A Benchmark of Prevalent Feature Selection Algorithms on a Diverse Set of Classification Problems

ANETTE KNIBERG

DAVID NOKTO
A Benchmark of Prevalent Feature Selection Algorithms on a Diverse Set of Classification Problems

ANETTE KNIBERG
DAVID NOKTO

This master thesis was performed in collaboration with Nordron AB
Supervisor: Torbjörn Nordling (CEO and Founder)

Degree project, Second cycle, 30 credits, 2018
KTH Royal Institute of Technology
Medical Engineering
School of Engineering Sciences in Chemistry,
Biotechnology and Health
SE-100 44 Stockholm, Sweden
Abstract
Feature selection is the process of automatically selecting important features from data. It is an essential part of machine learning, artificial intelligence, data mining, and modelling in general. There are many feature selection algorithms available and the appropriate choice can be difficult. The aim of this thesis was to compare feature selection algorithms in order to provide an experimental basis for which algorithm to choose. The first phase involved assessing which algorithms are most common in the scientific community, through a systematic literature study in the two largest reference databases: Scopus and Web of Science. The second phase involved constructing and implementing a benchmark pipeline to compare 31 algorithms’ performance on 50 data sets.

The selected features were used to construct classification models and their predictive performances were compared, as well as the runtime of the selection process. The results show a small overall superiority of embedded type algorithms, especially types that involve Decision Trees. However, there is no algorithm that is significantly superior in every case. The pipeline and data from the experiments can be used by practitioners in determining which algorithms to apply to their respective problems.

Keywords
Feature selection, variable selection, attribute selection, machine learning, data mining, benchmark, classification
Sammanfattning

Nyckelord
Variabelselektion, maskininlärning, datautvinning, klassificering
Acknowledgements

We thank our supervisor Torbjörn Nordling at Nordron AB for the opportunity, and the support we novices have received throughout this work. This thesis has given us a glimpse into a field that is interesting both from an engineering and philosophical point-of-view. We are also grateful to our families for boundless patience and encouragement.

Anette Kniberg
David Nokto
Table of Contents

Abstract .................................................................................................................. i
Sammanfattning ................................................................................................. iii
List of Figures and Tables ............................................................................... ix
1 Introduction .................................................................................................... 1
  1.1 Aim ............................................................................................................. 2
  1.2 Objectives .................................................................................................. 2
  1.3 Approach ................................................................................................... 2
  1.4 Delimitations .............................................................................................. 2
  1.5 Thesis overview ......................................................................................... 3
2 Theory .............................................................................................................. 5
  2.1 Machine learning ........................................................................................ 5
  2.1.1 Ethics ..................................................................................................... 6
  2.2 Feature selection ....................................................................................... 7
  2.2.1 Wrapper .................................................................................................. 9
  2.2.2 Embedded ............................................................................................. 10
  2.2.3 Filter ...................................................................................................... 10
  2.2.4 Univariate or multivariate selection ....................................................... 10
  2.3 Performance evaluation ............................................................................ 11
  2.3.1 Confusion matrix ................................................................................. 11
  2.3.2 Accuracy ............................................................................................... 12
  2.3.3 F-measure ............................................................................................. 12
  2.3.4 Runtime .................................................................................................. 13
  2.4 Cross-validation ......................................................................................... 13
  2.5 Hyperparameter optimisation ................................................................... 14
  2.6 Analysing benchmark results .................................................................. 15
    2.6.1 Parametric vs nonparametric statistical testing .................................. 15
    2.6.2 Multiple comparisons across multiple datasets ................................. 16
3 Method ........................................................................................................... 17
  3.1 Literature study ........................................................................................ 17
    3.1.1 Choosing feature selection algorithms ........................................... 17
    3.1.2 Choosing predictors .......................................................................... 18
    3.1.3 Choosing and pre-processing datasets .............................................. 19
  3.2 Experimental Setup .................................................................................. 19
    3.2.1 Implementation .................................................................................... 19
List of Figures and Tables

Figure 1: The increase of size and dimensionality of datasets. .................................................................1
Figure 2: Classification vs. regression problems. ..........................................................................................5
Figure 3: Curse of dimensionality. ..............................................................................................................8
Figure 4: Process of Sequential Forward Selection (SFS). ..........................................................................9
Figure 5: Process of Sequential Backward Selection (SBS). .................................................................10
Figure 6: Basic experimental process. .......................................................................................................11
Figure 7: Over- and underfitted models. ....................................................................................................14
Figure 8: Overview for one run in the pipeline...........................................................................................20
Figure 9: Tukey-boxplots of estimated accuracies and F-measures ...........................................................24
Figure 10: Comparison of estimated accuracies and F-measures .................................................................25
Figure 11: Tukey-boxplots of estimated rescaled runtimes from slowest to fastest (0-1). .........................27
Figure 12: Comparison of estimated and rescaled runtimes .....................................................................28
Figure 13: Discerning FSAs with best overall performance. ......................................................................29

Figure E2.1: Pseudocode for a part of the program....................................................................................52

Table 1: Simple data set with one feature and one output label...............................................................6
Table 2: A fictive diabetes diagnostic data set ............................................................................................7
Table 3: Summary of FSA types .................................................................................................................11
Table 4: A 3x3-dimensional confusion matrix ............................................................................................12
Table 5: Table 4 after micro averaging for B-class .....................................................................................12
Table 6: Cancer detection ...........................................................................................................................12
Table 7: Examples of runtime rescaling ....................................................................................................13
Table 8: A four - fold cross-validation .......................................................................................................14
Table 9: FSAs used in the experiment .......................................................................................................17
Table 10: Predictors used in the experiment. .............................................................................................19
Table 11: Hyperparameter settings for each FSA ......................................................................................21
Table 12: Hyperparameters for each predictor ..........................................................................................21
Table 13: Best performing group of FSAs regarding predictive performance and runtime ...23
Table 14: Comparisons of various feature selection benchmarks ..........................................................33
Table C1: Algorithms with the top 30 number of publications .................................................................47
Table D1: Data sets used in the experiment ...............................................................................................49
Table F1: Summary statistics of overall performance of each FSA across datasets.........................53
1 Introduction

This chapter explains the purpose of the project by introducing the topic and briefly explaining its importance in a broad context and the current state, including problems and limitations. The problems addressed by the project are specified, along with aims and objectives, to which approaches and delimitations are presented.

In recent years, the size of collected data sets are increasing, both regarding the number of observations and number of dimensions (Fig. 1). More data means potentially greater predicting power, discoveries and understanding of phenomenon. However, large amounts of raw data by itself is not particularly useful for analysis and must be processed to extract patterns or insight (1). One such pre-processing step is to determine which features reflect the underlying structure of the data.

Figure 1: The increase of size and dimensionality of datasets. The data points were collected from the UCI Data Set Repository (2).

Feature selection is the process of automatically selecting a subset of features from the original data set. Its importance for machine learning, artificial intelligence, statistics, data mining and model building is undisputed and the use will increase as data sets grow (3–5). Feature selection removes irrelevant, redundant and noisy data, which reduces computation costs, improves predictive performance and model interpretability (6).

There are many feature selection algorithms (FSAs) available but knowing which one to use for a given scenario is not an easily answered question (5). Several studies have concluded that different FSAs vary greatly in performance when applied on the same data (7). Others have claimed that all algorithmic performance is equal when averaged on every possible type of problem (8). Some have found that incorrectly applied FSAs can reduce predictive performance (1,9).

Finding the optimal feature selection algorithm remains an open research question and many benchmark studies have been undertaken. The problems with many existing benchmarks are
• No or poor motivation for the choice of algorithms included.
• Small number of algorithms and data sets included.

It therefore remains unclear for inexperienced practitioners from different fields which FSAs to use.

1.1 Aim

This report is intended for practitioners who are not data scientists, statisticians or machine learning experts. The aim is to benchmark common feature selection algorithms on a large and diverse set of classification problems. This diversity might help readers find results on data sets that are similar to their own. Hopefully it will help users in choosing an algorithm and serve as an introduction to the field. We also hope to entice the more experienced reader to try out alternative algorithms that are not routinely used within their field. Additionally, we will construct an experimental pipeline for benchmarking FSAs. The resulting software will be available for Nordron AB to extend and develop further into an interactive benchmarking tool.

1.2 Objectives

The objectives consist of answering the following questions:

1. Which feature selection algorithms are the most prevalent in literature?
2. Taking the previous objective into account, which feature selection algorithm(s) has the overall best performance?

1.3 Approach

The first objective was addressed quantitively, by searching certain scientific databases for how publications existed for each FSA. The second objective was solved by constructing a pipeline capable of performing the experiment and analysing the results. By testing 31 feature selection algorithms on 50 data sets using three performance metrics, we acquired a large amount of information that can be used for comparison.

1.4 Delimitations

• The number of selected features was set to 50% of the original features for all algorithms.
• Feature extraction techniques were not included.
• Performance in this work is evaluated in terms of accuracy, F-measure and runtime.
• No algorithms were written from scratch, only pre-made software packages were used.
• Due to the number of algorithms involved and the intended reader, this work focuses on the application, not on the inner workings of the algorithms.
• The quality (feature engineering) of the data sets were not questioned.
1.5 Thesis overview

The remaining chapters are structured as follows:

Chapter 2. Presents the theoretical background related to the thesis. It explains the concepts of machine learning, feature selection, how to evaluate algorithmic performance and statistical analysis.

Chapter 3. Explains the literature study, experimental setup.

Chapter 4. The results are presented in terms of predictive performance and runtime.

Chapter 5. The results are discussed.

Chapter 6. The conclusions are presented. It further brings up recommendations for practitioners and suggestions for future research.

Appendix A-F. Contains relevant but not essential information such as raw data from the experiments, search queries, definitions and software package information.
2 Theory

This chapter explains the theory that is relevant for understanding the thesis. It starts by describing machine learning in a broader sense and works towards more detailed subjects. It explains the concepts: machine learning, feature selection, performance evaluation, cross-validation, hyperparameter optimisation and strategies for analysis.

2.1 Machine learning

Machines learning is a scientific discipline concerned with algorithms that automate model building, in order to make data-driven predictions or analysis. It borrows from several other fields such as psychology, genetics, neuroscience and statistics (6). The applications are many and varied, to name but a few: medical diagnostics (10), image recognition (11), text categorisation (12), DNA analysis (13), recommendation systems (14), fraud detection (15), social media news feeds (16), search engines (17) and self-driving vehicles (18).

A central concept of human and machine learning is called the Classification Problem. It can be seen as a variant of the famous philosophical Problem of Induction (19), which questions how one can generalise from seen examples to unseen future observations. In classification, this means determining what category a new observation belongs to (Fig. 2.a). Knowing the class (or label) of an object means that one can foresee its properties and act accordingly. It is an important technique since it solves many practical problems such as deciding if an email is spam by reading the title or determining the gender of a person by looking at an image. The classes in these cases are categorical or discrete. If the class values are continuous it is a regression problem (Fig. 2.b). An example is to determine the price of a used car by looking at mileage, age etc.

Figure 2: Classification vs. regression problems. Classification problem (A): The two shapes belong to different classes. The line is the decision boundary made by the trained model and determines how the model classifies new examples. Regression problem (B): The line represents a model that has been fitted to data.
There are several ways to categorise learning types. If a problem has known class labels, it is called a *supervised* machine learning problem (also called function approximation). A problem without known class labels is an *unsupervised* problem. Some structure must then be derived from the relationships of the data points. *Semi-supervised* problems have both labelled and unlabelled data. The labelled data is used as additional information to improve unsupervised learning. All three types of learning can be applied to both classification and regression problems. This thesis concerns supervised learning algorithms applied on classification problems.

There are many terms used to refer to machine learning algorithms and the related concepts. In this report learning algorithms are referred to as *predictors*, while the term *algorithm* refers to both predictors and feature selection algorithms. The terms *class* and *label* are used to denote categories of data instances.

Each type of predictor has a different strategy for how it builds models depending on the underlying mathematics. The process of building models by loading data into the predictor is called *fitting* or *training*. It can be regarded as the search for a function whose input is an instance of the data and the output is a label. Given the data set shown in Table 1, a predictor would likely produce a simple function where the output is the input squared, \( y(x) = x^2 \). When this function is given a new input such as 10, it will make the prediction that the label is 100.

<table>
<thead>
<tr>
<th>instances</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>labels</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>25</td>
<td>36</td>
<td>49</td>
</tr>
</tbody>
</table>

Both the predictive performance and training time of the model depends on several factors, such as the data quality, size and dimensionality, in addition to the underlying mathematics and mechanisms of the predictor. Having access to bigger data sets means that the predictor gets more training examples which generates - but does not guarantee - more accurate models. It also increases the training time since more instances must be processed. Predictors with simple underlying mathematics are trained faster than complex ones since less computation is needed to build the model.

### Ethics

Machine learning is having a huge impact on the world and consequently raises a score of ethical questions. Poorly designed implementations of machine learning not only inherit but can amplify biases and prejudices. If a predictor is trained on a data set that contains biases it may project these upon use (20–22). An example of this regards risk assessment algorithms used to predict future criminal behaviour, of which one was shown to assign more false positives to certain groups of people (23). Some systems decide what news articles are displayed in social media, potentially misrepresenting and biasing public opinion and world awareness (24). Other situations raise the question of culpability such as who to blame when a self-driving car collides (25). There is also the question of artificial intelligence rendering a large portion of the human workforce redundant. Frey and Osborne (2017) estimate that 47 % of US jobs have a high risk (>0.7) of becoming automatable in the next few decades (26). Many machine learning systems are black boxes and thus unfair decisions on subjective problems could happen unbeknownst to the user or even the designer. In summary, this development puts a great weight on responsible data collection, design transparency and accountability. How to design fair algorithms is an open research question (27,28).
2.2 Feature selection

“Make everything as simple as possible, but no simpler” - Albert Einstein (as paraphrased by Roger Sessions in the New York Times, 8 January 1950)

The features of a data set are the semantic properties that describe the object or phenomenon of interest. When a data set is represented as a table, the features are usually the columns and each instance is a row. In supervised problems, there is also a column denoting class membership (Table 2). Consequently, every instance is made up by a feature vector, and a corresponding class label.

Table 2: A fictive diabetes diagnostic data set. Each instance represents a patient.

<table>
<thead>
<tr>
<th>High insulin resistance</th>
<th>Weight (kg)</th>
<th>Family history of disease</th>
<th>Height (cm)</th>
<th>Shoe size (paris point)</th>
<th>Favourite movie</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>123</td>
<td>yes</td>
<td>172</td>
<td>40</td>
<td>“Minority Report”</td>
<td>positive</td>
</tr>
<tr>
<td>yes</td>
<td>81</td>
<td>no</td>
<td>190</td>
<td>47</td>
<td>“Ex Machina”</td>
<td>negative</td>
</tr>
<tr>
<td>no</td>
<td>95</td>
<td>yes</td>
<td>163</td>
<td>37</td>
<td>“Her”</td>
<td>negative</td>
</tr>
</tbody>
</table>

There are many possible definitions of feature selection. Different aspects of the process have received varying emphasis depending on the researcher stating the definition. A proposed general definition, and used for this thesis is:

“Feature selection is the process of including or/and excluding some features or data in modelling in order to achieve some goal, typically a trade-off between minimising the cost of collecting the data and fulfilling a performance objective on the model.” - Torbjörn Nordling (Assistant Professor, Founder of Nordron AB, December 2016)

The difficulty in finding fully overlapping definitions could be because there are several types of FSAs with varying objectives. A collection of definitions rewritten as mathematical optimisation problems (Appendix A), resulted in three points that are central to the concept:

1. A chosen performance objective on the resulting model such as reduced training time or improved predictive accuracy.

2. The reduced feature vector—the feature subset—should be as small as possible.

3. The label distribution of the subset should be as close as possible to the original label distribution.

Features selection can thus be viewed as a pre-processing step applied on data before it is used to train a predictor. FSAs can themselves use predictors, and some predictors perform internal feature selection as part of the training process. In the examples in Table 2, a FSA would likely omit the “Shoe size” and “Favourite movie” columns since they contain useless information for diabetes diagnostics. The technique should not be confused with feature extraction, where features are altered, such as reducing the number of features by combining them; feature selection omits features without altering the remaining ones.

There is no guarantee that every feature is correlated with the label. A feature is relevant if its removal reduces predictive performance. Irrelevant features have low or no correlation with the label. Redundant features have a correlation with the label but their removal does not reduce performance due to the presence of another feature; both being relevant on their own. An example of this could be the same value given in two different units, represented as two different features. (29)
There are several reasons for removing features from data sets. Features with low or no correlation can lead to overfitting, meaning that the model is overly complex and describes noise and random error rather than underlying causality. The consequence is reduced predictive performance since the model does not generalise well on unseen data (6). Overfitting has been described as the most important problem in machine learning (30). Other causes and remedies for overfitting are mentioned in sections 2.4 Cross-validation and 2.5 Hyperparameter optimisation.

Another reason for using FS is higher learning efficiency. Since the data set is less complex the learning process is faster. Additionally, a data set with more features requires more instances for the predictor to find the best solution. This is due to the density of the instances decreasing as the dimensionality increases, known as the Curse of Dimensionality (Fig. 3). Since the number of instances often is fixed, reducing the number of features is tantamount to increasing the amount of instances (6).

![Figure 3: Curse of dimensionality. The three figures show the same data points plotted in one (A), two (B) and three (C) dimensions. Even though it is possible to draw a cleaner decision boundary in three dimensions than in two, the predictive performance will probably be worse due to increased data sparsity.](image-url)
Feature selection also improves model interpretability. A predictor trained on a high-dimensional data set will naturally produce a complex model, making it difficult to understand the modelled phenomenon.

Attempts have been made to categorise FSAs depending on how a feature subset is chosen: wrapper, embedded and filter. However, there is no established unified framework and there are discrepancies between the definitions of these categories. The descriptions below were formulated after analysis of eight publications that produced category definitions (1,4–7,31–33)

### 2.2.1 Wrapper

This category combines a search strategy with a predictor to select the best subset. By training with different candidate subsets and comparing their performances, the subset that gives the best performance is kept. An example: a data set consists of three features $D = \{x_1, x_2, x_3\}$. Three subsets $S \subset D$, $S_1 = \{x_1, x_2\}$, $S_2 = \{x_1, x_3\}$, $S_3 = \{x_2, x_3\}$ are used separately to train the same type of predictor, producing three models. The performance of the models is then compared on a chosen performance measure such as accuracy. The subset that yields the best performing model is selected as the final subset. Note that in this example only 42.8% of all feature combinations are examined. The only way to guarantee finding the best solution would require testing all seven feature combinations with an exhaustive search (1).

An exhaustive search is only feasible if the number of features is small (1,29). A data set with 90 features, such as the Libras data set in this thesis, has $2^{90} \approx 1$ possible subsets. If 100,000 subsets can be examined each second it would take more than $4 \times 10^{18}$ years to finish. Applying heuristic search strategies enables finding adequate solutions in a shorter timeframe, without testing all possible feature combinations. Some examples of heuristic search strategies are GRASP (34), Tabu Search (35), Memetic Algorithm (36), Sequential forward selection (SFS) and Sequential backward selection (SBS) (37).

The two strategies used in this work are SFS and SBS. The former works by starting with an empty feature set and sequentially adds the feature that yields the best score until the desired subset size is reached. The latter starts with the full feature vector and sequentially removes the feature that yields the worst score. Both algorithms have the same complexity but generate different candidate subsets, leading to different solutions as illustrated in Figures 4 and 5. In this thesis, the names of wrapper algorithms are combinations of the internal predictor and search strategy, for example: AdaBoost SBS, Decision Tree SFS and Perceptron SFS.

---

**Figure 4: Process of Sequential Forward Selection (SFS).** The objective function, for instance predictive accuracy, is represented by $J(\cdot)$. Starting from the top, features are added to the best performing subset. The best subset is found to be the combination of $x_1$, $x_2$ and $x_3$ (circled).
2.2.2 Embedded

Another way to use predictors for feature selection is to examine the model structure instead of performance. This can be illustrated in the case when the trained predictor is a linear polynomial. The model is represented by the function \( y \), \( \bar{x} \) is the feature vector and \( \bar{C} \) is a vector of coefficients:

\[
y(\bar{x}) = \bar{C} \bar{x} = C_1 x_1 + C_2 x_2 + C_3 x_3 + C_4 x_4
\]

These coefficients can be viewed as weights that the predictor has given each feature, which indicate their importance in the generated model. By choosing some threshold value of \( \bar{C} \) or length of \( \bar{x} \), unimportant features are omitted. If for example it is decided that half the features shall be omitted and \( C_3 > C_2 > C_1 > C_4 \) the resulting subset will be \( S = \{ x_3, x_2 \} \).

This type of embedded method, where the learning algorithm is trained only once, will be referred to as standard embedded. Another type is Recursive Feature Elimination (RFE) which checks the feature weights and iteratively re-trains the predictor after removing the least important feature with each iteration. In this thesis, the names of the embedded methods are the names of the internal predictors used, with the suffix ‘RFE’ added on the versions that use recursive feature elimination. Some examples are Random Forest, Random Forest RFE, Perceptron and LASSO.

2.2.3 Filter

Filter methods assess feature importance without the use of predictors by examining the general characteristics of the data set. If for example a feature value barely varies across all instances, it is ineffective for differentiating between labels. Therefore, calculating the \( \chi^2 \) or F-test statistic for a feature column are ways to estimate correlation with the label. Other filter methods, such as Minimum Redundancy Maximum Relevance (mRMR) and Correlation-based Feature Selection (CFS), also try to eliminate redundant features by calculating feature-to-feature correlation, thus further improving the data set quality.

2.2.4 Univariate or multivariate selection

Filter and embedded methods rank features according to their individual importance while wrapper methods choose entire subsets at a time (Table 3). Theoretically, the complexity of univariate methods should therefore scale linearly with the number of features and thus be faster than wrappers, especially when dealing with high-dimensional data sets. Wrappers, on the other
hand, directly test the predictive performance of entire feature subsets. This excels in situations where a combined set of features is optimal, regardless of their individual importance.

Table 3: Summary of FSA types.

<table>
<thead>
<tr>
<th>Involves internal predictors</th>
<th>Basis of selection</th>
<th>Uni- or multivariate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wrapper</td>
<td>Model performance</td>
<td>Multivariate</td>
</tr>
<tr>
<td>Embedded</td>
<td>Model structure</td>
<td>Univariate</td>
</tr>
<tr>
<td>Filter</td>
<td>Statistical measures</td>
<td>Univariate</td>
</tr>
</tbody>
</table>

2.3 Performance evaluation

The performance of a predictor depends on the quality of the data used for training. Since FS is a pre-processing step applied on the data, the performance of the predictor implicitly represents the performance of the FSA (Fig 6). The general process is described in the following steps:

1. The data set is split into two sets, one for training and one for testing.
2. The training set is run through the FSA which produces a subset with less features.
3. The subset is used to train an external predictor.
4. The test set (without its labels) is inputted into the predictor which produces predictions, in other words, guesses the label of each instance.
5. The predictions are compared to the true labels in order to measure predictive performance.

This approach has some problems that need to be resolved. Different predictors have different biases and might favour certain FSAs. It is therefore prudent to choose several predictors with varying underlying mathematics. It is also wise to test performance with different metrics. Firstly, a predictor may perform optimally on one metric and suboptimally on another (41). Secondly, the performance metrics in turn have various strengths and weaknesses. Thirdly, practitioners in different scientific fields prefer different metrics (and predictors) (41). The measures used in this experiment are accuracy, F-measure and runtime. In order to understand the former two, it is necessary to understand confusion matrices.

2.3.1 Confusion matrix

Several performance metrics for predictions are derived from the confusion matrix: a n×n-dimensional matrix with n being the number of classes in the dataset (Table 4). All the predictions on the diagonal are correct. A n×n-dimensional can be reduced to a 2×2 matrix by applying micro averaging (Table 5), enabling calculation of other compressed performances metrics, such as accuracy and F-measure. True positives and true negatives are the number of correctly predicted positive and negative cases respectively. False positives and false negatives are the number of falsely predicted positive and negative cases and are also known as Type I and Type II errors respectively.
Table 4: A 3x3-dimensional confusion matrix. There are for example 17 instances of A but seven of these are falsely classified as B and C i.e., false negatives.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class</td>
<td>A</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 5: Table 4 after micro averaging for B-class.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>B</th>
<th>Not B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class</td>
<td>B</td>
<td>10</td>
</tr>
<tr>
<td>Not B</td>
<td>6</td>
<td>23</td>
</tr>
</tbody>
</table>

2.3.2 Accuracy

The degree of which the predictions match the reality that is being modelled. It is defined as the number of true positives (TP) and true negatives (TN) divided by the total amount of predictions, which includes false positives and negatives (FP and FN):

\[
ACC = \frac{TP + TN}{TP + TN + FP + FN}
\]

Accuracy can be misleading when a data set has an imbalanced class distribution. Consider a cancer data set with 90500 negative cases and 1000 positive cases. The predictor in Table 6 would get a 99% accuracy even though it missed half of the cancer cases. Since many real-world data sets have imbalanced class distributions (6) accuracy alone is insufficient as a performance metric.

Table 6: Cancer detection.

<table>
<thead>
<tr>
<th>Actual Class</th>
<th>Positive</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Class</td>
<td>Positive</td>
<td>500</td>
</tr>
<tr>
<td>Negative</td>
<td>500</td>
<td>90000</td>
</tr>
</tbody>
</table>

2.3.3 F-measure

This is a mean between precision and recall. Precision (also called positive predictive value) is the portion of predicted positives that are actually positive. Low precision happens when few of the positive predictions are true. Recall (also called true positive rate and sensitivity) is the portion of true positives that are predicted positive. Low recall happens when few of the true positive cases are predicted at all. An algorithm with a good F-measure has both high precision and recall since it is heavily penalised if either has a small value. F-measure is therefore better than accuracy when dealing with imbalanced data sets. In the case with Table 6, the algorithm would only get a F-measure of 50%. However, it does not consider the number of true negatives which is a weakness
when the negative class label is interesting. Precision (P), recall (R) and F-measure (F) are calculated as follows:

\[
P = \frac{TP}{TP + FP} \\
R = \frac{TP}{TP + FN} \\
F = \frac{2 \cdot P \cdot R}{P + R}
\]

### 2.3.4 Runtime

The time it takes for a FSA to produce a subset. Speed is important when dealing with large, high-dimensional data sets. It also enables scrupulous calibrations and reruns which in turn can improve predictive performance.

In contrast to comparing FSAs runtime using only one data set, comparisons across multiple data sets requires further treatment. Calculating the mean of the runtimes for a FSA across data sets is not prudent since large data sets would result in misleading outliers. It would also make it difficult to visualise and compare all runtimes. This problem is addressed by rescaling the runtimes for each FSA and dataset. \(x'\) is the rescaled runtime, \(x\) is the initial runtime, \(x_{\text{max}}\) and \(x_{\text{min}}\) are the largest and smallest runtimes in the set respectively (42). The rescaled values range from 0 to 1 with 0 being the slowest and 1 being the fastest in the set. An example is shown in Table 7.

\[
x' = 1 - \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}
\]

Table 7: Examples of runtime rescaling.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Runtime</th>
<th>Rescaled runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Set 1</td>
<td>Data Set 2</td>
</tr>
<tr>
<td>FSA 1</td>
<td>15</td>
<td>300</td>
</tr>
<tr>
<td>FSA 2</td>
<td>45</td>
<td>8600</td>
</tr>
<tr>
<td>FSA 3</td>
<td>5</td>
<td>450</td>
</tr>
<tr>
<td>FSA 4</td>
<td>60</td>
<td>10 000</td>
</tr>
</tbody>
</table>

### 2.4 Cross-validation

Training and testing a predictor on the same data can lead to overfitted models since the instances used for training and testing are the same. To know how well an algorithm performs on unknown data it is therefore important to test the performance with unseen instances. A number of instances are chosen to be the training set and the rest is the testing set. This is known as a train/test split and enables a more realistic estimate of the predictive performance. A problem however, is that the performance highly depends on how the instances are distributed between the two sets; they may for example have disparate class distributions. A way to bypass this is to reuse the data several times with different equally sized folds, train the predictor separately with each training fold, test the performance with each test fold and average the measured performances. In Table 8 each partition is used three times for training and once for testing. The three training sets in the training fold are used in union to train the predictor in each run. This is known as cross-validation (CV) and is a better estimate of performance on unknown data. Another advantage of CV is a more economic use of data since every instance is recycled.
Table 8: A four-fold cross-validation.

<table>
<thead>
<tr>
<th></th>
<th>Train set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>First fold</td>
<td>1 2 3 4</td>
<td></td>
</tr>
<tr>
<td>Second fold</td>
<td>2 3 4 1</td>
<td></td>
</tr>
<tr>
<td>Third fold</td>
<td>3 4 1 2</td>
<td></td>
</tr>
<tr>
<td>Fourth fold</td>
<td>4 1 2 3</td>
<td></td>
</tr>
</tbody>
</table>

The type of CV applied in this experiment is called Stratified K-fold where the class distribution of the entire data set is approximately represented in each fold. This is done to reduce the variability of the predictor performance between folds. It has been proven to be superior to regular CV in most cases (29).

2.5 Hyperparameter optimisation

Hyperparameters are settings that need to be specified for many algorithms and have a big impact on performance, such as model flexibility and overfitting. Figure 7 show models that are over- and underfitted respectively due to poor hyperparameter choices. In order to do a fair benchmark of machine learning algorithms it is important they perform optimally (43). Hyperparameter optimisation (HPO) is therefore a vital step. Determining the correct hyperparameters is an optimisation problem that depends on the data set and performance metric in question. It must therefore be performed anew for both the FSAs and predictors every time a data set is processed.

Figure 7: Over- and underfitted models. Overfitted models (top row) and underfitted models (bottom row).
There are several variants of HPO, however to preserve reproducibility grid searches were implemented in this experiment. In grid searches, all parameters are manually chosen and the program exhaustively produces FSAs and predictors using each combination of parameter values. The procedure is computationally expensive but prudent when comparing algorithms since it does not involve randomness. An example is the Support Vector Machine FSA with two varied hyperparameters $C$ and $\gamma$:

$$C \in \{2^{-3}, 2, 2^3, 2^5, 2^9, 2^{13}\}$$

$$\gamma \in \{2^{-15}, 2^{-11}, 2^{-7}, 2^{-5}, 2, 2^3\}$$

Since both parameters have six values with each pair is used once, it is run 36 times within each cross-validation fold. Each time it produces a new—but not necessarily–unique feature subset, which is then sent to the predictor. The predictor in turn is trained 36 times for each of its own hyperparameter settings. In this benchmark, the reason for also performing HPO on the predictors even though only the FSAs are being compared, is to ensure the predictors’ default settings do not randomly favour particular FSAs.

Feature subset size can also be considered a hyperparameter. Many FSAs require this to be specified, but the optimal number of features is most often unknown. If the subset is too large, it may eliminate the purpose of FSA; if the subset size is too small, it may omit relevant features and result in a biased predictor. In practice one usually tests the performance in a grid search fashion with subsets of varying sizes and picks the size with the best performance (4).

### 2.6 Analysing benchmark results

In order to discern which FSAs perform better, statistical hypothesis testing can be used. The many different strategies available can easily become overwhelming, each with their various benefits and limitations. The following section explains the two main categories of these and how to choose between them. The subsequent section explains briefly the issue with multiple comparisons, as with the case of using several different data sets, describing specifically the method that was used in this thesis for data analysis.

#### 2.6.1 Parametric vs nonparametric statistical testing

Parametric hypothesis tests make strong assumptions about the distribution of the underlying data such as normality, and can be powerful in rejecting a false null hypothesis when conditions are met. In contrast, nonparametric tests are less powerful but make weaker assumptions, allowing for nonnormality (44). Machine learning and data mining communities have voiced concerns regarding misuse and misinterpretations of hypothesis testing, which can lead to misinformed conclusions. Strict conditions on data distribution can incorrectly be assumed fulfilled resulting in overly confident results and false rejection of the null hypothesis. Yet as with nonparametric tests, the decreased power can result in interesting finds being missed, effectively blocking potential new discoveries (45).

When choosing between methods of statistical analysis, the distribution of the performance scores needs inspection. Departures from normality can be seen with Tukey boxplots of the performance score for each FSA as modality, skewness and unequal variances. However, distributions can be deemed pseudo-normal depending on how severe these violations are, which can still warrant for a parametric method. If severe, the less powerful conservative nonparametric method is necessary, but offers consolation regarding observational value. In the case of FSA comparison, the value of mean performances across different datasets lack meaning, whereas a more interesting observational value is how the FSAs rank on each dataset. Nonparametric techniques in general make use of these individual ranks, in contrast to parametric techniques which use absolute values.
2.6.2 Multiple comparisons across multiple datasets

There are many ways of comparing algorithms, and the choice depends on the experimental setting and conditions. The following is an example of multiple testing:

1. State a null hypothesis for every pair of algorithms in the study, such as “A is equivalent to B”, and “A is equivalent to C”.
2. Calculate the p-value for every pair, using for instance pairwise t test.
3. Reject or retain each null hypothesis based on the p-values.

The problem with this procedure is well known, and referred to as the multiple comparisons, multiplicity or multiple testing problem. In essence: with many null hypothesis, there is risk of one getting rejected by chance. A choice is therefore required among statistical techniques that can deal with multiple comparisons.

Although there is yet no established strategy for comparing several predictors on several datasets, among the most well-known nonparametric techniques is the Friedman test. Iman and Davenport (1980) formulated an extension to the Friedman’s $\chi^2_F$ statistic in order to increase power (46); it was therefore the chosen technique for our experiment. Formally, $k$ FSAs are applied on $N$ datasets. Let $r_{ij}$ be the rank of FSA $j$ on dataset $i$ regarding performance score. Using the average rank $R_j = \frac{1}{N} \sum_i r_{ij}$, the Friedman statistic $\chi^2_F$ is calculated:

$$
\chi^2_F = \frac{12N}{k(k+1)} \left[ \sum_j R_j^2 - \frac{k(k+1)^2}{4} \right]
$$

If the ranks are tied, the average of the tied ranks can be assigned. Missing values can be handled by receiving rank zero, and adjusting $N$ to $N' = N - \text{count of missing values}$ before averaging. The Iman-Davenport $F_F$ statistic is further calculated using

$$
F_F = \frac{(N-1)\chi^2_F}{N(k-1) - \chi^2_F}
$$

which in turn is distributed along the $F$ distribution with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom.

Multiple-hypothesis tests check if there exists at least one pair of FSA that are significantly different. Which pair or pairs that differ requires post hoc testing. The rationale of the Nemenyi post hoc test is that two FSAs differ in performance if their corresponding average ranks $R_j$ differ by at least the critical distance (CD), defined as

$$
CD = q_{\alpha} \sqrt{\frac{k(k+1)}{6N}}
$$

with the Nemenyi critical value $q$ at significance level $\alpha$, which is derived from the studentised range statistic divided by $\sqrt{2}$ (46).
3 Method

This chapter further explains the approaches presented in chapter 1. The resulting choice of FSA, predictors and datasets are presented, followed by a description of the experimental setup for FSA implementation.

3.1 Literature study

The literature study resulted in 31 feature selection algorithms, three predictors and 50 data sets to include in the experiments. The following three sections describe how these were selected.

3.1.1 Choosing feature selection algorithms

The first criterion for inclusion of a FSA in this benchmark was prevalence. With the vast body of different algorithms, it can be difficult to discern which ones are most used by researchers. Due to diverse terminology it was necessary to have a comprehensive search string to uncover as many publications as possible. For this purpose, the list found in (47) was modified and extended. This set of terms is referred to as search string Feature Selection (ssFS) and is made available in Appendix B. Using ssFS to search Scopus, Web of Science and Google Scholar, FSA names were collected from review articles to a total of 102 algorithms. Every synonym and abbreviation for each name were used in combination with ssFS for searches of FSA publications in Web of Science and Scopus. For example, the search string used for the algorithm LASSO was (“LASSO” OR “least absolute shrinkage and selection operator”) AND (“feature selection” OR “attribute selection” OR “variable selection” OR …). The number of hits acted as an approximation of a method’s prevalence. The top results of these searches are found in Appendix C, along with a discussion about the difficulties of such a search.

The second criterion was to include as many categories of FSAs as possible, with a variety of underlying mathematics; the assumption being that different types have varying advantages and limitations. Taking the intended reader/user into account, the choice of FSAs was also restricted to the availability of pre-made software packages. Some highly prevalent methods, such as Genetic Algorithm and Multilayer Perceptron, were omitted since they were judged to require a high expertise to implement. The final chosen FSAs are listed in Table 9.

<table>
<thead>
<tr>
<th>Name</th>
<th>Machine Learning Category</th>
<th>Basic description</th>
<th>FSA Category</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost</td>
<td>Tree Ensemble</td>
<td>Builds models with weighted combinations of simple decision trees. Weights are assigned depending on how the trees misclassify.</td>
<td>Embedded</td>
<td>(48)</td>
</tr>
<tr>
<td>AdaBoost RFE</td>
<td></td>
<td></td>
<td>Wrapper</td>
<td></td>
</tr>
<tr>
<td>AdaBoost SBS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AdaBoost SFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ANOVA</td>
<td>Statistical</td>
<td>Selects the features with the highest variance.</td>
<td>Filter</td>
<td>(49)</td>
</tr>
<tr>
<td>Correlation Based FS  (CFS)</td>
<td>Statistical</td>
<td>Selects features that have a high correlation with the label and low correlation with each other.</td>
<td>Filter</td>
<td>(7)</td>
</tr>
<tr>
<td>Chi Squared</td>
<td>Statistical</td>
<td>Selects the features with the highest variance.</td>
<td>Filter</td>
<td>(50)</td>
</tr>
<tr>
<td>Name</td>
<td>Machine Learning Category</td>
<td>Basic description</td>
<td>FSA Category</td>
<td>Source</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------</td>
</tr>
<tr>
<td>Decision Tree</td>
<td></td>
<td>Builds a tree structure where the nodes are features and leaves are labels. Parent nodes have a higher feature-to-label correlation.</td>
<td>Embedded</td>
<td>(51)</td>
</tr>
<tr>
<td>Decision Tree RFE</td>
<td>Decision Trees</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decision Tree SBS</td>
<td></td>
<td></td>
<td>Wrapper</td>
<td></td>
</tr>
<tr>
<td>Decision Tree SFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fast Correlation Based FS (FCBF)</td>
<td>Statistical</td>
<td>Selects features that have a high correlation with the label and low correlation with each other.</td>
<td>Filter</td>
<td>(52)</td>
</tr>
<tr>
<td>Least Angle Regression (LARS)</td>
<td></td>
<td></td>
<td>(53)</td>
<td></td>
</tr>
<tr>
<td>Least Absolute Shrinkage and Selection Operator (LASSO)</td>
<td>Regularization and linear models</td>
<td>Builds model by drawing a regression line and estimates feature importances by calculating the sum of squared errors.</td>
<td>Embedded</td>
<td>(54)</td>
</tr>
<tr>
<td>LASSO LARS</td>
<td></td>
<td></td>
<td>(53)</td>
<td></td>
</tr>
<tr>
<td>Linear Regression</td>
<td></td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Low Variance</td>
<td>Statistical</td>
<td>Selects the features with the highest variance.</td>
<td>Filter</td>
<td>(49)</td>
</tr>
<tr>
<td>Minimum Redundancy Maximum Relevance (mRMR)</td>
<td>Statistical</td>
<td>Selects features that have a high correlation with the label and low correlation with each other.</td>
<td>Filter</td>
<td>(55)</td>
</tr>
<tr>
<td>Perceptron</td>
<td></td>
<td></td>
<td>Embedded</td>
<td>(56)</td>
</tr>
<tr>
<td>Perceptron RFE</td>
<td>Neural Networks</td>
<td>Iteratively trains predictive neurons and assigns feature weights depending on misclassification.</td>
<td>Embedded</td>
<td></td>
</tr>
<tr>
<td>Perceptron SBS</td>
<td></td>
<td></td>
<td>Wrapper</td>
<td></td>
</tr>
<tr>
<td>Perceptron SFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Forest</td>
<td></td>
<td></td>
<td>Embedded</td>
<td>(57)</td>
</tr>
<tr>
<td>Random Forest RFE</td>
<td>Tree Ensemble</td>
<td>Splits data and trains a separate decision tree for each split. It then averages the decision trees.</td>
<td>Embedded</td>
<td></td>
</tr>
<tr>
<td>Random Forest SBS</td>
<td></td>
<td></td>
<td>Wrapper</td>
<td></td>
</tr>
<tr>
<td>Random Forest SFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReliefF</td>
<td>Statistical</td>
<td>Selects features that have a high correlation with the label and low correlation with each other.</td>
<td>Filter</td>
<td>(58)</td>
</tr>
<tr>
<td>Support Vector Machines (linear)</td>
<td></td>
<td></td>
<td>Embedded</td>
<td>(59)</td>
</tr>
<tr>
<td>Support Vector Machines (linear) RFE</td>
<td>Support Vector Machines</td>
<td>Draws a linear decision boundary and maximises its margin. Bigger margins mean more informative features.</td>
<td>Embedded</td>
<td></td>
</tr>
<tr>
<td>Support Vector Machines (nonlinear) SBS</td>
<td></td>
<td></td>
<td>Wrapper</td>
<td></td>
</tr>
<tr>
<td>Support Vector Machines (nonlinear) SFS</td>
<td></td>
<td></td>
<td>(59)</td>
<td></td>
</tr>
<tr>
<td>Support Vector Machines (nonlinear) SFS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.1.2 Choosing predictors

The predictors were chosen to have different underlying mathematics. A thorough search for prevalence was not performed, however inspiration was taken from the work of Wu et al. (60). Their
work attempted to identify the top ten predictors used in data mining. The final choice is presented in Table 10.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Naive Bayes (NB)</td>
<td>Bayesian</td>
<td>(61)</td>
</tr>
<tr>
<td>K-Nearest Neighbour (KNN)</td>
<td>Instance</td>
<td>-</td>
</tr>
<tr>
<td>Decision Trees (DT)</td>
<td>Decision Trees</td>
<td>(51)</td>
</tr>
</tbody>
</table>

3.1.3 Choosing and pre-processing datasets

To reduce the risk of introducing bias by a particular choice of data set, the study consisted of 50 data sets from different scientific fields and preferably derived from real world measurements. The number of instances, features and classes varied, however upper limitations to these were imposed due to increased computational costs. Well known data sets were used; avoiding customisations as much as possible to conserve reproducibility. Therefore, to fit the required input type for the algorithms, the data types were limited to numerical. Categorical text, features and labels were in a few cases transformed into appropriate integers. Data sets with missing values were avoided, since these require choosing among many available imputation strategies, of which the optimal choice remains as an open research question. However, if < 5 % of all instances had missing values, the dataset was included with the instances removed. Time series were also avoided, since these require special treatment in terms of sampling. In summary, data sets fulfilled following requirements for inclusion in the study:

- has an upper limit on the number of instances and features in the order of $10^5$ and $10^2$ respectively,
- is well known, preferably highly cited in prominent journals,
- has categorical labels, in other words is a classification problem,
- consists of numerical datatypes,
- is not a time series with data points that are progressing through time,
- is not a multi-label classification task, meaning one instance can only belong to one class,
- has no or very few (< 5 %) instances with missing values.

The data sets and their corresponding information can be found in Appendix D.

3.2 Experimental Setup

The performance of 31 FSAs on 50 data sets were compared using predictions from three predictors, over which the scores were averaged. For contrast, performance was also measured without feature selection, and was included in the comparison. Implementations in Python and HPO settings of algorithms are explained in the following sections.

3.2.1 Implementation

One observation consisted of one FSA applied on one data set. The resulting subsets were used to train one predictor resulting in a performance measure. The program was written in Python, using pre-existing software packages for the algorithms. Links to these are listed in Appendix E, along
with pseudocode for part of the program summarised below. One run of the pipeline was performed in following steps, which are illustrated in Figure 8:

1. **Load and split data.** The data set is loaded from a csv-file and split into five CV folds. 80 % of each fold is used for training and the rest for testing. Each fold is stratified so the class distribution mirrors the complete data set. The training set is relayed to the FSA while the test set is kept for later evaluation.

2. **Feature selection.** The training set is processed by the FSA which outputs an array of subsets, one for each hyperparameter setting. The runtime of this process is noted and averaged over hyperparameter settings.

3. **Prediction.** The subsets are used to train three predictors, resulting in a model for each subset and their individual hyperparameter settings. The models are then applied on the test set to get predicted lables.

4. **Evaluation.** The accuracy and F-measure are calculated by comparing the predictions to the true class labels.

5. **Validation.** The best accuracy and F-measure values from each fold are saved and averaged to get the cross-validated performance score. The mean of the runtimes from each fold is calculated.

![Figure 8: Overview for one run in the pipeline. Shows what happens within each CV-fold. The Validation step happens outside the folds. Note: The number of folds, feature subsets and predictions (represented as sheets) are only illustrative.](image)

### 3.2.2 Hyperparameter Settings

Hyperparameter settings for each FSA and predictor are presented in Tables 11 and 12, with magnitudes chosen above and below the default code setting. For reasons discussed under the title ‘The choice of subset size and hyperparameters’ in section 5.2.4, we chose to have all FSAs return subsets of the same size.
Table 11: Hyperparameter settings for each FSA. The FSAs without hyperparameters are not included. \(N\) = total number of features, \(M\) = total number of instances.

<table>
<thead>
<tr>
<th>FSA</th>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaBoost, AdaBoost RFE, AdaBoost SBS, AdaBoost SFS</td>
<td>Number of inner decision stumps</td>
<td>10, 20, 40, 80, 160</td>
</tr>
<tr>
<td></td>
<td>Learning rate</td>
<td>1, 2, 4, 16, 32</td>
</tr>
<tr>
<td>DT, DT RFE, DT SBS, DT SFS</td>
<td>Number of features examined when looking for node splits.</td>
<td>(\sqrt{N}, 0.2N, 0.4N, 0.8N)</td>
</tr>
<tr>
<td></td>
<td>Minimum number of samples required for node splits</td>
<td>(1.4, \frac{M}{8}, 4)</td>
</tr>
<tr>
<td>LASSO</td>
<td>Alpha</td>
<td>0.05, 0.1, 0.25, 0.5, 1, 1.5</td>
</tr>
<tr>
<td>LASSO Lars</td>
<td>Number of iterations</td>
<td>3, 5, 10, 20</td>
</tr>
<tr>
<td>Perceptron, Perceptron RFE, Perceptron SBS, Perceptron SFS</td>
<td>Number of inner decision trees</td>
<td>10, 20, 40, 160</td>
</tr>
<tr>
<td></td>
<td>Number of features examined when looking for split</td>
<td>(\text{auto}, 0.2, 0.4, 0.8)</td>
</tr>
<tr>
<td></td>
<td>Minimum number of samples required for split</td>
<td>(1.4, \frac{M}{8}, 4)</td>
</tr>
<tr>
<td>ReliefF</td>
<td>Number of nearest miss points</td>
<td>2, 5, 10, 20</td>
</tr>
<tr>
<td>SVM, SVM RFE, SVM SBS, SVM SFS</td>
<td>C</td>
<td>(2^{-3}, 2^{-1}, 2^{-5}, 2^{-9}, 2^{13})</td>
</tr>
<tr>
<td></td>
<td>Gamma</td>
<td>(2^{-15}, 2^{-11}, 2^{-7}, 2^{-5}, 2, 2^{3})</td>
</tr>
</tbody>
</table>

Table 12: Hyperparameters for each predictor. Gaussian Naive Bayes had no hyperparameters. \(N\) = total number of features, \(M\) = total number of instances.

<table>
<thead>
<tr>
<th>FSA</th>
<th>Hyperparameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>K-Nearest Neighbour</td>
<td>Number of neighbours to consider</td>
<td>(2, 4, \frac{M}{8}, \frac{M}{2})</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>Number of features examined when looking for split</td>
<td>(\sqrt{N}, 0.2N, 0.4N, 0.8N)</td>
</tr>
<tr>
<td></td>
<td>Minimum number of samples required for split</td>
<td>(1.4, \frac{M}{8}, 4)</td>
</tr>
</tbody>
</table>
4 Results

This chapter is the presentation of the experimental results. Initially the results of the comparison are summarized, followed by more detailed presentations of predictive performance and runtimes respectively. The chapter concludes with a closer description of the comparison between FSAs for overall performance.

4.1 Summary

The result from the experiments for each performance measure consisted of a \( m \times n \)-matrix: \( m \) datasets and \( n \) FSAs, containing performance scores averaged over the three predictors. If any combination of FSA, predictor and dataset failed to produce a score, even within a single CV fold, the experiment was noted as failed for that particular FSA-dataset combination. Consequently, this led to some sparsity in the results matrix, but the magnitude was judged not to impede the comparison.

The best performing group of FSAs is presented in Table 13. Both accuracy and F-measure shared the same best performing group, all of which were embedded types. With addition to SVM RFE, this group consisted of the only FSAs that performed better than omitting feature selection entirely, the case from which the remaining 24 FSAs were indistinguishable. Runtimes revealed a different group, with types varying between filter and embedded. One FSA appeared in both of the best performing groups: Decision Tree. Two FSAs were present among the best performing within one measure while also having noteworthy high comparative performance in the other: Decision Tree RFE and Lasso.

<table>
<thead>
<tr>
<th>Predictive performance</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSA</td>
<td>Type</td>
</tr>
<tr>
<td>Random Forest RFE</td>
<td>E</td>
</tr>
<tr>
<td>Random Forest</td>
<td>E</td>
</tr>
<tr>
<td>Decision Tree RFE</td>
<td>E</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>E</td>
</tr>
<tr>
<td>AdaBoost RFE</td>
<td>E</td>
</tr>
<tr>
<td>AdaBoost</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the following sections, the background to this summary is presented in more detail. For the curious reader, summary statistics are found in Appendix F.

4.2 Comparison of predictive performance: accuracy and F-measure

To assess the distribution of the results, Tukey boxplots were used for each feature selection method and for both accuracy and F-measure (Fig. 9). The mean (red box) and median (red horizontal line) are both superimposed in addition to the underlying raw accuracies, overlaid as semi-transparent dots. All distributions exhibited large but similar interquartile ranges (IQR) and most displayed negative skew. The skewness, along with modal tendencies, suggested departure from normality, which warranted a nonparametric statistical method for significant comparison.
Figure 9: Tukey-boxplots of estimated accuracies (top) and F-measures (bottom) for each FSA, across all datasets. Overlaid raw data points are illustrated with semi-transparent red dots. Whiskers (dashed blue lines) extend to Q3 + 1.5 ∙ IQR (upper) and Q1 - 1.5 ∙ IQR (lower). Singular data points outside these limits are considered possible outliers (additional black "*"). Within the boxes, the mean (red squares) and median (red horizontal lines) across are displayed.

The null-hypothesis $H_0$ stated for accuracy and F-measure respectively:

- $H_{0,\text{Accuracy}}$: There is no difference in estimated accuracy between FSAs
- $H_{0,\text{F-measure}}$: There is no difference in estimated F-measure between FSAs
The calculated $F_F$ statistic was $F_F \approx 27.01$ and was distributed according to the $F$-distribution with 31 (upper) and 1519 (lower) degrees of freedom. The calculation depended on $k$ number of FSAs (including “No FS”) and $N$ number of data sets as $k = 32$, and $N = 50$ respectively. The tabular critical $F$-value for $F(31,1519) = 1.46$. Since $F_F > F(31,1519)$, both $H_0$, Accuracy and $H_0$, F-measure are rejected at significance level 0.05.

To find which FSA or groups of FSAs that differ, the Nemenyi post-hoc test was performed using $k$ and $N$ from earlier and tabular Nemenyi critical value $q_{0.05} = 3.78$, resulting in calculated Nemenyi critical distance (CD) at significance level 0.05 as $CD_{0.05} \approx 7.09$. The performance of two FSAs can be considered different at significant level 0.05 if their average ranks $R_j$ differ by at least $CD_{0.05}$, illustrated as FSAs with non-overlapping bars in the following graphs for accuracy and F-measure (Fig. 10).

![Figure 10: Comparison of estimated accuracies (top) and F-measures (bottom) between FSAs across all datasets. Dots represent the average ranking (AR) of each FSA, with upper and lower boundaries representing the Nemenyi critical distance (CD). The FSAs are ordered from lowest (best performance) to highest (worst performance) AR (+/-0.5 CD_{0.05}). Groups with non-overlapping bars are considered significantly different (a = 0.05).]
Random Forest RFE was observed as the top performing FSA regarding predictions. However, since the following five algorithms cannot be distinguished as different at the required significance level, a group of the best performing FSAs was determined. Both performance metrics shared the same group:

- Random Forest RFE
- Random Forest
- Decision Tree RFE
- Decision Tree
- AdaBoost RFE
- AdaBoost

Upon comparing with the case of omitting feature selection: FSAs that performed better with significance was the aforementioned group of six FSAs with addition of SVM RFE. The remaining 24 FSAs were indistinguishable from “NoFS” in Figure 10, meaning that no FSAs performed worse than not performing FS at all.

4.3 Comparison of runtimes

Analogous to the previous section, the distribution of rescaled runtimes from slowest to fastest (0-1) for each FSA on each dataset were inspected with raw data overlaying Tukey boxplots. The mean (red boxes) and median (red horizontal line) are both superimposed in addition to the underlying raw runtimes, as semi-transparent dots. Since the dispersion varied greatly between FSAs, complementary subplots were added, which divided groups of FSAs within a fitting range of values (next page Fig. 11, note that y-axes are scaled differently between plots). The observed skewness, along with modal tendencies strongly implied nonnormality. In combination with the largely varying degree of dispersion between different FSAs, a nonparametric statistical method was a justified choice over parametric for comparison.
Figure 11: Tukey-boxplots of estimated rescaled runtimes from slowest to fastest (0-1). Top: Measured and rescaled runtimes for each FSA, performed on all available datasets. Overlaid raw data points are illustrated with semi-transparent red dots. Whiskers (dashed blue lines) extend to Q3 + 1.5 ∙ IQR (upper) and Q1-1.5 ∙ IQR (lower). Singular data points outside these limits are considered possible outliers (additional black "-". Within the boxes, the mean (red squares) and median (red horizontal line) of runtimes across data sets are displayed. Bottom: The top box plots, but with FSAs divided into three separate plots with differently scaled y-axis. Note: the FSAs in all plots are sorted in descending order with respect to their bottom outlier/minimal value.
The null hypothesis $H_0$ for runtime stated:

$H_{0,\text{Runtime}}$: There is no difference in estimated runtime between FSAs.

With $k = 31$ number of FSAs and $N = 50$ number of datasets, the calculated $F_F$ statistic was $F_F \approx 89.62$, distributed according to the $F$-distribution with 30 (upper) and 1470 (lower) degrees of freedom. The tabular critical $F$-value for $F (30, 1470) = 1.46$. Since $F_F > F (31, 1519)$, $H_{0,\text{Runtime}}$ is rejected at significance level 0.05.

As with predictive performance, to discern which FSA, or groups of FSAs differ, the Nemenyi post-hoc test was performed, resulting in $CD_{0.05} \approx 6.85$ using previously defined $k$ and $N$ and tabular critical value $q_{0.05} = 3.76$. The runtimes of two FSAs can be considered different at significance level 0.05 if their corresponding average ranks differ by at least the calculated $CD_{0.05}$, which translates to FSAs with non-overlapping bars in the following plot (Fig. 12).

![Figure 12: Comparison of estimated and rescaled runtimes between FSAs across all datasets. Dots represent the average rank (AR) of each FSA across datasets, with upper and lower boundaries together representing the Nemenyi critical distance (CD). The FSAs are ordered from lowest (best performance) to higher (worst performance) AR (+/- 0.5 $CD_{0.05}$). Groups with non-overlapping bars are considered significantly different ($a = 0.05$).](image)

Low Variance Filter was observed as the fastest FSA. However, since the following six algorithms cannot be distinguished as different with confidence, a group of the fastest FSAs is presented below:

- Low Variance Filter
- Decision Tree
- Anova
- Chi Squared
- Lasso
- Perceptron
- LassoLars
4.4 Discerning best overall performance

The overall best performance was attributed to the FSA or group of FSAs that achieve among the best average ranks with respect to both predictive performance (accuracy and F-measure) and runtime. Consequently, plots presenting CD for accuracy, F-measure (Fig.10) and runtime (Fig.12) from earlier sections were compared, considering only the best performing group and the consecutive statistically indistinguishable group of FSAs (Fig.13).

As presented in Figure 13, Decision Tree (red) was present in the best performing groups of FSAs with respect to both predictive performance and runtime. Decision Tree RFE (orange) was in the best performing group regarding prediction (left and middle plot), while being indistinguishable from the majority of best performing FSAs regarding runtime (right plot). Analogously, Lasso (blue) was present in best performing group regarding runtime, while being indistinguishable from the majority of the FSAs in the best performing group regarding predictive performance.
5 Discussion

This chapter discusses the results and the resulting insights, in addition to factors of weakness and strengths in different parts of the study.

5.1 The performance of the examined feature selection algorithms

The experiment produced two top performing groups of FSAs: one for predictive performance and one for runtime. It is notable that all the top performing FSAs regarding predictive performance were of type embedded, and involved predictors that use tree structures: Decision Tree, Random Forest and AdaBoost (Table 13). The respective wrapper versions of these were also top ranked among wrapper methods, except for AdaBoost SFS. The produced models seem to include good estimations of feature importance. The predictive performances of these methods were not significantly different (Fig.10) but Decision Tree RFE was found to also be fast (Fig. 13). However, Decision Tree was the only FSA present in both best performing groups. Within the framework of this experiment, it can therefore be considered as the best performing FSA.

Apart from a few filter types, the majority of the best performing FSAs regarding speed were also of type embedded. Lasso in particular was found to be among the fastest FSAs while also having good predictive performance (Fig. 13). However, generalisations based on FSA types is approached with caution, since the number of FSAs included in the study within each category is unbalanced. This is kept in mind along with the further discussions below.

5.1.1 Runtime

It was unexpected to find any of the slowest FSAs to be of filter type (Fig. 12). Filter methods that only assess feature-to-label correlation, i.e. only assess feature relevance (Low Variance Filter, ANOVA and Chi Squared) were among the fastest of all FSA. However, two filter methods that in addition calculate feature-to-feature correlation (feature redundancy), ReliefF and CFS, were among the slowest. Apparently calculating redundancy can take a longer time than training learning algorithms.

Several FSAs had both wrapper and embedded versions. These were expected to have the same ordering of runtimes, from fastest to slowest: embedded, embedded RFE, SFS, SBS. This is intuitive since it reflects the amount of training rounds and the feature vector sizes within each round. Standard embedded only involves one training round while the number of rounds for embedded RFE depends on the size of the original feature vector. Both wrapper types have a similar number of rounds, however SFS starts with one feature and sequentially increases the number while SBS starts with all features and sequentially decreases the number. The two should be equally fast, but in our benchmark we imposed a fixed 50% restriction on subset size. This means that SBS becomes slower than SFS, since it trains on larger feature vectors. Both wrappers have several times more training rounds than the embedded types. It was therefore surprising that Random Forest and SVM had a different ordering, with wrapper versions faster than embedded: SFS, SBS, embedded, embedded RFE for Random Forest and embedded, SFS, SBS, embedded RFE for SVM (Fig.11). Although these cannot be distinguished from each other with statistical confidence (Fig. 12), it may be that it generally takes a longer time for these algorithms to extract feature importances from the model structure than it does to construct it.

The runtime results indicate that univariate (filter and embedded) FSAs in several cases are slower than multivariate methods (wrappers). Most papers consistently claim the opposite
Our results might have reflected this if data sets with higher dimensionality were included.

### 5.1.2 Predictive performance

We believe that the fixed restriction on subset sizes had large implications on the results. For instance, if the structure of a data set in question could have been sufficiently represented by less features, noisy features would have been retained and thus degrade performance. Conversely, if more than half of the features was optimal, the subset size restriction would also have degraded performance. This could have especially affected the wrapper methods since they examine feature combinations instead of individual feature importances; the experimental setup might have put the wrapper methods at a disadvantage. Other search strategies than SFS and SBS might have handled the subset size limitation better.

A noteworthy observation is that the means and medians of the predictive performances were so similar across all FSAs (Fig. 9). Despite the varied underlying mathematics of the FSAs and the different kinds of data sets involved, it is interesting that the performances were so similar. We believe that this is due to a combination of the following reasons:

- There could be larger differences in predictive performance, however since the feature subset size was fixed at 50% reduction, it minimised the likelihood for FSAs to choose largely different feature subsets. If this is the case, adding more data sets would increase the differences in performance.
- The various strengths and weaknesses of the FSAs are evened out across the problems in the experiment. The starting assumption when comparing FSAs across many data sets was: if algorithm A has a slightly better performance than algorithm B, this difference will be accentuated in proportion to how many times A and B are compared. Paradoxically, considering the No Free Lunch Theorem (64), adding more data sets would not accentuate but minimise the difference.

The first point might be solved by designing a more intricate experiment where every subset size is examined for every FSA. One approach is to add inner CV loops within each FSA that test which size is optimal, saving the results from all subset sizes for later comparison.

Although beyond the scope of this study, a way to address the second point could be to assess FSA performance in relation to characteristics of the data sets. This could be general quantitative properties such as the number of instances, number of features or feature-to-instance ratio. For a more advanced approach, Wang et al., (62) suggest a series of data set meta-features on which the basis of a FSA recommendation method could be developed. The raw data generated from our benchmark can be used to explore such an idea.

### 5.2 Method discussion

The following sections presents related research and the rationale behind the choice of method. Practical limitations, including other viable approaches and suggestions for improvement are discussed.

#### 5.2.1 Related research: other FSA benchmark studies

Several papers have been published where FSA benchmarks are performed using a similar strategy as in this thesis. Table 14 shows how many data sets, FSAs and predictors have been included as well as the chosen performance metrics. Even though the general structure is similar, the various problems and perspectives differ between the studies. Generally, there is a big difference in the
choice of data sets. Many of these studies examine FSAs for specific implementations and the data sets are chosen to reflect this.

An alternative strategy is to use synthetic data sets where all feature importances are known in advance (33). Instead of testing predictive performance these experiments examine how many relevant, irrelevant and redundant features have been selected. This has the advantage of testing FSAs without a layer of obfuscation added by predictors. Conversely, synthetic data sets do not reveal how well FSAs perform on real world data, since the structures and noise of these are difficult to mimic.

5.2.2 The custom benchmark pipeline

For an experiment with these amounts of data sets and algorithms, the combination of data set, FSA and predictor is large. It becomes prodigious when also considering combinations of hyperparameter settings. To manually initialise a separate run for each combination would be infeasible. There exists to our knowledge no platform that offers a complete pipeline to solve this problem and contains most of the algorithms listed in Table 9. Due to these reasons, it was necessary to develop the pipeline ourselves.

One specific point in which the pipeline could have been improved regards the granularity of the experiments. In our pipeline, the results were averaged over CV folds before comparing the FSAs. In retrospect, we could have saved intermediate results for each fold, allowing for comparison on a lower level. This would have revealed potential variability between folds. To our knowledge, existing platforms which automate FS do not allow for fold-wise comparison, which we could have addressed during the building stage of our pipeline. Ultimately, we kept our setup but chose a conservative statistical test to address this decreased granularity, along with reasons discussed in the following section.

5.2.3 The choice of nonparametric statistical tests

Upon inspection of side-by-side Tukey box plots (Fig. 9) of the performance, nonnormality was noticeable, such as modality and skewness. Runtimes displayed very unequal variances between FSAs (Fig. 11), which is a requirement for some parametric methods. In light of the inspection of distribution, the more conservative nonparametric test was used. A more powerful parametric
method would probably have revealed greater differences and discriminate more FSAs from each other.

5.2.4 The choice of subset size and hyperparameters

As mentioned in section 2.5 ‘Hyperparameter optimisation’, subset size can be considered a hyperparameter. The performance of each size can be tested in a grid search fashion and the one with the best performance can be picked. However, this strategy is computationally expensive since the predictors must be re-trained with each subset size, just as with other hyperparameters. We chose instead to have all FSAs return subsets of the same size, omitting 50% of the features for each data set by default. This is computationally prudent since we are not interested in the best predicting performance that can be achieved on a given dataset, but rather how FSAs compare under the same circumstances. Keeping the subset size fixed thus eliminated a potential source of variability. Unfortunately, this setting was not possible for one of the algorithms: FCBF. It has an internal filter that omits features it deems redundant and tampering with this setting would alter the algorithm in unknown ways.

Ideally, the choice of hyperparameters to include in HPO would be more tailored to the task at hand. In this work, the same range of was used on many different problems. It was therefore remarkable that no FSAs performed significantly worse than not performing feature selection at all (Fig. 10), even though the latter allows for all features to be used for prediction. Considering the crude HPO and the fixed subset size reduction, this demonstrates the utility of FS even under suboptimal conditions.
6 Conclusions

This chapter presents the conclusions of this work, recommendations for designing a benchmark pipeline and what we hope to see in future research.

A comparison of predictive performance and runtime of 31 prevalent feature selection algorithms on 50 classification problems has been performed. The results revealed a leading group of FSAs that can be used as an initial starting point for practitioners.

For predictive performance (Group 1): Random Forest RFE, Random Forest, Decision Tree RFE, Decision Tree, AdaBoost RFE, AdaBoost.

For speed (Group 2): Low Variance Filter, Decision Tree, Anova, Chi Squared, Lasso, Perceptron, LassoLars.

Within these two groups, viable choices for overall performance can include Decision Tree RFE and Lasso. Decision Tree RFE could not be distinguished from the majority of Group 2. Analogously, Lasso could not be distinguished from the majority of Group 1. Ultimately, Decision Tree is the motivated choice in terms of overall performance, since it was present in both groups.

6.1 Recommendations

In the design of a benchmarking pipeline, we recommended to start very simple and build from the beginning to the end. Examples of this is to use fast, easily understood algorithms, a simple train/test split without CV and a fraction of the data set’s instances to speed up learning. Only after a functioning output is attained should each part of the system be improved until the output is optimal. It is recommended to store the various subsets produced by the FSAs and the models constructed by the predictors instead of just their performance scores. This can be done with functions such as the pickle module in Python and greatly helps in avoiding time-consuming reruns. It is also strongly recommended to use parallel or distributed processing, since it considerably improves several aspects, such as better HPO and more reruns when mistakes are found.

6.2 Future research

When benchmarking FSAs it would be interesting add some performance metrics that were not included in this study, such as a stability index. Given a fixed subset size, the index measures how reliably a FSA pick the same features given that other factors are varied such as hyperparameters or data set splits. One such measure is the Kuncheva Index (68). Other metrics for predictive performance such as Matthews correlation coefficient or AUC (Area under the Receiver Operating Curve) might be better scores when with characteristics such as unbalanced datasets, but requires reducing multi-class problems to binary ones.

Benchmarking experiments are greatly alleviated by having access to a high number of algorithms, performance metrics and data sets without having to write bridges between different programming languages. This is especially true for practitioners or scientists whose main field in not within the machine learning discipline. Hopefully, easy-to-use, open source machine learning libraries such as Scikit-learn and the ASU feature selection repository will continue to grow. It is our opinion that this will have the greatest effect on feature selection and machine learning research and by extension, research fields that require data processing and inference.
Bibliography


17. Sullivan D. Google uses RankBrain for every search, impacts rankings of &quot;lots&quot; of &quot;them&quot; [Internet]. Search Engine Land. 2016. Available from: http://searchengineland.com/google-loves-rankbrain-uses-for-every-search-252526


50. Pearson K. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. Philos Mag Ser 5. Taylor & Francis Group; 1900 Jul;50(302):157–75. Available from: http://www.tandfonline.com/doi/abs/10.1080/1476440009463897


86. Allimoglu F. Combining Multiple Classifiers For Pen-Based Handwritten Digit Recognition. Bogazici University;


Appendix A FS definitions

In order to infer a robust definition of the feature selection problem various papers that produced a definition were examined during the pre-study on the subject. Since these definitions were presented in varying ways (some as text, some as mathematical formulas), they have been rewritten with a common mathematical notation in order to simplify comparison.

Notation:
- $J(\cdot)$: criterion/objective function
- $C(\cdot)$: cost function
- $\text{card}(\cdot)$: cardinality of a set
- $\epsilon$: limit on the value of the criterion/objective function
- $Q$: limit on the size of the selected subset
- $T$: limit on the total cost of the selected subset
- $N$: total number of features
- $S$: subset of chosen features
- $\xi$: target variable
- $\mathcal{V}$: set of indices of the features
- $\phi_j$: feature number $j$ ($j \in \mathcal{V}$)
- $\zeta$: weight
- $\Gamma$: class distribution, i.e., relative number of instances in each class

Pudil1994 [1]:
Select $\epsilon$ and $S$ so that
\[
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q
\]
\[
s.t. \ J(f(\phi_j | j \in S) - \xi) < \epsilon
\]
\[
Q < N
\]

Narendra1977 [2]:
Select $S$ so that
\[
\min J(f(\phi_j | j \in S) - \xi)
\]
\[
s.t. \ \text{card}(S) = Q
\]
\[
Q < N
\]

Molina2002 [3]:
- a) Select $S$ so that
\[
\min J(f(\phi_j | j \in S) - \xi)
\]
\[
s.t. \ \text{card}(S) = Q
\]
\[
Q < N
\]
- b) Select $\epsilon$ so that
\[
\min Q
\]
\[
s.t. \ J(f(\phi_j | j \in S) - \xi) \leq \epsilon
\]
- c) Select $\zeta$ so that a desired trade-off is made between $Q$ and $J(\cdot)$:
\[
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q
\]

Belanche2011 [4]:
- a) Select $S$ so that
\[
\min J(f(\phi_j | j \in S) - \xi)
\]
\[
s.t. \ \sum_{j \in S} C(\phi_j) \leq T
\]
\[
T \leq \sum_{j \in \mathcal{V}} C(\phi_j)
\]
- b) For as specified value $\epsilon$ of the evaluation measure $J(\cdot)$ select $S$ so that
\[
\min_{j \in S} C(\phi_j)
\]
\[
s.t. J(f(\phi_j | j \in S) - \xi) \leq \epsilon
\]

Guyon2007 [5]:
Select $\zeta$ so that a desired trade-off is made between $Q$ and $J(\cdot)$:
\[
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q
\]

George2000, Eklund2012 [6]:
Select $S$ so that
\[
J(f(\phi_j | j \in S) - \xi) = \epsilon
\]

Dash1997: [7]
Select $S$ so that
\[
\min Q + \zeta (\Gamma_{\mathcal{V}}(f(\phi_j)) - \Gamma_{S}(f(\phi_j)))
\]
\[
s.t. \ J(f(\phi_j | j \in S) - \xi) < \epsilon
\]

Kira1992 [8]:
For a specified $\epsilon$ of $J(\cdot)$ select $S$ so that:
\[
\min Q
\]
\[
s.t. \ J(f(\phi_j | j \in S) - \xi) = \epsilon
\]

Koller1995 [9]:
- a) For a specified $\epsilon$ of $J(\cdot)$ select $S$ so that
\[
\min Q
\]
\[
s.t. \ J(f(\phi_j | j \in S) - \xi) \leq \epsilon
\]
- b) Select $S$ so that
\[
\min Q + \zeta (\Gamma_{\mathcal{V}}(f(\phi_j)) - \Gamma_{S}(f(\phi_j)))
\]

Balabin2011 [10]:
Select $S$ so that
\[
\min J(f(\phi_j | j \in S) - \xi)
\]
Yusta2009 [11]:

Select $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) \\
\text{s.t.} \quad \text{card}(S) = Q
\end{align}$$

Liu2009 [12]:

Select $S$ so that some criterion of $S$ is minimised while limiting the change in classification error

$$\begin{align}
\min J_1(S) \\
\text{s.t.} \quad J_2(f(\phi_j | j \in S) - \xi) - J_2(f(\phi_j | j \in Y) - \xi) \leq \epsilon \\
\text{card}(S) = Q
\end{align}$$

Krupka [13]:

Select $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q \\
\text{s.t.} \quad Q < N
\end{align}$$

Chandrashekar2014 [14]:

Vague definition that does not clearly state what to minimise and what to constrain, so we chose to interpret it in this way.

Select $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q \\
\text{s.t.} \quad J(f(\phi_j | j \in S) - \xi) < \epsilon \\
Q < N
\end{align}$$

Fan2009 [15]:

Vague definition that does not clearly state what to minimise and what to constrain, but they mention sparse modelling and the following is the typical formulation in sparse modelling and regularised estimation. Select $\epsilon$ and $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q \\
\text{s.t.} \quad J(f(\phi_j | j \in S) - \xi) < \epsilon \\
Q << N
\end{align}$$

Fraidman2016 [16]:

Vague definition that does not clearly state what to minimise and what to constrain, so we chose to interpret it in this way.

Select $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) \\
\text{s.t.} \quad Q < N
\end{align}$$

Mohammad2004 [17]:

Vague definition that does not clearly state what to minimise and what to constrain, so we chose to interpret it in this way.

Select $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) + \zeta Q \\
\text{s.t.} \quad Q < N
\end{align}$$

Peng2005 [18]:

Select $S$ so that

$$\begin{align}
\min J(f(\phi_j | j \in S) - \xi) \\
\text{s.t.} \quad \text{card}(S) = Q
\end{align}$$
Appendix B Search String and Terminology

The following search string was used for retrieving results regarding feature selection (ssFS):

"feature selection" OR "feature reduction" OR "feature ranking" OR "attribute selection" OR "attribute reduction" OR "attribute ranking" OR "variable selection" OR "variable reduction" OR "variable ranking" OR "feature subset selection" OR "feature subset reduction" OR "attribute subset selection" OR "attribute subset reduction" OR "variable subset selection" OR "variable subset reduction" OR "selection of feature" OR "selection of features" OR "reduction of feature" OR "reduction of features" OR "ranking of feature" OR "ranking of features" OR "selection of attribute" OR "selection of attributes" OR "reduction of attribute" OR "reduction of attributes" OR "ranking of attribute" OR "ranking of attributes" OR "selection of variable" OR "selection of variables" OR "reduction of variable" OR "reduction of variables" OR "ranking of variable" OR "ranking of variables" OR "selection of feature subset" OR "selection of feature subsets" OR "selection of attribute subset" OR "selection of attribute subsets" OR "selection of variable subset" OR "selection of variable subsets"

OR "reduction of feature subset" OR "reduction of feature subsets" OR "reduction of attribute subset" OR "reduction of attribute subsets" OR "reduction of variable subset" OR "reduction of variable subsets" OR "ranking of feature subset" OR "ranking of feature subsets" OR "ranking of attribute subset" OR "ranking of attribute subsets" OR "ranking of variable subset" OR "ranking of variable subsets" OR "dimensionality reduction" OR "reduction of dimensionality" OR "dimension reduction" OR "feature subspace selection" OR "feature subspace reduction" OR "attribute subspace selection" OR "attribute subspace reduction" OR "feature subspace" OR "feature subspace selection" OR "feature subspace reduction" OR "attribute subspace" OR "attribute subspace selection" OR "attribute subspace reduction"

OR "selection of feature subspaces" OR "selection of attribute subspaces" OR "selection of variable subspaces" OR "reduction of feature subspaces" OR "reduction of attribute subspaces" OR "reduction of variable subspaces" OR "ranking of feature subspaces" OR "ranking of attribute subspaces" OR "ranking of variable subspaces" OR "sparse regularization" OR "sparse regularisation" OR "sparse regression"

The diverse terminology in machine learning and related fields can make it difficult for novice researchers to find their way around the scientific landscape. The following points are groups of terms that are used interchangeably.

- target variable, class, label, predictor, category, kind, dependent variable, criterion variable, response variable
- sample, instance, data point, row, observation
- feature, attribute, variable, parameter, predictor, characteristic, independent variable, descriptor, property
- continuous FS, ranked FS, weighted FS, univariate
- binary selection, subset selection, subspace selection, multivariate
- learning algorithm, classifier, learner, predictor, estimator
- train, learn, fit, model, model construction, model selection
B.1 Difficulties with accessing FSA prevalence

There are several factors to consider when regarding the prevalence search:

- There is a general difficulty in finding the optimal set of terms for FS since there is no standardised taxonomy and since it is applicable across fields. There is a trade-off between ensuring broad scope across disciplines while limiting the irrelevant results.

- The search string was only applied to titles, abstracts and keywords. There are surely cases where the method names are only displayed in the full text.

- Some FS methods have names that result in false positives such as “INTERACT” or “gain ratio”.

- Using ssFS when searching does not guarantee that the algorithm in question is used for feature selection and not in other tasks.

- Only searching in WoS and Scopus does not cover all papers published and the contents of scientific databases are constantly changed.

Even with all the shortcomings of this approach we still believe that it is a better way to determine which FS methods are common than what we have found in various publications. Researchers seem to pick them semi-arbitrarily from their own experience and intuition since no research paper read during the literature study had any concrete explanation for choosing the algorithms they did. In some cases, the algorithms might have been cherry picked to suit the experiment and exalt the researchers new, proposed algorithm.
## Appendix C Algorithm search results

**Table C2: Algoritms with the top 30 number of publications (hits) in WoS and Scopus**

<table>
<thead>
<tr>
<th>FS method name</th>
<th>Hits in Web of Science</th>
<th>Hits in Scopus</th>
<th>String used with ssFS (Appendix A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machines (SVM)</td>
<td>5888</td>
<td>6533</td>
<td>&quot;Support Vector Machines&quot; OR &quot;SVM&quot;</td>
</tr>
<tr>
<td>Genetic algorithms (search method)</td>
<td>2487</td>
<td>3248</td>
<td>&quot;genetic algorithm&quot; OR &quot;genetic algorithms&quot; OR &quot;GA&quot;</td>
</tr>
<tr>
<td>Decision Tree (DT)</td>
<td>2213</td>
<td>2413</td>
<td>&quot;decision tree&quot; OR &quot;decision trees&quot;</td>
</tr>
<tr>
<td>LASSO</td>
<td>1549</td>
<td>941</td>
<td>&quot;lasso&quot; OR &quot;least absolute shrinkage and selection operator&quot;</td>
</tr>
<tr>
<td>Random forest (RT)</td>
<td>471</td>
<td>857</td>
<td>&quot;random forest&quot;</td>
</tr>
<tr>
<td>Perceptron (P)</td>
<td>401</td>
<td>718</td>
<td>&quot;perceptron&quot;</td>
</tr>
<tr>
<td>Information Gain</td>
<td>366</td>
<td>642</td>
<td>&quot;information gain&quot; OR &quot;IG&quot;</td>
</tr>
<tr>
<td>Multilayer Perceptron</td>
<td>353</td>
<td>585</td>
<td>&quot;Multilayer Perceptron&quot; OR &quot;MLP&quot;</td>
</tr>
<tr>
<td>AdaBoost (AB)</td>
<td>310</td>
<td>463</td>
<td>&quot;adaboost&quot; OR &quot;ada boost&quot;</td>
</tr>
<tr>
<td>Single Radial Basis Function</td>
<td>274</td>
<td>426</td>
<td>&quot;Single Radial Basis Function&quot; OR &quot;RBF&quot;</td>
</tr>
<tr>
<td>ANOVA</td>
<td>159</td>
<td>380</td>
<td>&quot;ANOVA&quot; OR &quot;analysis of variance&quot;</td>
</tr>
<tr>
<td>Euclidean distance</td>
<td>218</td>
<td>359</td>
<td>&quot;Euclidean distance&quot;</td>
</tr>
<tr>
<td>mRMR (minimum Redundancy Maximum Relevance)</td>
<td>248</td>
<td>315</td>
<td>&quot;mrmr&quot; OR &quot;Minimum Redundancy Maximum Relevance&quot;</td>
</tr>
<tr>
<td>Correlation-based Feature Selection (CFS)</td>
<td>276</td>
<td>301</td>
<td>&quot;correlation-based Feature Selection&quot; OR &quot;CFS&quot; OR &quot;correlation based feature selection&quot; OR &quot;correlation-based fs&quot;</td>
</tr>
<tr>
<td>t-test</td>
<td>204</td>
<td>290</td>
<td>&quot;t-test&quot; OR &quot;t test&quot;</td>
</tr>
<tr>
<td>Simulated annealing (search method)</td>
<td>167</td>
<td>252</td>
<td>&quot;Simulated annealing&quot;</td>
</tr>
<tr>
<td>ReliefF</td>
<td>154</td>
<td>239</td>
<td>&quot;reliefF&quot; OR &quot;relief-f&quot; OR &quot;relief f&quot;</td>
</tr>
<tr>
<td>Sequential forward selection</td>
<td>162</td>
<td>231</td>
<td>&quot;Sequential forward selection&quot; OR &quot;SFS&quot;</td>
</tr>
<tr>
<td>Chi2</td>
<td>119</td>
<td>230</td>
<td>&quot;chi2&quot; OR &quot;chi (2)&quot; OR &quot;chi 2&quot; OR &quot;chi squared&quot; OR &quot;chi-2&quot; OR &quot;chi^2&quot;</td>
</tr>
<tr>
<td>Successive projections algorithm</td>
<td>214</td>
<td>184</td>
<td>&quot;successive projections algorithm&quot; OR &quot;SPA&quot;</td>
</tr>
<tr>
<td>Recursive Feature Elimination for Support Vector Machines (SVM-RFE)</td>
<td>142</td>
<td>162</td>
<td>&quot;SVM-RFE&quot; OR &quot;Recursive Feature Elimination for Support Vector Machines&quot;</td>
</tr>
<tr>
<td>Gain Ratio</td>
<td>73</td>
<td>116</td>
<td>&quot;gain ratio&quot;</td>
</tr>
<tr>
<td>Tabu Search (search method)</td>
<td>79</td>
<td>107</td>
<td>&quot;tabu search&quot;</td>
</tr>
<tr>
<td>Markov blanket</td>
<td>101</td>
<td>104</td>
<td>&quot;markov blanket&quot;</td>
</tr>
<tr>
<td>Uninformative variable elimination (UVE)</td>
<td>78</td>
<td>104</td>
<td>&quot;uninformative variable elimination&quot; OR &quot;UVE&quot;</td>
</tr>
<tr>
<td>MARS</td>
<td>32</td>
<td>69</td>
<td>&quot;mars&quot; OR &quot;multi angle regression and shrinkage&quot;</td>
</tr>
<tr>
<td>Fast correlation based feature selection (FCBF)</td>
<td>39</td>
<td>57</td>
<td>&quot;Fast correlation based feature selection&quot; OR &quot;FCBF&quot; OR &quot;fast correlation based FS&quot;</td>
</tr>
<tr>
<td>f-test</td>
<td>27</td>
<td>47</td>
<td>&quot;f-test&quot; OR &quot;ftest&quot;</td>
</tr>
<tr>
<td>Memetic Algorithms (search method)</td>
<td>40</td>
<td>44</td>
<td>&quot;memetic algorithms&quot; OR &quot;memetic algorithm&quot;</td>
</tr>
<tr>
<td>Stepwise multiple linear regression</td>
<td>40</td>
<td>44</td>
<td>&quot;Stepwise multiple linear regression&quot; OR &quot;MLR-step&quot;</td>
</tr>
</tbody>
</table>
## Appendix D Experimental data sets

The 'Source' column shows references to the first published work where the data sets were introduced and links to where they can be downloaded. Some original sources could not be found; however, the data sets can be found in the UCI [https://archive.ics.uci.edu/ml/datasets.html](https://archive.ics.uci.edu/ml/datasets.html) or KEEL repositories [https://archive.ics.uci.edu/ml/datasets.html](https://archive.ics.uci.edu/ml/datasets.html).

<p>| Name            | Field                  | Number of instances | Number of features | Number of classes | Source |
|-----------------|                       |                    |                   |                  |        |
| Bank_aut        | Image recognition     | 1372               | 4                 | 2                | (69)   |
| Iris            | Biology               | 150                | 4                 | 3                | (70)   |
| Mammographic    | Medicine              | 830                | 5                 | 2                | (71)   |
| Monk2           | Synthetic             | 432                | 6                 | 2                | (72)   |
| Appendicitis    | Medicine              | 106                | 7                 | 2                | (73)   |
| Ecoli           | Biology               | 336                | 7                 | 8                | (74)   |
| Leddigits       | Computer Science      | 500                | 7                 | 10               | (51)   |
| Pi_diabetes     | Medicine              | 768                | 8                 | 2                | (75)   |
| Yeast           | Biology               | 1484               | 8                 | 10               | (76)   |
| Abalone         | Biology               | 4174               | 8                 | 28               | (77)   |
| Fertility       | Medicine              | 100                | 9                 | 2                | (78)   |
| Breast cancer Wisconsin | Medicine        | 683                | 9                 | 2                | (79)   |
| Contraceptive   | Sociology             | 1473               | 9                 | 3                | (80)   |
| Breast_tissue   | Medicine              | 106                | 9                 | 6                | (81)   |
| Glass           | Chemistry             | 214                | 9                 | 7                | (82)   |
| Liver           | Medicine              | 579                | 10                | 2                | (83)   |
| Magic           | Physics               | 19 020             | 10                | 2                | (84)   |
| Page_blocks     | Image recognition     | 5473               | 10                | 5                | (85)   |
| Whitewine       | Chemistry             | 4898               | 11                | 11               | (86)   |
| Redwine         | Chemistry             | 1599               | 11                | 11               | (87)   |
| Wine            | Chemistry             | 178                | 13                | 3                | (88)   |
| Cleveland       | Medicine              | 297                | 13                | 5                | (89)   |
| Marketing       | Business              | 6876               | 13                | 9                | (90)   |
| Vowel           | Speech recognition    | 990                | 13                | 11               | (91)   |
| Leaf            | Biology               | 340                | 14                | 40               | (92)   |
| Pendigits       | Image Recognition     | 3498               | 16                | 10               | (93)   |
| Cms             | Physics               | 540                | 18                | 2                | (94)   |
| Seismic         | Geology               | 2584               | 18                | 2                | (95)   |
| Vehicle         | Image recognition     | 846                | 18                | 4                | (96)   |
| Hepatitis       | Medicine              | 80                 | 19                | 2                | (97)   |
| Band            | Materials science     | 365                | 19                | 2                | (98)   |
| Dia_retina      | Medicine              | 1151               | 19                | 2                | (99)   |</p>
<table>
<thead>
<tr>
<th>Name</th>
<th>Field</th>
<th>Number of instances</th>
<th>Number of features</th>
<th>Number of classes</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image_seg</td>
<td>Image recognition</td>
<td>210</td>
<td>19</td>
<td>7</td>
<td>(92)</td>
</tr>
<tr>
<td>Statlog</td>
<td>Image recognition</td>
<td>2310</td>
<td>19</td>
<td>7</td>
<td>(92)</td>
</tr>
<tr>
<td>Ring</td>
<td>Synthetic</td>
<td>7400</td>
<td>20</td>
<td>2</td>
<td>(93)</td>
</tr>
<tr>
<td>Twonorm</td>
<td>Synthetic</td>
<td>7400</td>
<td>20</td>
<td>2</td>
<td>(93)</td>
</tr>
<tr>
<td>Cardiotocography</td>
<td>Medicine</td>
<td>2126</td>
<td>20</td>
<td>10</td>
<td>(94)</td>
</tr>
<tr>
<td>Thyroid</td>
<td>Medicine</td>
<td>7200</td>
<td>21</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Parkinsons</td>
<td>Medicine</td>
<td>195</td>
<td>22</td>
<td>2</td>
<td>(95)</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>Physics</td>
<td>351</td>
<td>34</td>
<td>2</td>
<td>(96)</td>
</tr>
<tr>
<td>Derma</td>
<td>Medicine</td>
<td>366</td>
<td>34</td>
<td>6</td>
<td>(97)</td>
</tr>
<tr>
<td>Satimage</td>
<td>Image recognition</td>
<td>6435</td>
<td>36</td>
<td>7</td>
<td>(98)</td>
</tr>
<tr>
<td>Waveform</td>
<td>Physics</td>
<td>5000</td>
<td>40</td>
<td>3</td>
<td>(51)</td>
</tr>
<tr>
<td>Texture</td>
<td>Image recognition</td>
<td>5500</td>
<td>40</td>
<td>11</td>
<td>(99)</td>
</tr>
<tr>
<td>Heart</td>
<td>Medicine</td>
<td>267</td>
<td>44</td>
<td>2</td>
<td>(100)</td>
</tr>
<tr>
<td>Lung_cancer</td>
<td>Medicine</td>
<td>27</td>
<td>56</td>
<td>3</td>
<td>(101)</td>
</tr>
<tr>
<td>Spam</td>
<td>Text analytics</td>
<td>4658</td>
<td>57</td>
<td>2</td>
<td>(102)</td>
</tr>
<tr>
<td>Sonar</td>
<td>Physics</td>
<td>208</td>
<td>60</td>
<td>2</td>
<td>(103)</td>
</tr>
<tr>
<td>Digits</td>
<td>Image recognition</td>
<td>1797</td>
<td>64</td>
<td>17</td>
<td>(104)</td>
</tr>
<tr>
<td>Libras</td>
<td>Image recognition</td>
<td>360</td>
<td>90</td>
<td>15</td>
<td>(105)</td>
</tr>
</tbody>
</table>
Appendix E Software tools and pseudocode

E.1 Software tools and packages

Anaconda
Package manager and collection of hundreds of open source python packages.

https://www.continuum.io/downloads

Pandas
Open source library for data analysis.

http://pandas.pydata.org/

Scikit-learn
Collection of tools for machine learning and data analysis. Build on NumPy, SciPy and matplotlib.

http://scikit-learn.org/stable/

Scikit-feature
Collection of tools for feature selection. Based on Scikit-learn.

http://featureselection.asu.edu/

E.2 Pseudocode

The following pseudocode (Fig. E2.1) displays a part of the program used to generate the results for one run. One run means getting result for a single predetermined dataset and FS method.
---

**Prerequisites:**

// A mapping of names of different types of FS methods, predictors and metrics to functions.

Example:

```
CASE methodname OF:
  "Adaboost" : CALL do_adaboost  // does FS with adaboost
  "Decision Tree" : CALL do_decisionTree  // trains, predicts with Decision Tree
ENDCASE;
```

**Input:**

// the full dataset (S), consisting of N features (x) and corresponding class (y)

```
S = \{x_1, x_2, ..., x_N, y\}
```

// names required for mapping to functions and describing final results

datasetname, FSname, predictor_names, metric_names

**Output:**

`CV_score` and `total_fsRuntime` together with names of dataset, FS method, predictor and metric

---

```plaintext
begin:
  // A function in python sklearn to get data indices for k stratified CV folds, i.e. k different training/test splits (80/20) of S, each with the same class distribution as S.
  CALL stratifiedCV with Y =(y_1, ..., y_n) and k=5 RETURNING list of k trainTest_index

  // setting train/test subsets of S as 5 different folds, using the indices
  SET folds = S[trainTest_index] = \{(S'_train, S'_test), ..., (S''_train, S''_test)\}

  SET foldres = ∅
  FOR each \{(S'_train, S'_test)\} in folds:
    CALL do_FS with (S'_train, S'_test) and FSname: // using map in prerequisites
      // function do_FS
      SET a grid for iteration over all possible FS hyperparameter settings
      SET all_red_subsets = ∅
      CALL start_Timer
      FOR each params of n combinations of FS hyperparameter settings:
        CALL sklearn_FS with params RETURNING red_subset = \{(S'_train, S'_test)\}
        APPEND red_subset to all_red_subsets
      ENDFOR;
      CALL stop_Timer RETURNING seconds
      CALL sklearn_calcMetric with ...
      APPEND best_score to all_metricscores
  ENDFOR;

  SET pred_scores = ∅
  FOR each \{(S'_train, S'_test)\} in folds:
    CALL do_predict with all_red_subsets and pred_name: // using map in prerequisites
      // function do_predict
      SET a grid for iteration over possible predictor hyperparameter settings
      SET all_predY = ∅
      FOR each params of m combinations of predictor hyperparameter settings:
        CALL sklearn_createPredictor with params RETURNING predictor all_predY = \{(predictor)\}
        APPEND predictor to all_predY
      ENDFOR;

      FOR each red_subset in all_red_subsets:
        FOR each predictor in all_predY:
          CALL sklearn_fitPredict with predictor and red_subset
          RETURNING predictions of y ∈ \{(S'_test)\} as \{predY\}
          APPEND predY to all_predY
        ENDFOR;
      ENDFOR;

      SET all_metricscores = ∅
      FOR each metricname in metric_names: // using map in prerequisites
        CALL sklearn_calcMetric with metricname, all_ytest and all_predY
        // function do_calcMetric
        RETURNING score and APPEND to scores
      ENDFOR;

      GET best_score from scores
      RETURNING best_score

      APPEND best_score to all_metricscores
      APPEND all_metricscores to pred_scores
  ENDFOR;

  RETURN pred_scores and fs_durationTime to foldres

  COMPUTE CV_score as average over k folds, FOR every performance metric and predictor
  COMPUTE CV_fsRuntime as average across k folds

end;
```

*Figure E2.1: Pseudocode for a part of the program used to generate the results.*
### Appendix F Results from experiments

The summary statistics of overall estimated performance across datasets for each FSA and performance metric are presented in Table 17, in which the best values are underlined. The number of failed experiments are discerned through the sample sizes below 50.

**Table F1. Summary statistics of overall performance of each FSA across datasets.** Runtime lacks results for No FS, since no feature selection was performed prior prediction. Sample size depicts the amount of data sets included successfully (50 in total were included in the experiments, but some experiments failed). The best scores are underlined, i.e., the highest value for mean and median and the lowest for interquartile range (IQR) and average rank (AR).

<table>
<thead>
<tr>
<th>FSA</th>
<th>Sample size</th>
<th>Accuracy</th>
<th>F-measure</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Median</td>
<td>IQR</td>
<td>Mean</td>
</tr>
<tr>
<td>No FS</td>
<td>50</td>
<td>0.749</td>
<td>0.805</td>
<td>0.249</td>
</tr>
<tr>
<td>AB</td>
<td>47</td>
<td>0.788</td>
<td>0.847</td>
<td>0.234</td>
</tr>
<tr>
<td>AB RFE</td>
<td>46</td>
<td>0.785</td>
<td>0.832</td>
<td>0.244</td>
</tr>
<tr>
<td>AB SBS</td>
<td>48</td>
<td>0.753</td>
<td>0.810</td>
<td>0.236</td>
</tr>
<tr>
<td>AB SFS</td>
<td>50</td>
<td>0.728</td>
<td>0.770</td>
<td>0.288</td>
</tr>
<tr>
<td>Anova</td>
<td>50</td>
<td>0.743</td>
<td>0.797</td>
<td>0.291</td>
</tr>
<tr>
<td>CBF</td>
<td>49</td>
<td>0.731</td>
<td>0.780</td>
<td>0.295</td>
</tr>
<tr>
<td>Chi Squared</td>
<td>35</td>
<td>0.697</td>
<td>0.717</td>
<td>0.290</td>
</tr>
<tr>
<td>DT</td>
<td>50</td>
<td>0.785</td>
<td>0.834</td>
<td>0.232</td>
</tr>
<tr>
<td>DT RFE</td>
<td>50</td>
<td>0.784</td>
<td>0.837</td>
<td>0.249</td>
</tr>
<tr>
<td>DT SBS</td>
<td>50</td>
<td>0.754</td>
<td>0.810</td>
<td>0.244</td>
</tr>
<tr>
<td>DT SFS</td>
<td>50</td>
<td>0.752</td>
<td>0.815</td>
<td>0.257</td>
</tr>
<tr>
<td>FCBF</td>
<td>48</td>
<td>0.653</td>
<td>0.657</td>
<td>0.319</td>
</tr>
<tr>
<td>LARS</td>
<td>50</td>
<td>0.731</td>
<td>0.784</td>
<td>0.264</td>
</tr>
<tr>
<td>LASSO</td>
<td>50</td>
<td>0.763</td>
<td>0.810</td>
<td>0.244</td>
</tr>
<tr>
<td>LASSOOLA RS</td>
<td>47</td>
<td>0.754</td>
<td>0.806</td>
<td>0.262</td>
</tr>
<tr>
<td>Linear Regression (LR)</td>
<td>50</td>
<td>0.732</td>
<td>0.781</td>
<td>0.285</td>
</tr>
<tr>
<td>Low Variance (LV)</td>
<td>50</td>
<td>0.728</td>
<td>0.785</td>
<td>0.285</td>
</tr>
</tbody>
</table>

53
<table>
<thead>
<tr>
<th>FSA</th>
<th>Sample size</th>
<th>Accuracy Mean</th>
<th>Accuracy Median</th>
<th>Accuracy IQR</th>
<th>F-measure Mean</th>
<th>F-measure Median</th>
<th>F-measure IQR</th>
<th>Runtime Mean</th>
<th>Runtime Median</th>
<th>Runtime IQR</th>
<th>AR</th>
</tr>
</thead>
<tbody>
<tr>
<td>mRMR</td>
<td>50</td>
<td>0.739</td>
<td>0.789</td>
<td>0.282</td>
<td>20.630</td>
<td>0.692</td>
<td>21.170</td>
<td>0.936</td>
<td>0.939</td>
<td>0.070</td>
<td>20.220</td>
</tr>
<tr>
<td>P</td>
<td>50</td>
<td>0.764</td>
<td>0.815</td>
<td>0.245</td>
<td>13.920</td>
<td>0.723</td>
<td>13.570</td>
<td>0.999</td>
<td>0.999</td>
<td>0.002</td>
<td>5.680</td>
</tr>
<tr>
<td>P RFE</td>
<td>50</td>
<td>0.763</td>
<td>0.816</td>
<td>0.241</td>
<td>13.310</td>
<td>0.722</td>
<td>13.070</td>
<td>0.991</td>
<td>0.994</td>
<td>0.010</td>
<td>11.080</td>
</tr>
<tr>
<td>P SBS</td>
<td>50</td>
<td>0.739</td>
<td>0.792</td>
<td>0.265</td>
<td>21.270</td>
<td>0.693</td>
<td>21.360</td>
<td>0.914</td>
<td>0.939</td>
<td>0.076</td>
<td>20.770</td>
</tr>
<tr>
<td>P SFS</td>
<td>50</td>
<td>0.738</td>
<td>0.786</td>
<td>0.250</td>
<td>23.240</td>
<td>0.695</td>
<td>22.470</td>
<td>0.927</td>
<td>0.947</td>
<td>0.067</td>
<td>19.170</td>
</tr>
<tr>
<td>RF</td>
<td>50</td>
<td>0.800</td>
<td>0.867</td>
<td>0.234</td>
<td>2.740</td>
<td>0.766</td>
<td>2.870</td>
<td>0.931</td>
<td>0.968</td>
<td>0.093</td>
<td>18.340</td>
</tr>
<tr>
<td>RF RFE</td>
<td>50</td>
<td>0.798</td>
<td>0.847</td>
<td>0.225</td>
<td>2.460</td>
<td>0.763</td>
<td>0.258</td>
<td>2.670</td>
<td>0.572</td>
<td>0.719</td>
<td>0.665</td>
</tr>
<tr>
<td>RF SBS</td>
<td>50</td>
<td>0.761</td>
<td>0.823</td>
<td>0.250</td>
<td>14.250</td>
<td>0.720</td>
<td>13.700</td>
<td>0.939</td>
<td>0.941</td>
<td>0.076</td>
<td>20.000</td>
</tr>
<tr>
<td>RF SFS</td>
<td>50</td>
<td>0.761</td>
<td>0.816</td>
<td>0.250</td>
<td>13.670</td>
<td>0.715</td>
<td>14.050</td>
<td>0.946</td>
<td>0.953</td>
<td>0.071</td>
<td>18.560</td>
</tr>
<tr>
<td>reliefF</td>
<td>50</td>
<td>0.742</td>
<td>0.796</td>
<td>0.244</td>
<td>18.770</td>
<td>0.694</td>
<td>19.200</td>
<td>0.541</td>
<td>0.679</td>
<td>0.941</td>
<td>24.360</td>
</tr>
<tr>
<td>SVM with Linear Kernels</td>
<td>50</td>
<td>0.763</td>
<td>0.820</td>
<td>0.243</td>
<td>13.250</td>
<td>0.722</td>
<td>13.390</td>
<td>0.972</td>
<td>0.986</td>
<td>0.044</td>
<td>14.220</td>
</tr>
<tr>
<td>SVM RFE</td>
<td>50</td>
<td>0.763</td>
<td>0.816</td>
<td>0.246</td>
<td>12.330</td>
<td>0.726</td>
<td>11.710</td>
<td>0.824</td>
<td>0.915</td>
<td>0.209</td>
<td>22.800</td>
</tr>
<tr>
<td>SVM SBS</td>
<td>45</td>
<td>0.723</td>
<td>0.779</td>
<td>0.265</td>
<td>20.733</td>
<td>0.673</td>
<td>21.189</td>
<td>0.795</td>
<td>0.924</td>
<td>0.194</td>
<td>21.733</td>
</tr>
<tr>
<td>SVM SFS</td>
<td>44</td>
<td>0.727</td>
<td>0.772</td>
<td>0.248</td>
<td>20.295</td>
<td>0.677</td>
<td>20.602</td>
<td>0.844</td>
<td>0.938</td>
<td>0.193</td>
<td>20.386</td>
</tr>
</tbody>
</table>