the y-axis the value for the original accuracy, the correction factor and the FS\_accuracy.

## 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 5). 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 means 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 are conducted. 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.

## 4.3 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. The average performance for all datasets is also computed for clarification

Table 5: The four meta-parameters with their possible values in the first experiment.

Variable	Description	Values
Dataset	The datasets used (subsection 2.1)	Psoriasis RSCTC Arcene Micro-Organisms –
Ranking method	The method used for ranking the features (subsection 2.2.1)	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 4.1)	Naive Bayes Logistic Regression Support Vector Machine Decision Tree Nearest Neighbours

purposes. An overview of feature selection methods is made (Table 6) 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 wrapper methods was chosen to be Random (no special ordering) and Mutual Information (special ordering). The evaluation was chosen to be Naive Bayes as it is simple and not much affected by conditional dependency. The  $\alpha$  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 newly proposed FS-accuracy and the computation time is shown as well for  $\alpha = 0.99$  (Subsection 4.1) to show the relation between computation time on one hand and the FS-accuracy on the other hand. Several discussed algorithms are chosen to be excluded from the evaluation.

Table 6: 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
	Forward selection (Algorithm 8)	- Machine Learning: SVM, RandomForests - Threshold: 50, 100, 150 features
Embedded methods	Forward selection (Algorithm 8)	- Machine Learning: SVM, RandomForests - Threshold: 50, 100, 150 features

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 (Section 2.2).

#### 4.4 TPOT Feature selection integration

TPOT is an effective tool to find the best machine learning pipeline for a certain dataset (Subsection 2.3.2). 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*

TPOT seemed to have an issue inserting new algorithms. This issue 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 identical 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\_accuracy (Subsection 4.1). Using this FS\_accuracy 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 with regards to 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 (Subsection 2.2.1) and embedded (subsection 2.2.3) are present. Both of these select percentages of feature selection, though, which can still result in a high number of features. On top of that, no wrapper methods (Subsection 2.2.2) 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 start the original population with guaranteed 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\_accuracy makes TPOT search for solutions with fewer features, the removal of algorithms that cause errors and memory issues creates better optimization and the more effective use of the insert evolution, makes the algorithm search that possibility better. 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 7). All four datasets are tested (Subsection 2.1) and the accuracy is changed to the feature sensitive FS\_accuracy with  $\alpha = 0.99$  (Equation 4.1). For testing TPOT an optimization time of 720 minutes (12 hours) was chosen as a reasonable time constraint (one night), an algorithm was not allowed to run for longer than ten minutes, a population size of 20 was chosen to not be too selective at the start and a training set size of 0.90 which is a general training

Table 7: The experiment details for testing the non-trivial changes in TPOT.

Experiment factors	Detailed values	Remarks
<b>Datasets</b>	MicroOrganisms Arcene RSCTC Psoriasis	High number of features (Subsection 2.1)
<b>Performance measurement</b>	One type: FS_accuracy - $\alpha = 0.99$	Addition of correction factor (Equation 4.1)
<b>TPOT input parameters</b>	One set of input values: - max. optimization time = 720 min - max. alg. evaluation time = 10 min - population size = 20 - train size = 0.9	Explanation of input values: - The time one run should take (12 hours) - The evaluation time of one pipeline - The number of pipelines in one generation - The number of samples used for training
<b>Pipeline selection</b>	Regular selection Feature selection focused	Possible obligatory addition of a feature selection algorithm
<b>Change in feature selection algorithm set</b>	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

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  $4(\text{datasets}) \times 2(\text{pipeline selection}) \times 2(\text{feature selection set}) = 16$  different experiments.

## 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.

### 5.1 Feature Selection Exploration Results

The results were plotted for every dataset separately (Appendix A). An average of the four datasets was also created to show the difference between validation and test score (Figure 7), between machine learning quality measures (Figure 8) and between data sets with ranking methods (Figure 9).

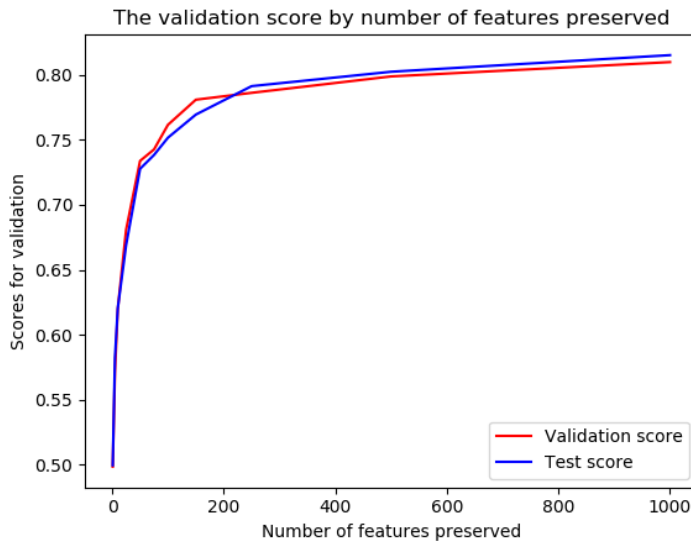


Figure 7: The average validation and test scores after averaging the scores for the data sets and ranking methods.

As could be seen (Figure 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 overfitting 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 8 and 9).

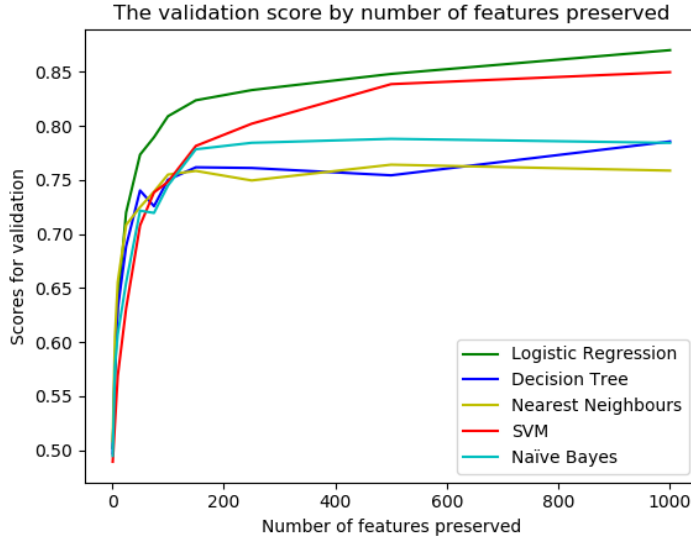


Figure 8: The average validation scores shown per machine learning quality measurement.

Logistic regression gave the best validation scores taking all data sets combined, followed by SVM. Also, when looking at all eight plots separately (Appendix A), 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'.

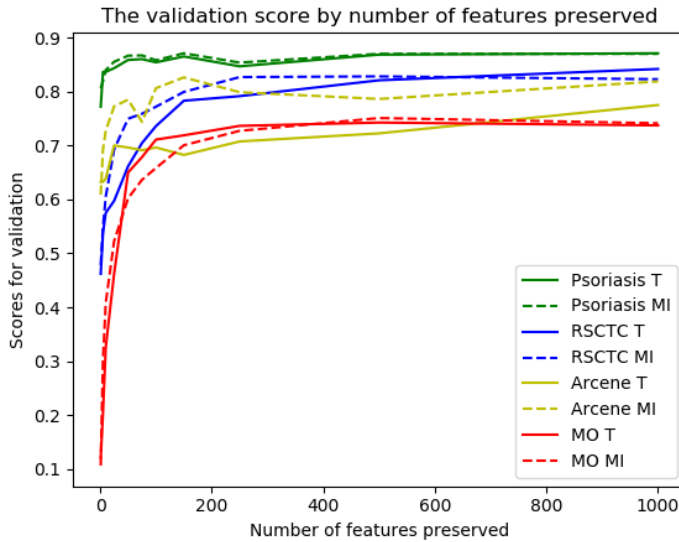


Figure 9: The average validation scores shown per dataset and rank.

A difference in score quality for the datasets was visible (Figure 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 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 (Figure 10 and Appendix B). These scores showed that the accuracy is a plausible way to depict quality. Only the Arcene dataset shows some difference between accuracy, precision, recall and the F1-score but these differences would not affect any conclusions.

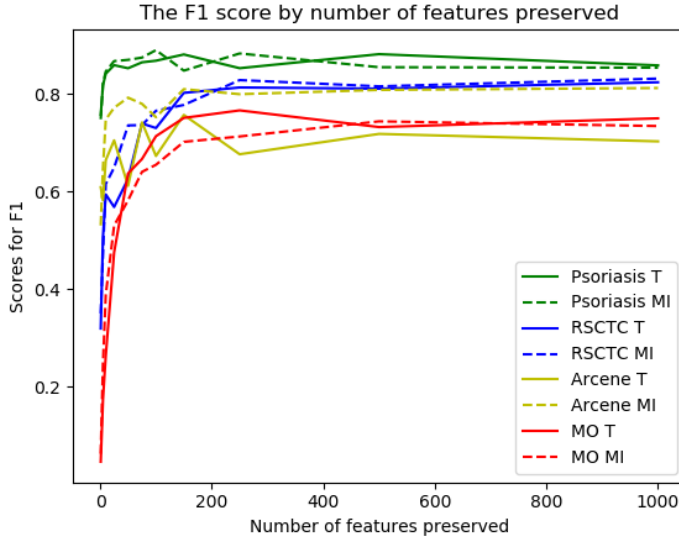
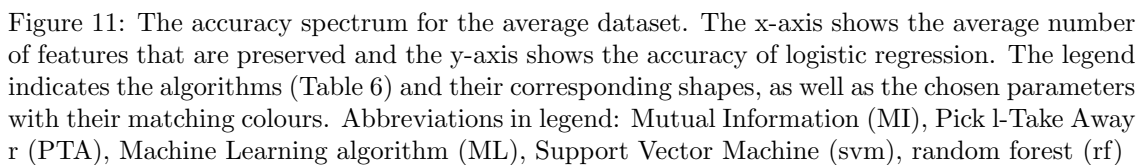


Figure 10: The average F1-scores shown per dataset and rank.

## 5.2 Feature Selection Algorithms Evaluation Results

Spectra using all basic filter methods and wrapper sequential search methods were created, averaging the accuracy (Figure 11), precision (Figure 12), recall (Figure 13) and F1 score (Figure 14) 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.



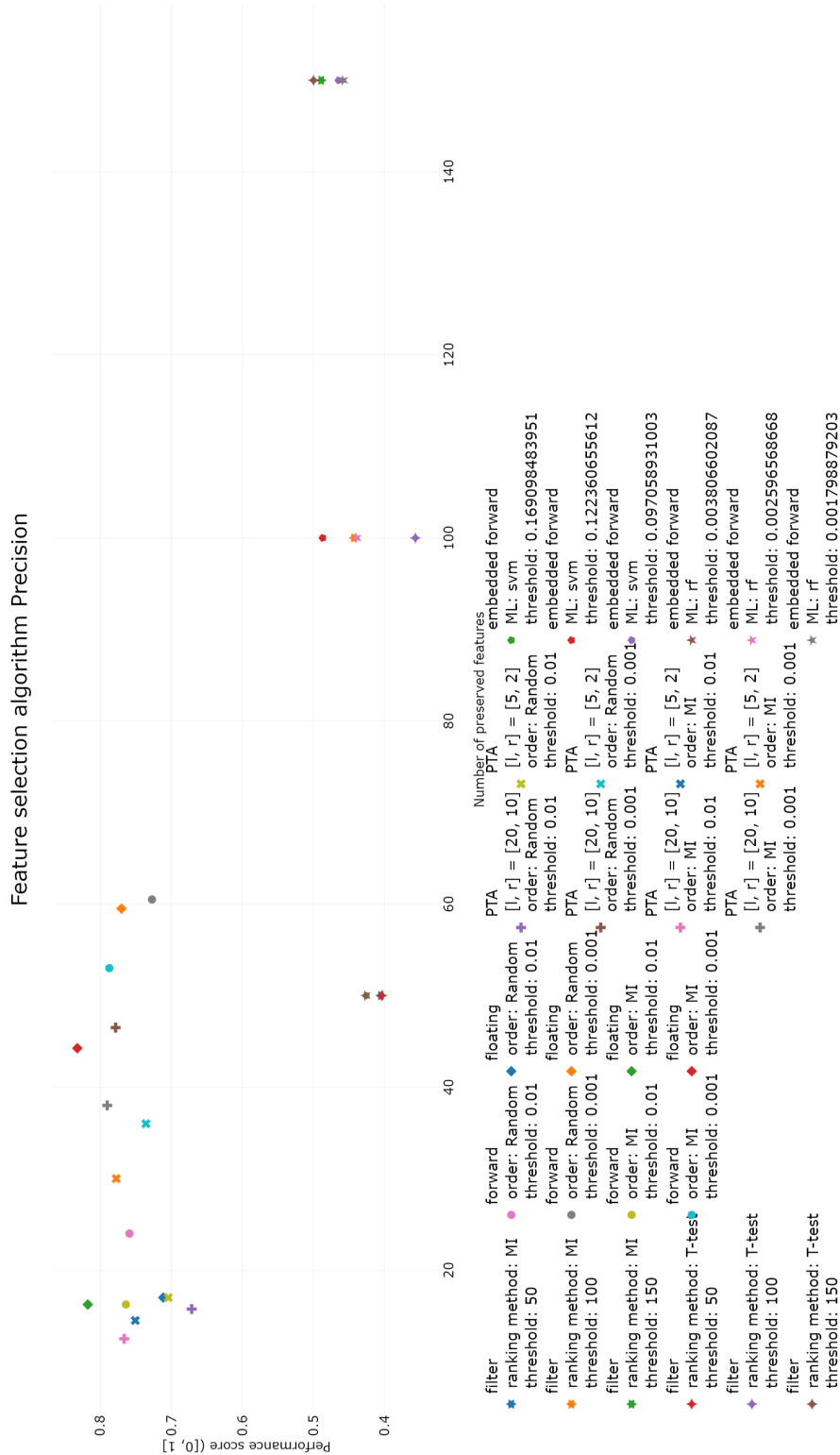


Figure 12: The precision spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the precision of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

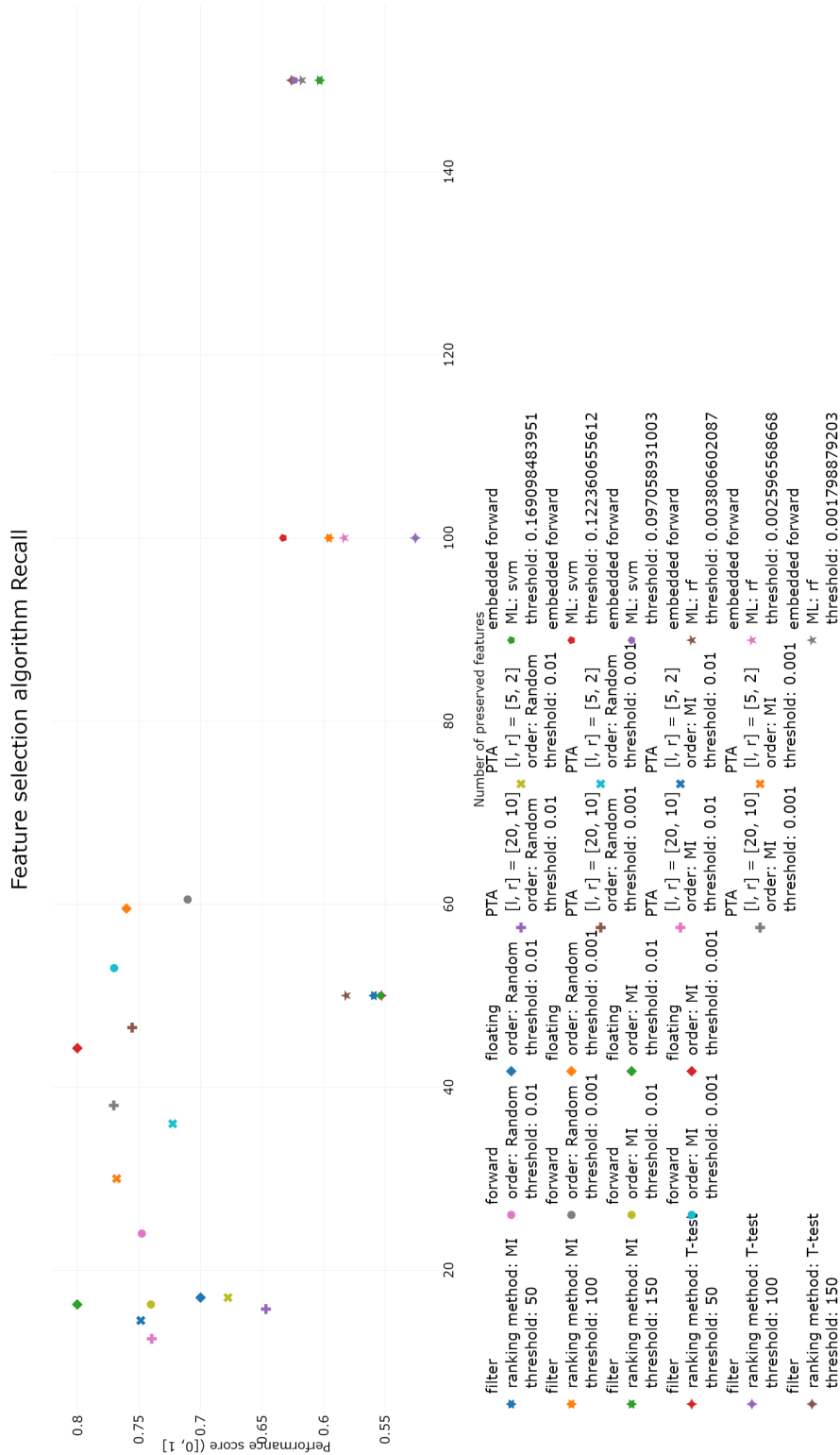


Figure 13: The recall spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the recall of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. Abbreviations in legend: Mutual Information (MI), Pick l-Take Away r (PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)



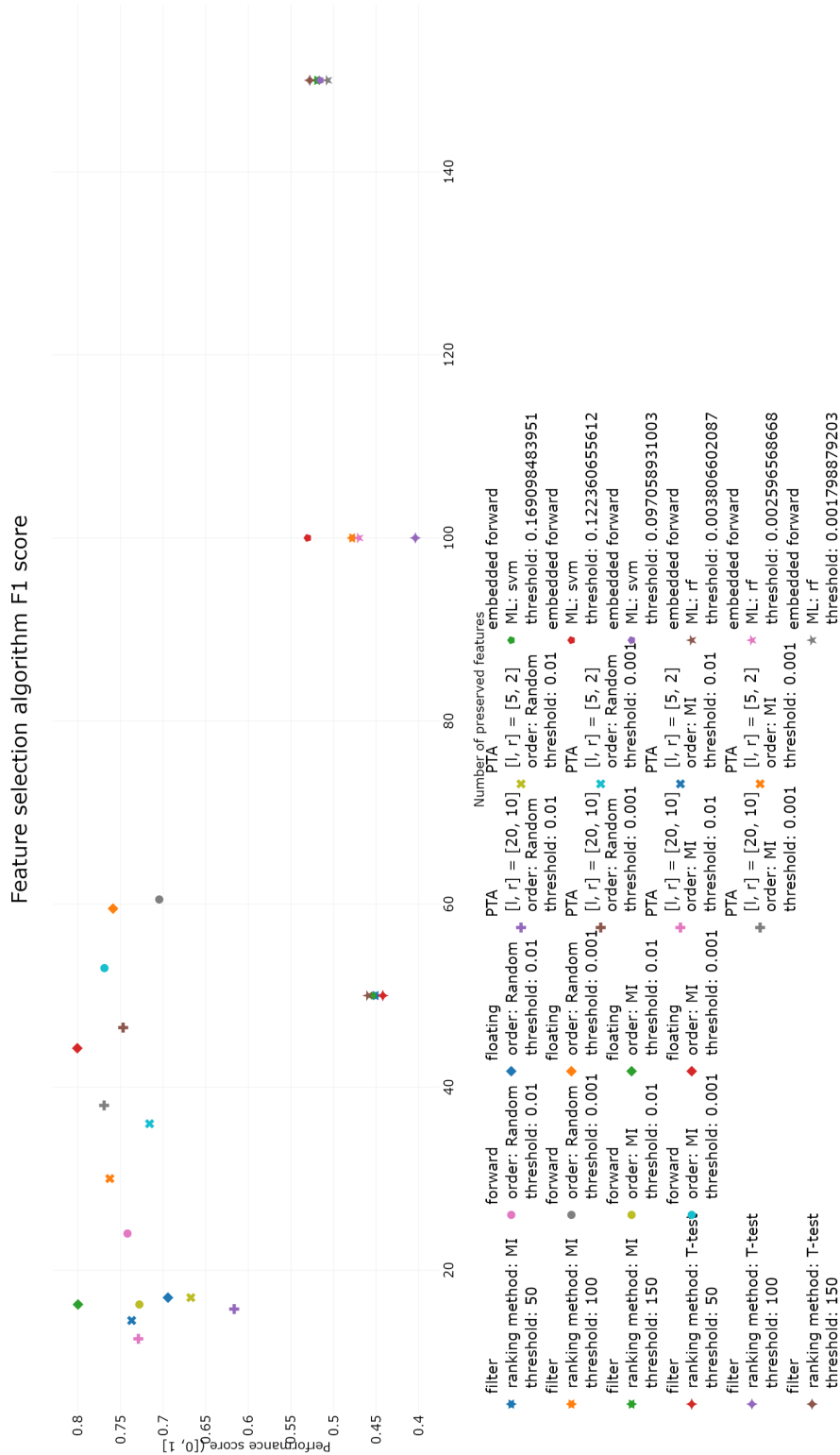


Figure 14: The F1 spectrum for the average dataset. The x-axis shows the average number of features that are preserved and the y-axis shows the F1 score of logistic regression. The legend indicates the algorithms and their corresponding shapes, as well as the chosen parameters with their matching colours. 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 15). This chart immediately showed that the wrapper methods took significantly more computation time, with floating search to be take the longest. There was no major difference between forward selection and both PTA's, 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.

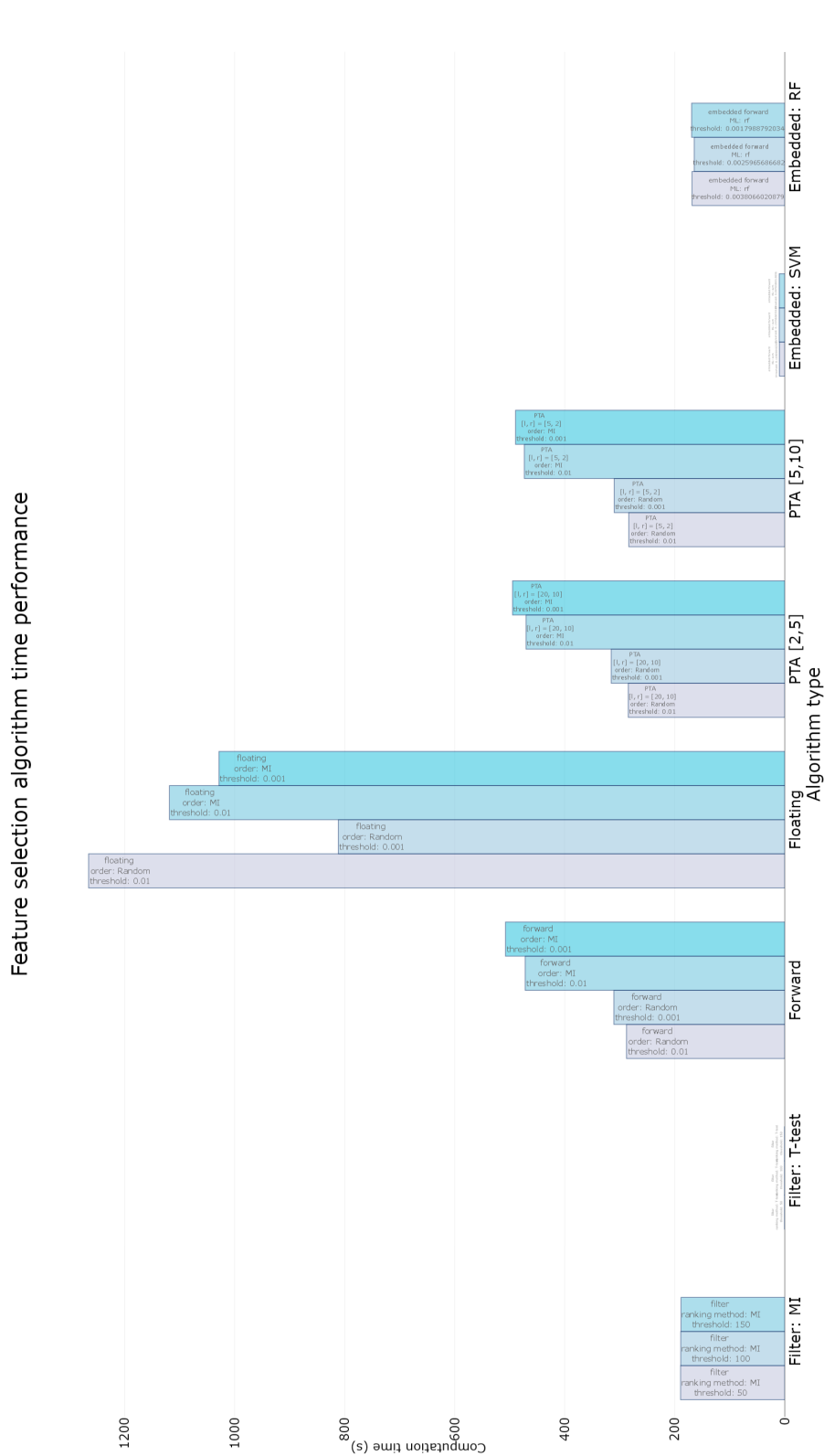
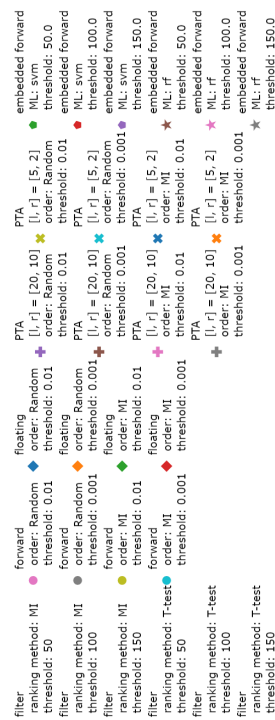


Figure 15: A bar chart showing the average computation time of all evaluated algorithms. The x-axis corresponded to the different algorithms, for which the computation time for multiple parameter combinations were given. The y-axis corresponded to the computation time in seconds. Abbreviations in legend: 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\_accuracy, the corrected accuracy, are visualized (Figure 16). 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 takes 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\_accuracy.
- For all wrapper methods the addition of an ordering and choosing a threshold of  $\alpha = 0.01$  over  $\alpha = 0.001$  gives better results. Using these hyper-parameters however usually add computation time.
- Forward selection and PTA do not show a big difference between each other.



(PTA), Machine Learning algorithm (ML), Support Vector Machine (svm), random forest (rf)

<b>Algorithm</b>	selection availability	regular selection		always feature selection	
		regular algorithms	feature selection algorithms	regular algorithms	feature selection algorithms
<b>Datasets</b>	MicroOrganisms	0.79	0.65	0.89	0.85
	Arcene	0.89	0.89	0.88	0.84
	RSCTC	0	0.55	0.91	0.60
	Psoriasis	0.78	0.77	0.91	0.72

### 5.3 TPOT Feature Selection Integration Results

## 6 Discussion

Several scientist researched the possibility of feature selection, both with a data analytical[25] as well as a biomedical[19, 20, 26] point of view. These research projects usually focused on datasets with a low number of features, for which feature selection is less relevant. Doing 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.

## 7 Conclusions

After evaluation of the results in the first experiment set-up, it can be concluded that there is not a big difference between using T-test/ANOVA or Mutual Information as a ranking method. Both seem to give similar results, with the exception of one dataset (Figure 9). The T-test/ANOVA is computationally faster than Mutual Information as could be seen in the second experiment (Figure 15), however also works with the assumption that the classes are normally distributed. This may not always be the case. A rule of thumb for choice of ranking method would be to use t-test, except for when the data is not normally distributed. Both of them should definitely be considered to be used in the framework.

A second conclusion can be drawn from looking at the accuracy with the number of features preserved. After a threshold of 200 features, additional features do not raise the validation score as much as the first 200 features seem to do. This can indicate a second rule of thumb, that at least 200 features must be preserved after using a filter method. This rule of thumb is according to the hypothesized notion that at most 1000 features should be preserved to still show an efficient result, however is much lower than the expected than the upper bound.

After evaluation of both filter, wrapper and embedded methods, wrapper methods seem to be significantly better at selecting a smaller fraction of features while preserving a similar test score. Whereas the expectation was for 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 is not the case however as embedded methods had a near identical quality to filter methods and performed much worse than wrapper methods. The outcome indicates that wrapper methods are significantly different in results and filter and embedded methods being near identical.

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

There is 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 is recommended for 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  $\alpha = 0.01$  gives a smaller feature subset at the cost of a lower classification score. If a smaller subset of high influence features is needed, a bigger threshold should be chosen, whereas it will be smaller if the quality of the feature subset is better.

In this research 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 into in future research.

## References

- [1] N. Gehlenborg, S. I. O'donoghue, N. S. Baliga, A. Goesmann, M. A. Hibbs, H. Kitano, O. Kohlbacher, H. Neuweiler, 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] D. Capitani, A. P. Sobolev, and L. Mannina, "Nuclear magnetic resonance–metabolomics," *Food Authentication: Management, Analysis and Regulation*, p. 177, 2017.
- [6] 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.
- [7] D. F. Sittig, A. Wright, J. A. Osherooff, 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.
- [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. 11, 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] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] 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- $\kappa$ b pathways,” *Nature genetics*, vol. 41, no. 2, pp. 199–204, 2009.

- [28] 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.
- [29] 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.
- [30] Y. Yao, L. Richman, C. Morehouse, M. De Los Reyes, B. W. Higgs, A. Boutrín, 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.
- [31] 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.
- [32] 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.
- [33] 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.
- [34] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, “Openml: Networked science in machine learning,” *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.
- [35] Z. Félix Garza, J. Liebmann, M. Born, P. Hilbers, and N. van Riel, “A dynamic model for prediction of psoriasis management by blue light irradiation,” 2017.
- [36] M. T. Madigan, J. M. Martinko, J. Parker, *et al.*, *Brock biology of microorganisms*, vol. 13. Pearson, 2017.
- [37] I. Guyon and A. Elisseeff, *An Introduction to Feature Extraction*, pp. 1–25. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
- [38] 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.
- [39] 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.
- [40] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Computers & Electrical Engineering*, vol. 40, no. 1, pp. 16–28, 2014.
- [41] Y. Saeys, I. Inza, and P. Larrañaga, “A review of feature selection techniques in bioinformatics,” *bioinformatics*, vol. 23, no. 19, pp. 2507–2517, 2007.
- [42] W. Duch, *Filter Methods*, pp. 89–117. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
- [43] R. M. Heiberger and B. Holland, *Statistical analysis and data display*. Springer, 2004.
- [44] E. Jones, T. Oliphant, and P. Peterson, “{SciPy}: open source scientific tools for {Python},” 2014.

- [45] 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.
- [46] 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.
- [47] 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.
- [48] 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.
- [49] M. A. Hall, "Correlation-based feature selection of discrete and numeric class machine learning," 2000.
- [50] 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.
- [51] 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.
- [52] 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.
- [53] J. Reunanen, *Search Strategies*, pp. 119–136. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
- [54] B. Alsallakh and L. Ren, "Powerset: A comprehensive visualization of set intersections," vol. 23, pp. 1–1, 01 2016.
- [55] I. Tsamardinos, G. Borboudakis, P. Katsogridakis, P. Pratikakis, and V. Christophides, "Massively-parallel feature selection for big data," *arXiv preprint arXiv:1708.07178*, 2017.
- [56] 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.
- [57] H. Zhang, "The optimality of naive bayes," *AA*, vol. 1, no. 2, p. 3, 2004.
- [58] 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.
- [59] 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.
- [60] 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.
- [61] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

- [62] 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.
- [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] 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.
- [66] 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.
- [67] 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.
- [68] 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.
- [69] 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.
- [70] 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.
- [71] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.
- [72] 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.
- [73] P. Gijsbers, "Automatic construction of machine learning pipelines," 2017.
- [74] 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.
- [75] 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.
- [76] 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.
- [77] 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.

- [78] 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.
- [79] Y.-W. Chen and C.-J. Lin, “Combining svms with various feature selection strategies,” in *Feature extraction*, pp. 315–324, Springer, 2006.
- [80] 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.
- [81] M. A. Hall and L. A. Smith, “Practical feature subset selection for machine learning,” 1998.

## A Feature Selection Exploration Plots

The validation and test score for all combinations of dataset, ranking method and machine learning quality measure (Table 5) in figures (Figures 17, 18, 19, 20, 21, 22, 23 and 24).

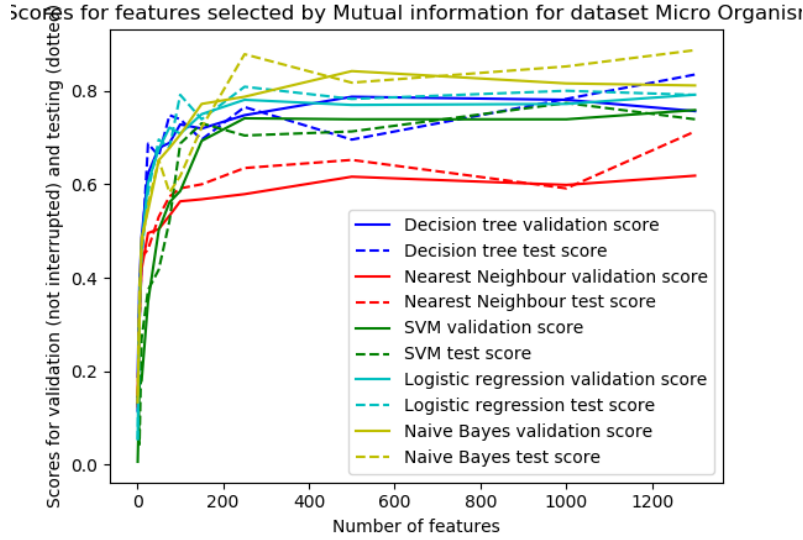


Figure 17: The validation and test score per feature preserved for the micro organisms data set with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

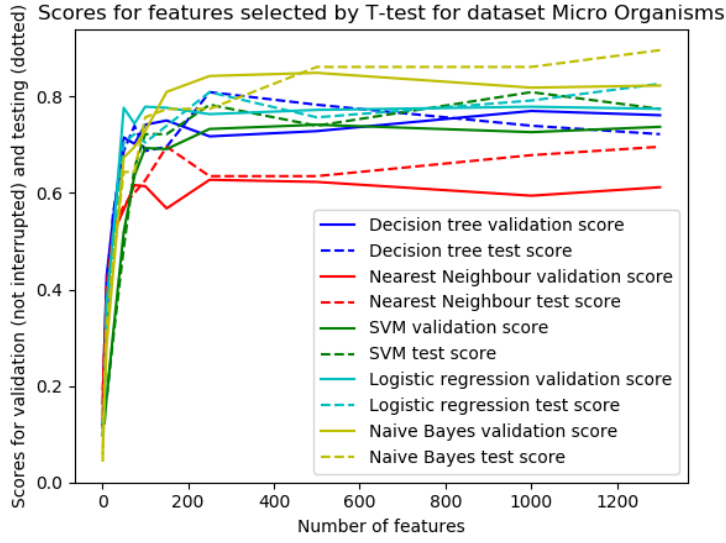


Figure 18: The validation and test score per feature preserved for the micro organisms dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

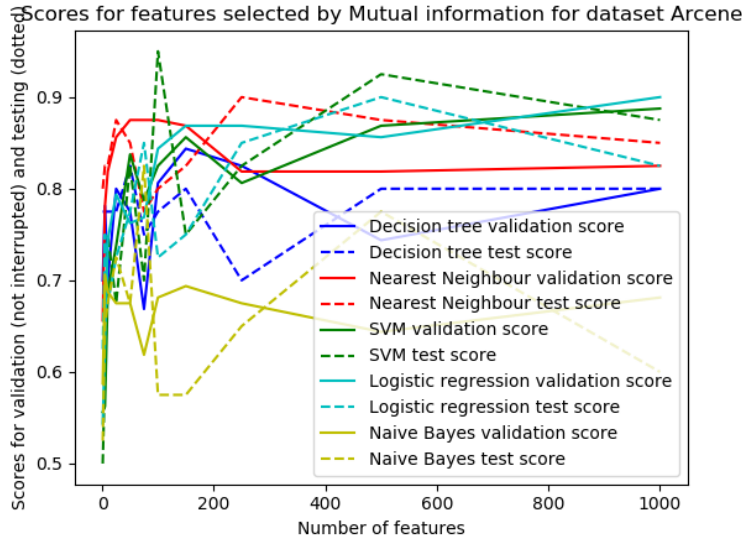


Figure 19: The validation and test score per feature preserved for the Arcene dataset with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

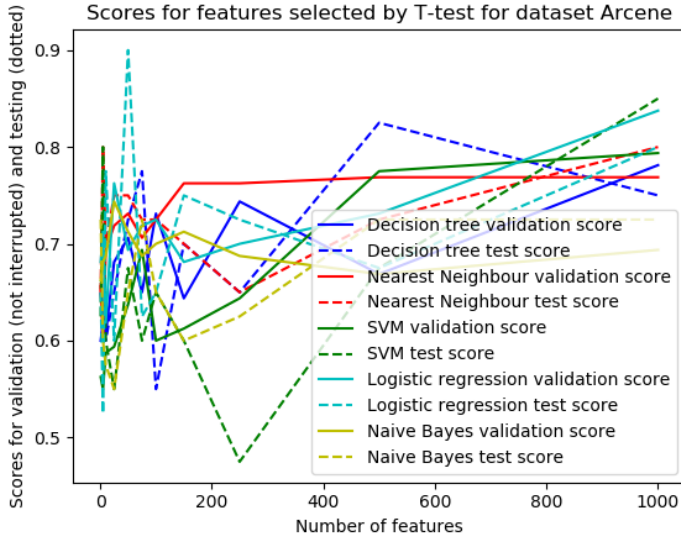


Figure 20: The validation and test score per feature preserved for the Arcene dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

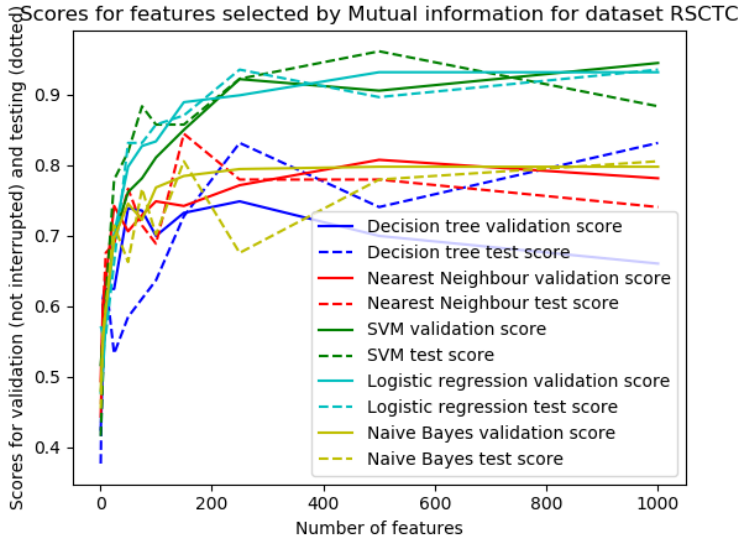


Figure 21: The validation and test score per feature preserved for the RSCTC dataset with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.

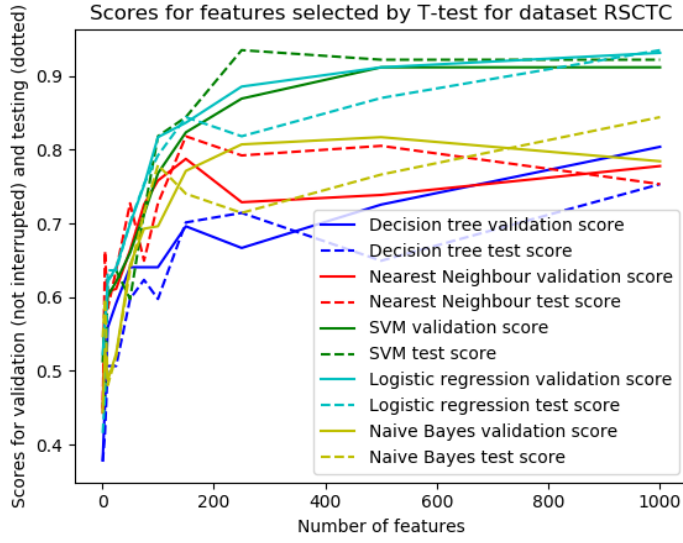


Figure 22: The validation and test score per feature preserved for the RSCTC dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

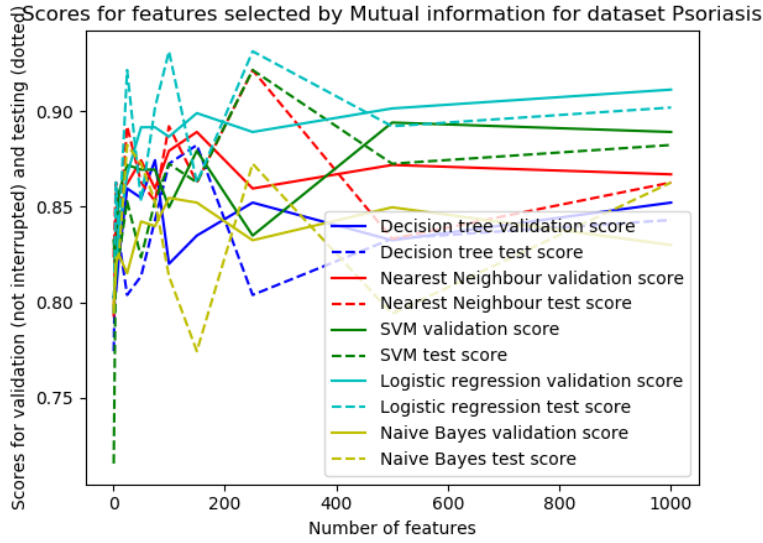


Figure 23: The validation and test score per feature preserved for the Psoriasis dataset with ranking method mutual information. Five different classification algorithms are used to define this classification and test score.



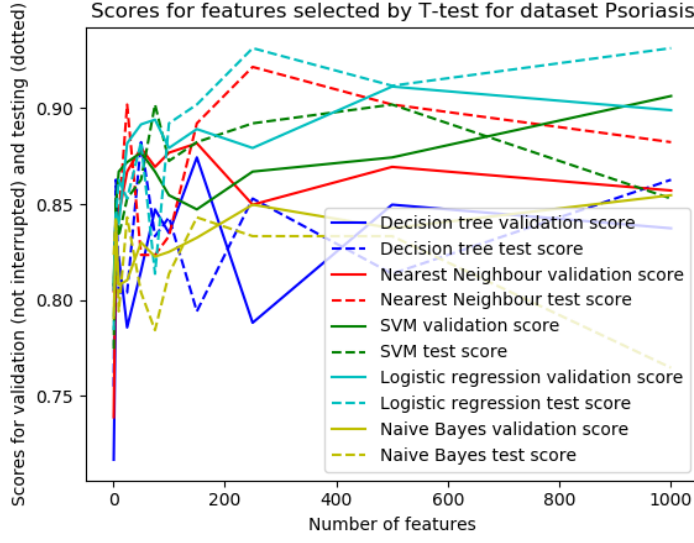


Figure 24: The validation and test score per feature preserved for the Psoriasis dataset with ranking method T-test. Five different classification algorithms are used to define this classification and test score.

## B Feature Selection Exploration Precision And Recall

The average precision (Figure 25) and recall (Figure 26). These are shown per data set and per ranking method.

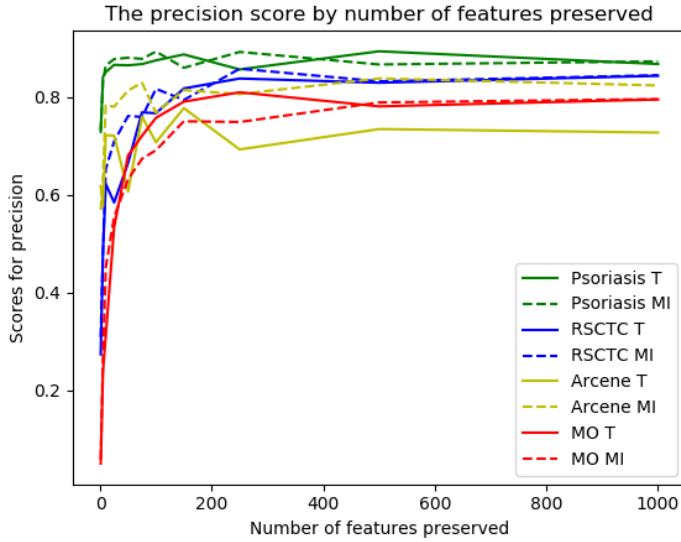


Figure 25: The average precision shown per dataset and rank.

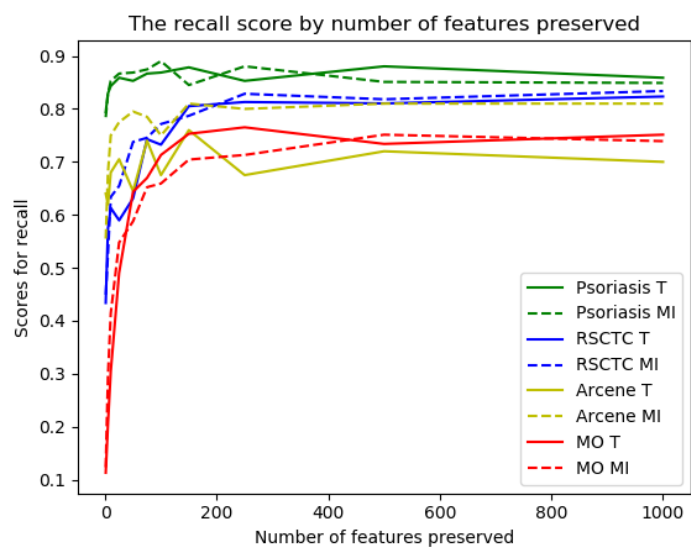


Figure 26: The average recall shown per dataset and rank.