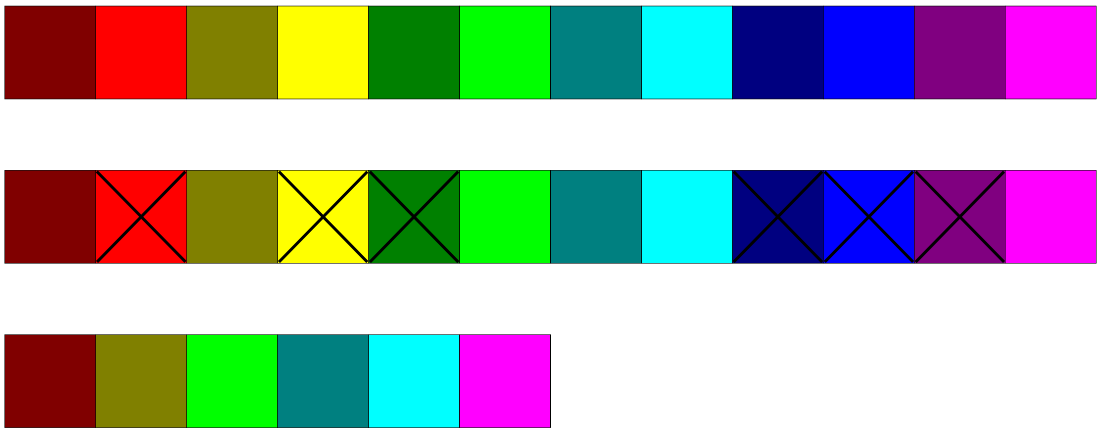
CHALMERS
UNIVERSITY OF TECHNOLOGY



Feature selection in an industrial data set

Development of a genetic algorithm for selecting categorical features

Master's thesis in Complex Adaptive Systems

Philip Andreasson

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019

Feature selection in an industrial data set

Development of a genetic algorithm for selecting categorical features

PHILIP ANDREASSON

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2019

Feature selection in an industrial data set
Development of a genetic algorithm for selecting categorical features
PHILIP ANDREASSON

Supervisor: Jimmy Stormig, Sigma IT
Examiner: Mats Granath, Department of Physics

Master's Thesis 2019
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A visualization of the feature selection problem, where a subset of the original full set of features is selected.

Typeset in LaTeX
Gothenburg, Sweden 2019

Feature selection in an industrial data set
Development of a genetic algorithm for selecting categorical features
PHILIP ANDREASSON
Department of Physics
Chalmers University of Technology

Abstract

Feature selection is a technique for reducing the dimensionality of data sets which can provide benefits in terms of computational time, performance and interpretability. This thesis presents the development of a genetic algorithm for feature selection in an industrial data set on investigations, where a large proportion of the features are categorical. The genetic algorithm is designed to always select one-hot encoded categorical features as a group. The quality of a proposed feature selection subset was assessed using Naive Bayes classifiers, decision trees, artificial neural networks, support vector machines and logistic regression classifiers. The classification performance of the subsets obtained from the genetic algorithm were further compared to stepwise forward selection, Relief, LASSO and random forests. The results showed that the dimensionality of the data set could be reduced drastically while maintaining a good classification accuracy. Most significant results were obtained for the Naive Bayes classifier, where the genetic algorithm and stepwise forward selection managed to produce subsets with prediction performances that significantly exceeded both the full data set and the subsets from the other feature selection algorithms. For the other classifiers, the differences were smaller. Given the extensive time required to run the genetic algorithm and stepwise forward selection, the other feature selection algorithms are a better choice for these classifiers.

Keywords: feature selection, genetic algorithms, categorical features

Acknowledgements

I would like to thank Jimmy Stormig for valuable discussions throughout this project and also my examiner Mats Granath for taking on this project. Also, I would like to thank Niclas Gustafsson and Alexander Ask for their time investment in setting this project up. Finally, a big thank you to Ellen for your valuable support.

Philip Andreasson, Gothenburg, December 2019

Contents

List of Figures

List of Tables

Abbreviations

SFS	Sequential forward selection
GA	Genetic algorithms
LASSO	Least Absolute Shrinkage and Selection Operator
RF	Random forest
SVM	Support vector machine
ANN	Artificial neural network
MLP	Multilayer perceptron
DT	Decision tree
NB	Naive Bayes classifier
LogReg	Logistic regression classifier

1
Introduction

During the last few decades, the amount of collected data from enterprises has increased drastically, both in terms of the number of observations that are collected and the number of variables that are treated. In 2011, the total size of copied and created data was of order 10^{21} bytes, with a prospect of doubling in size every second year [1]. The amount of sources that generate data are also increasing, with Internet of things and cloud computing becoming more prevalent [2]. These large volumes of data can be useful in a variety of fields, including commerce [3], connected cities [4] and medicine [5]. However, there are challenges that need to be addressed in order to gain knowledge from such huge data sets. The most obvious one is computational time, which can be immense for a data set with a large number of features, to the point where it is impossible for machine learning algorithms to work with the data [6]. One must also be aware of the curse of dimensionality, which is an umbrella term for bad effects that occurs in high-dimensional space due to the exponential increase in volume, which makes data sparse and difficult to handle for machine learning algorithms [7]. Finally, it is also challenging to interpret how the features and the output are related in a high-dimensional data set.

One way of tackling the challenges with big data is to try to reduce the dimensionality. Dimension reduction techniques can be divided into two categories: feature extraction and feature selection [6]. Feature extraction involves projecting the data onto a lower-dimensional space, creating new features [6]. Two of the most common methods to this end are principal component analysis (PCA) and non-negative matrix factorization (NMF) [8]. Feature selection, on the other hand, preserves the original features and aims to identify redundant and irrelevant features for the output values in the data set. Both techniques are of importance in terms of computational time and for enhancing the performance of machine learning algorithms, but feature selection also makes it possible to gain insight on how the output values relate to the original features [9–11].

For a data set that consists of p features, there are $2^p - 1$ possible feature subsets (the subset where no feature is chosen is excluded). Except for very small data sets, an exhaustive search over all possible subsets is unfeasible, creating the need for sophisticated algorithms. Feature selection algorithms are traditionally divided into three different categories: wrapper, filter and embedded methods. Wrapper methods are methods that directly take into account the performance of a classification algorithm in the feature selection procedure, while filter and embedded methods use more general information about the data set. An interesting approach in the wrapper setting is to use genetic algorithms, which is a paradigm of algorithms that starts with a random set of suggested solutions and evolve better solutions, in terms of increased classification accuracy, in a fashion that is inspired by natural selection.

In this thesis, an industrial data set on inspections will be used. The data set consists of a set of features that contain general information about the inspection object and each label describes the result of the inspection. It is of interest to investigate to what degree it is possible to predict the result of an inspection given the general features and if the prediction performance can be enhanced using feature selection. A vast majority of the features in this data set are categorical, providing a need for a feature selection algorithm suitable for this setting. Inspired by the formalism of genetic algorithms, the possibility of developing a feature selection algorithm suitable for categorical data will be explored.

1.1 Aim

The aim of this thesis is to investigate how a genetic algorithm can be constructed to handle categorical feature selection. The algorithm will be applied on an industrial data set consisting of a mixture of numerical and categorical features, where the vast majority is categorical. The new algorithm will be compared to frequently used wrapper, filter and embedded techniques in the context of the given data set through a classification study.

1.2 Limitations

One of the goals of this thesis is to provide an interpretable model in terms of the original features. Thus, the focus will be entirely on feature selection as a dimensionality reduction technique. Feature extraction methods like PCA and NMF will not be covered.

2
Feature selection algorithms

This chapter gives an introduction to the main types of wrapper, filter and embedded feature selection algorithms. Focus will be on commonly used algorithms from each of these categories and especially on genetic algorithms, which is a wrapper method that provides a clean framework for feature selection.

2.1 Wrapper methods

A wrapper method uses the performance of a classifier as its metric for evaluating candidate subsets of features. This means that the selected features will be tuned for this classifier, which makes the selection less general but can enhance prediction performance for that specific classifier [12–14]. Wrapper methods are generally computationally expensive since they require a classifier to be invoked each time the metric is calculated, but their prediction performance is believed to exceed that of filter and embedded methods [14].

Stepwise selection methods are the most common type of algorithms in the wrapper setting [13]. While these algorithms are easy to understand and easy to use, they violate some of the basic properties of statistics and have thus been criticized. A brief description of one basic stepwise algorithm and the criticism against these methods are described below. Thereafter follows the theory of genetic algorithms, which is a potentially more suitable framework for feature selection in the wrapper setting.

2.1.1 Sequential forward selection

Perhaps the most simple stepwise feature selection method is sequential forward selection (SFS), which was introduced by Whitney in 1971 [15]. This method starts with an empty set of selected features and proceeds by adding relevant features to this set, one by one. In SFS, the feature to be added is decided by the classification score of a classifier. The feature that provides the largest increase in accuracy is chosen and added to the set of selected features [16].

This is more rigorously described as follows. Assume that a data set Y consists of p features $Y = \{y_1, y_2, ..., y_p\}$. Initially, the set of selected features is empty, which is denoted as X_0. The output of the algorithm will be a set X_d, containing the d most relevant features according to the metric, where d is specified prior to running the algorithm. Denote the value of this selection metric for an individual feature y_i as $J(y_i)$. In SFS, features are selected according to

$$y_{select} = \arg\max_i J(X_k + y_i), \text{where } y_i \in Y - X_k, \tag{2.1}$$

i.e., the feature in set of unselected features that increases the value of the criterion the most is selected. After a feature y_{select} has been selected, it is added to the set of k selected features X_k, yielding a new set

$$X_{k+1} = X_k + y_{select}. \tag{2.2}$$

This procedure continues until a total of d features have been selected and the set X_d is returned by the algorithm. More advanced versions of the basic SFS algorithm have been developed. One of the most frequently used is sequential forward float selection (SFFS) which was introduced by Pudil et al. in 1994 [17]. This algorithm allows added features to be removed, if that creates a better solution [16].

2.1.2 Common criticism against stepwise selection methods

Several issues with stepwise feature selection methods have been raised by multiple authors [18–21]. The main issue concerns multiple hypothesis testing [20], which is a common issue in statistics. Most statistical tests are designed to be used for one comparison only. Traditionally, one specifies a p value, which defines the probability of observing a result that is more extreme than what was observed in the test. If the p value is low, it is likely that the test shows statistical significance. Now, consider an example of multiple testing where 50 tests are to be conducted simultaneously and where the p value of statistical significance is specified to be 0.05. The probability of observing at least one significant test is calculated as

$$P(\text{at least one significant}) = 1 - P(\text{no significant}) = 1 - (1 - 0.05)^{50} \approx 0.92. \tag{2.3}$$

In other words, there is a high probability to observe a significant result for at least one test, regardless of whether there is a significant feature in the set or not. As the number of tests approach infinity, this probability approaches 1. In stepwise feature selection, we conduct one test for each feature that has not yet been included in the selection and draw conclusions based on the improvement in test score. This multiple testing procedure can lead to erroneous inferences about a feature's significance.

2.1.3 Genetic algorithms

Genetic algorithms (GAs) have been around since the 1960's, when they were first introduced by John Holland. Lately, they have gained attention due to the new possibilities that have arisen due to increased computational power, which have increased their applicability in many areas, including feature selection [22–24]. This section gives a brief description of how the different components of a GA work and how they should be modified to fit the feature selection problem. The coverage is restricted to the components and methods that are used for feature selection in this thesis.

2.1.3.1 Initialization and encoding schemes

In GAs, the term generation is frequently used to describe the current set of candidate solutions. The first generation is usually constructed randomly. The information in a solution is encoded in a string of bits called a chromosome and each bit is referred to as a gene. Various ways of encoding information have been proposed, and two common types are binary encoding and real-value encoding. For the feature selection problem, it is natural to use binary encoding, since we can interpret a 1 as a feature being present in the selected subset and a 0 as the feature being excluded. This is advantageous, since binary encoded chromosomes are more simple to work with. [25]

2.1.3.2 The fitness function

In order to assess the quality of the feature subset represented in a chromosome, a fitness value is computed which makes it possible to compare different proposed subsets. [25] There are several ways of specify the function that computes the fitness value. For feature selection, it is of most interest to assess the classification accuracy that is achieved by a subset of features. To this end, the fitness value is computed by training a classifier and taking the test score of the classifier as the fitness value.

2.1.3.3 Selection

Selection is the step where the chromosomes that should form the next generation are chosen. Several methods for doing this exist, and the one that will be used in this thesis is tournament selection. In this method, two chromosomes are chosen at random and eventually one of them will be selected. With a specified probability p, the chromosome with the highest fitness value is selected, else the other chromosome is selected. To mimic behavior in nature, it is appropriate to choose $p > 0.5$ since this gives better solutions a larger impact on future generations. However, it is important to sometimes select the worse chromosome, in order to explore the feature space further and not get stuck in local optima. [25]

2.1.3.4 Crossover

Crossover is a powerful operation that combines parts of two selected chromosomes to construct two new chromosomes for the next generation. A splitting point is selected at random at which both chromosomes will be split. The first new chromosome is formed from the first part of chromosome 1 and the last part of chromosome 2. The second new chromosome is formed by the last part of chromosome 1 and the first part of chromosome 2. A graphical example of the crossover procedure is illustrated in Figure 2.1

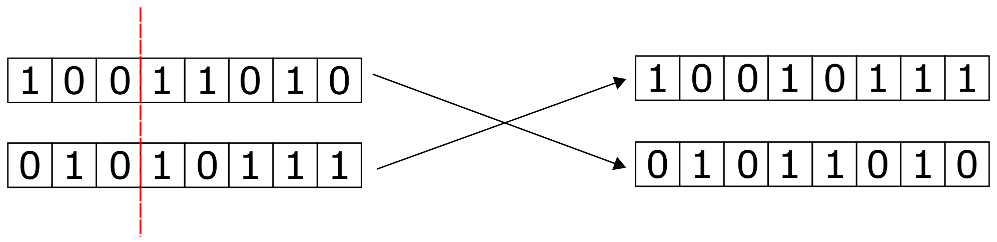


Figure 2.1: In the crossover procedure, a splitting point (red dashed line) is selected at random and two new chromosomes are formed by combining the different parts after the split.

It has been noted that this operator can be too powerful, and narrow down the search space to a local optimum. Thus, it is usual practice to only do crossover with a specified probability. In the remaining cases, the selected chromosomes are copied as they are to the new generation. [25]

2.1.3.5 Elitism and mutations

In order to not lose the best solution that has been found, one can use elitism. Elitism means that the best chromosome is directly copied to the next generation, unmodified. It is also possible to create more than one copy of the best individual and copy them to the next generation. [25]

It has been found that the performance of a GA can be enhanced by introducing mutations to the chromosomes. This is especially simple when binary encoded chromosomes are used. If a gene is mutated, its value changes from 0 to 1 or vice versa. Mutation expands the solutions' coverage in search space and plays an important role in avoiding to get stuck in sub-optimal areas. However, this is an element of randomness and should only occur with a relatively low probability, usually $p <= 0.05$. [25]

2.2 Filter methods

Filter techniques for feature selection cover both basic methods such as dropping correlated features but also more sophisticated algorithms that use general patterns in the data. No assumptions are made regarding any classification algorithm that may be used after the feature selection has been conducted.

The following sections cover different ways of measuring correlation between features. Depending on whether the features for which the correlation is calculated are both numerical, both categorical or one of each, different measures must be used. Thereafter, a more complex but common filter algorithm known as Relief, which has had great impact on the development of filter techniques, is presented.

2.2.1 Measures of correlation

The correlation between two numerical features can be calculated by the Pearson correlation coefficient, which is an intuitive and commonly used measure. If the two features are categorical, the uncertainty coefficient can be used. For the case where one feature is numerical and one is categorical, the correlation ratio is an appropriate choice. The data set that is studied in this thesis contains both numerical and categorical data and therefore, all three measures are employed. They are described in the following sections.

2.2.1.1 Pearson Correlation Coefficient

If the population distribution is known, the Pearson Correlation Coefficient (PCC) is defined mathematically for two random variables X and Y as

$$\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}, \tag{2.4}$$

where $\mathrm{Cov}(X,Y)$ is the covariance between X and Y, and σ_X and σ_Y are the standard deviations for X and Y respectively [26]. In practice, the population parameters are not known and have to be estimated from a sample. The approximation of the PCC, $r_{x,y}$ for random samples $x = (x_1, x_2, ..., x_n)$ and $y = (y_1, y_2, ..., y_n)$ can be computed as

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}. \tag{2.5}$$

Intuitively explained, the PCC computes the ratio between how much X and Y vary with each other, and how much each of the two variables varies themself. If the absolute value of the PCC is close to 1, the variables are strongly correlated. If the PCC is close to zero, they are uncorrelated.

2.2.1.2 Uncertainty coefficient

The uncertainty coefficient, also known as Theil's U, is a correlation measure for categorical features based on information theory and the concept of entropy [27]. Entropy is a measure of uncertainty for a random variable and is defined as

$$H(f) = -\sum_{i=1}^{n} P(f_i)\log_2(P(f_i)), \tag{2.6}$$

for a sample $f = (f_1, f_2, ..., f_n)$ of a feature in the data. One also defines conditional entropy between two features f and g as

$$H(f|g) = -\sum_{i=1}^{n}\sum_{j=1}^{m} P(f_i, g_j)\log(P(f_i|g_j)). \tag{2.7}$$

These concepts are used to form the uncertainty coefficient as

$$U(f) = \frac{H(f) - H(f|g)}{H(f)}. \tag{2.8}$$

The numerator in this expression is the mutual information and explains how much information about f that is gained by knowledge of feature g. Since the mutual information is divided by the entropy for feature f, the uncertainty coefficient measure the proportion of information in f that can be predicted using knowledge of feature g.

2.2.1.3 Correlation ratio

In order to measure correlations in a data set that consists of both categorical and numerical values, the correlation ratio measure has been developed. Assume that f is a numerical feature and g is a categorical feature. The mean $\bar{f}_x$ of f when the categorical feature g has the category x is defined as

$$\bar{f}_x = \frac{\sum_i f_{xi}}{n_x}, \tag{2.9}$$

where f_{xi} is the numerical value of feature f at observation i and n_x is the number of observations with category x [28]. This is used to define the mean of the entire feature f as

$$\bar{f} = \frac{\sum_x n_x \bar{f}_x}{\sum_x n_x}. \tag{2.10}$$

We use these quantities to define the squared correlation ratio between a categorical and a numerical feature η^2 as

$$\eta^2 = \frac{\sum_x n_x (\bar{f}_x - \bar{f})^2}{\sum_x \sum_i (f_{xi} - \bar{f})^2}. \tag{2.11}$$

2.2.2 Relief algorithms

The original Relief algorithm was introduced by Kira and Rendell in 1992 [29] and was formulated for the binary classification problem, where only two classes of objects exist. A refined algorithm called ReliefF was developed by Kononenko et al. in 1997 [30] to cover more general classification problems. The core of Relief algorithms is the computation of distances to the closest element that is in a different class (near-miss) and the closest element within the same class (near-hit). Kononenko et al. used the L1 distance between elements, in contrary to the L2 distance that was used by Kira and Rendell. Each feature i is assigned a weight W_i, which is updated according to

$$W_i \longleftarrow W_i - dist(O_k, \text{near-hit})_i + dist(O_k, \text{near-miss})_i, \tag{2.12}$$

where O_k is an observation chosen at random and $dist(\cdot,\cdot)$ computes the element wise L1 distance between two observations if the feature is numerical. If the feature is categorical, $dist(\cdot,\cdot)$ returns the value 1 if the feature has the same category in the two observations and 0 otherwise. The weights are initialized as zero. Expressed in words, this means that a feature gains a large weight update if the distance to the closest element with the same label is short, and the distance to the closest element with a different label is large. If a weight is large, the corresponding feature is considered to be important. The updates are calculated based on m observations, chosen at random. The procedure is illustrated in Figure 2.2.

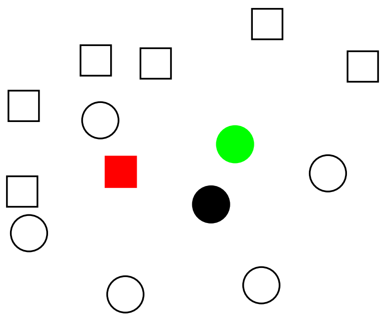


Figure 2.2: Relief selects an observation at random (black circle) and computes the element wise distance to the closest observation in the same class (green circle) and the closest observation in the other class (red square).

2.3 Embedded methods

Methods that implicitly select features in the process of another task are called embedded methods. Embedded methods typically include decision trees and penalized regression methods. This section will treat one special case of penalized regression methods, the LASSO. A common method for feature selection via decision trees is to use a random forest, which is a larger collection of uncorrelated decision trees. The following sections introduce the LASSO and random forests.

2.3.1 LASSO

Least Absolute Shrinkage and Selection Operator (LASSO) is a type of embedded method for feature selection, popularized by Tibshirani in 1996 [31]. The LASSO tackles the optimization problem

$$\hat{\beta} = \arg\min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 . \tag{2.13}$$

The first term computes the ordinary least square (OLS) estimation of the linear regression coefficients. The second term is the LASSO penalty, which constrains the size of the coefficients β via a L1 penalty. The parameter λ controls the severity of the constraint and is usually selected via cross-validation. A range of possible λ values are specified and for each λ, the LASSO coefficients $\hat{\beta}$ are calculated and a validation data set is classified. In cross-validation, the validation data is chosen at random and the procedure is repeated several times for the same λ, selecting a new validation set each time. The λ that minimizes the cross-validation error is selected. The difference between the LASSO and the OLS in a 2-dimensional setting is illustrated in Figure 2.3. The figure shows how the OLS coefficients are shrunk to fit the LASSO constraint, causing one of the coefficients to become 0.

The feature selection point of view of the LASSO can be accessed by the values of the $\hat{\beta}$ coefficients. Larger values of a coefficient means that the corresponding feature is important. One can specify a threshold for which variables that should be considered selected by the LASSO. In the example in Figure 2.3, the feature corresponding to β_1 is excluded from the relevant feature subset since its value is shrunk to 0.

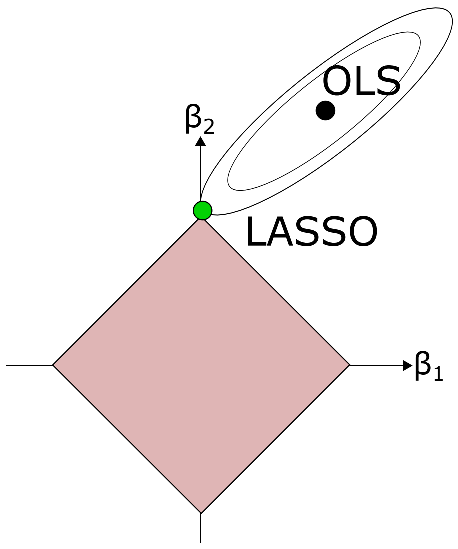


Figure 2.3: In the LASSO, the β coefficients obtained from an OLS estimation are shrunk to fit a constraint on the sizes of the coefficients. In the case shown, β_1 is shrunk to 0 and interpreted as excluded from the subset selection.

The LASSO must use one-hot encoded data, which means that a categorical feature f will be represented by $n_f - 1$ indicator columns, where n_f is the number of categories in the feature f. The category that is present in an observation is represented by the value 1 and the other columns have the value 0. This creates a risk that only some of the categories of a categorical feature will be selected. Several variants of the original LASSO algorithm have been proposed, including the Group LASSO, which was developed by Yuan and Lin in 2006 to resolve this issue [32]. In Group LASSO, a number of groups are specified prior to running the algorithm, where each group is considered together during the feature selection procedure. Each group corresponds to the categories of a categorical feature. The numerical features are treated as a group with a single member.

2.3.2 Random forests

A random forest is another type of embedded feature selection method that consists of a collection of uncorrelated decision trees [33]. Decision trees are described below in Section 3.4. They can be summarized as a sequence of variable splits that creates different branches which results in a classification at a leaf node, which is a point where the branching ends. The idea behind random forests is to make many trees classify the same data and take the majority vote as the final classification result.

In order for this to be useful, it is necessary that the trees are uncorrelated. The feature selection is achieved during the construction of the random forest.

Suppose that the data set consists of n samples. The random forest algorithm will construct a collection of N decision trees based on N bootstrap samples with replacement of size n from the original sample. Assuming that the data set consists of p features, the next step is to select a random set of $m < p$ features that are candidates for splitting. Using only a subset of the features as candidates for splitting helps to make the collection uncorrelated. Among the m features, the best splitting feature is decided and the tree is split at that location. The construction continues in this fashion with a new set of m features until the tree's stopping criterion is reached. A common choice of stopping criterion is the maximum depth of the tree, which helps avoid creating too complex trees that are prone to overfitting. Several trees are grown analogously to create the random forest.

A common measure of feature importance is the decrease in node impurity [33]. Node impurity is calculated using the Gini index, as

$$\sum_{k=1}^{K} \hat{p}_k(1 - \hat{p}_k), \tag{2.14}$$

where K is the number of classes and $\hat{p}_k$ is the proportion of class k at a specified node. If there is only one class present at a node, the Gini index obtains the value 0, which means that the node is pure. If a split at a feature leads to a large decrease in node impurity, this feature is important for partitioning the feature space into different class labels. The Gini index above is defined for a single decision tree. In a random forest, the decrease in node impurity for each feature is averaged over all trees in the forest.

3
Classification Algorithms

Feature selection efficiency is dependent on the classification algorithm used. As described above, wrapper methods directly utilize classification algorithms for choosing the best subset of features, yielding a selection that is tuned for a specific classification algorithm. Depending on the properties of the classifier, different selections may be obtained. In this thesis, five commonly used classification algorithms will be considered: artificial neural networks, support vector machines, the Naive Bayes classifier, decision trees and logistic regression. These are used for selecting features in the wrapper algorithms, as well as for evaluating the performance of the final subset selections for all feature selection algorithms. These classifiers are described briefly in the following sections.

3.1 Artificial neural networks

An artificial neural network (ANN) is a computation method, based on a network of simple computational nodes, called artificial neurons, that are connected through weighted edges [34]. The neurons are usually depicted in different layers, as illustrated in Figure 3.1. A basic type of an ANN is the multilayer perceptron (MLP), which consists of layers of neurons where all neurons in one layer are connected to every neuron in the following layer. The first layer is the input layer that reads the input signals and passes them on to the next neuron layer. Although the input layer does not consist of neurons, they are usually illustrated in the same way as neurons are. Each connection in the network has an associated weight and each neuron has a bias, which are parameters that are tuned during the training. The final layer is the output layer, from which interpretations of the network's prediction are made.

3.2 Support vector machines

A support vector machine (SVM) aims to optimize a hyperplane that separates different classes in the feature space [35]. Support vectors utilize the data points that are closest to the decision boundary that separates two classes, which are the points that have impact on the hyperplane's orientation. An SVM finds the optimal hyperplane in the sense that the distance between the tips of the support vectors from each class is maximized. This is illustrated for the binary classification problem in Figure 3.2.

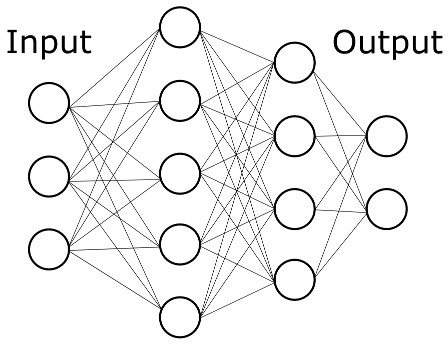


Figure 3.1: A simple ANN with two hidden layers of neurons. Each edge between neurons has an associated weight and each neuron has a bias, which are parameters that are fitted to training data. The prediction result is read from the output layer values.

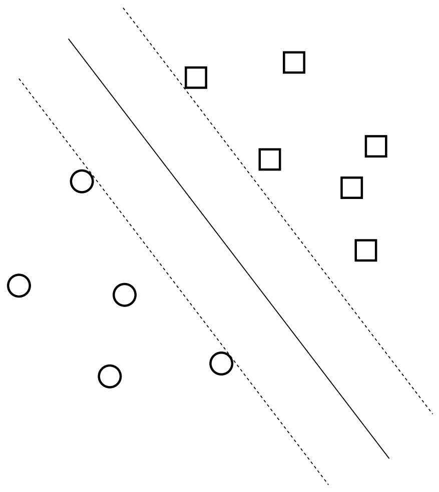


Figure 3.2: A support vector machine uses support vectors defined by the observations that are closest to the decision boundary to create supporting hyperplanes (dashed lines). These hyperplanes are used to find an optimal hyperplane (thick line) to separate different classes.

3.3 Naive Bayes classifiers

The Naive Bayes (NB) classifier is based on Bayes theorem,

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \tag{3.1}$$

where $X = (x_1, x_2, ..., x_n)$ is the feature vector and y is the label [36]. One strong assumption that the Naive Bayes classifier makes is full independence between features. It is from this “naive” assumption that the classifier got its name. If this assumption is made, the conditional probability $P(X|y)$ can be written as

$$P(X|y) = P(x_1, x_2, ..., x_n|y) = P(x_1|y)P(x_2|y)...P(x_n|y). \tag{3.2}$$

The denominator in Bayes theorem can be ignored when predicting the class label, as it does not depend on y. The predicted class label by the Naive-Bayes classifier is the label that maximizes the numerator in Equation 3.1, which is formulated as

$$y = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y). \tag{3.3}$$

If the features are categorical, it is straightforward to calculate the probabilities by simply counting the number of occurrences as

$$P(x_i|y) = \frac{\#(x_i, y)}{\#y}. \tag{3.4}$$

If the features are real-valued, an assumption on the feature distribution must be made. It is common to assume normal distribution, which means that the probabilities are computed as

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right). \tag{3.5}$$

3.4 Decision trees

A decision tree is a simple classifier that provides an intuitive perspective of how the classification is made. In the construction of a decision tree, a set of optimal splits in the data is decided. The optimal split is found by first finding the best splitting point for each feature in the data set. Several classification criteria exist for determining the best splitting point for a feature. A simple and common criterion to use is the Gini index, defined as

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{3.6}$$

where $\hat{p}_{mk}$ is the proportion of observations of class k in the candidate splitting point in node m [33]. The candidate that results in the greatest decrease in the Gini index value is chosen as splitting point. The difference is calculated as the difference between the current node's Gini index and all children nodes' Gini indices. An example for the case of two features in a binary classification problem is illustrated in Figure 3.3. Each node in the tree corresponds to a region in feature space. As is seen in the figure, one split has been made which created two new regions. The leftmost region in the figure corresponds to a node with Gini index 0, since there is only one class present in that region. The construction continues with new splits until a stopping criterion is reached. A common stopping criterion is a specified maximum depth of the tree, which helps to avoid overfitting.

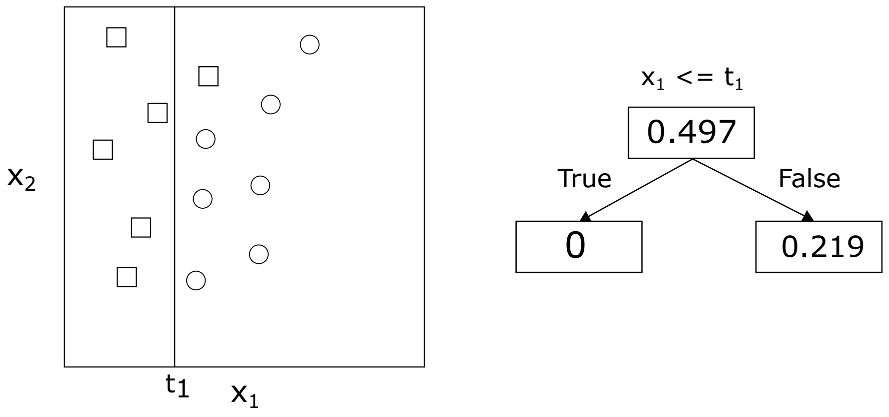


Figure 3.3: A decision tree after one split. An optimal splitting point t_1 was found in feature x_1. The partitioning of the feature space after the split contains one region which is pure since its Gini index is 0.

3.5 Logistic regression classifiers

In logistic regression, a logistic function (commonly called a sigmoid function) is employed, which transforms the variable values to be in the the range [0,1] [33], via the function

$$f(x) = \frac{1}{1 + \exp(-x)}. \tag{3.7}$$

In the basic formulation of logistic regression, binary classification is assumed. Denote the probability of an observation belonging to class 1 as $p = P(Y = 1|X)$. The logistic regression formula for p is obtained by feeding the standard linear regression equation

$$y = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_n x_n, \tag{3.8}$$

into the sigmoid function, yielding

$$p = P(Y = 1|X) = \frac{1}{1 + \exp(-(\hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_n x_n))}. \tag{3.9}$$

The logistic regression parameter vector $\vec{\hat{\beta}} = (\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_n)$ is fitted to training data using Maximum-Likelihood Estimation (MLE) [33].

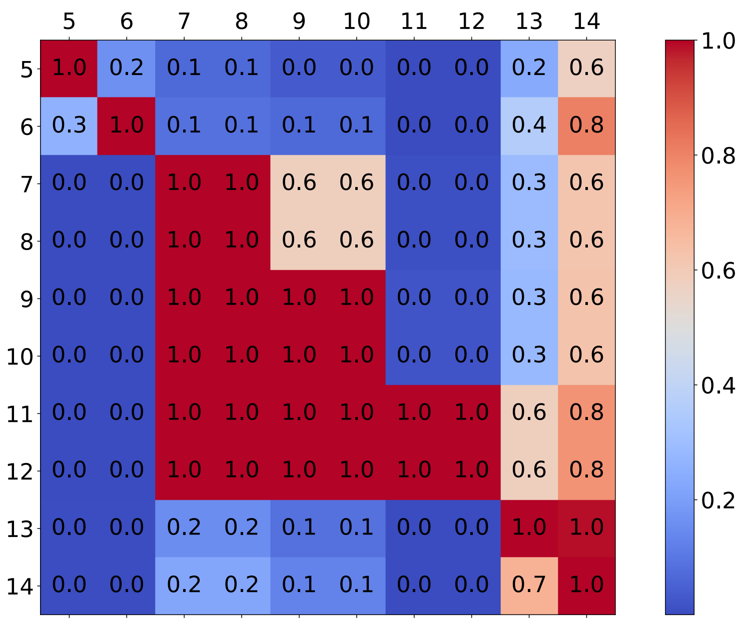
4
Method

The raw data set that is collected from a database consists of 107 000 observations and 25 features, where 21 of these features can be viewed as categorical. In order to be able to work with the data and to get useful results, it is necessary with preprocessing of the data, which is one of the most important steps in the work procedure. It includes handling missing values, transforming features into more convenient forms, handling small categories of categorical features and combat issues with text features that have been entered manually. The steps taken to this end are described below. Thereafter, the design of the GA and the experimental procedure is explained. The experiments include running the wrapper, filter and embedded methods described in the theory section.

4.1 Data preprocessing

Correlations were mentioned as a filter technique in the theory section and due to its generality, it is often included in the preprocessing of the data. Figure 4.1 shows a partition of features that are fully correlated. The correlation values have been calculated using Pearson's correlation criteria, the uncertainty coefficient and the correlation ration, described in Section 2.2.1. Full correlation means that no additional information about the data is gained by including more than one of the correlated features. Instead, the additional features are sources of undesirable noise that could provide difficulties for the machine learning algorithms to classify the data. They are also contributing to an increased execution time of the programs. The first step is to remove all features that are close to fully correlated to another feature. Features that are not shown in the figure have lower correlation rates and are not affected by the preprocessing at this step.

In some cases, it can be beneficial to transform a feature into a different form. An example of this can be found in one of the features that specify the identification number of the care firm that handles the object. It is known that many objects do not have any care firm and these entries are represented by empty values. Since this is a decision that the company has made, these empty values should not be considered to be missing values where the data is unknown. However, there is a need to handle the empty values in order to use those observations in the feature selection algorithm. This is achieved by transforming this feature to a boolean feature that specifies whether or not an object has a care firm. By doing this, information about the care firm identification number is lost, but it allows the usage of a larger



	5	6	7	8	9	10	11	12	13	14
5	1.0	0.2	0.1	0.1	0.0	0.0	0.0	0.0	0.2	0.6
6	0.3	1.0	0.1	0.1	0.1	0.1	0.0	0.0	0.4	0.8
7	0.0	0.0	1.0	1.0	0.6	0.6	0.0	0.0	0.3	0.6
8	0.0	0.0	1.0	1.0	0.6	0.6	0.0	0.0	0.3	0.6
9	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.3	0.6
10	0.0	0.0	1.0	1.0	1.0	1.0	0.0	0.0	0.3	0.6
11	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6	0.8
12	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	0.6	0.8
13	0.0	0.0	0.2	0.2	0.1	0.1	0.0	0.0	1.0	1.0
14	0.0	0.0	0.2	0.2	0.1	0.1	0.0	0.0	0.7	1.0

Figure 4.1: Correlation matrix for a partition of features in the data. Some features are fully correlated to each other and should therefore be removed.

proportion of the data, which is more important in this case.

Two features contain date-time information which must be transformed in order to be usable. This is achieved by transforming them into numerical features, where the number of days that differ between the entry and a reference date is calculated. The choice of reference date is arbitrary and is in this case chosen as the date of data retrieval from the database. All numerical features are thereafter scaled using the `scikit-learn` StandardScaler function in Python [37], which sets the mean of that feature to 0 and the standard deviation to 1. Standardizing numerical features is essential for achieving good performance, since classifiers otherwise struggle with comparing features that have different units, which is the case in general.

A majority of the features in the data set is of categorical nature, and some features can obtain a large number of different values. For instance, one of features in the data set is categorical with roughly 9000 different possible categories. Simply using one-hot encoding on this feature would give 9000 columns, where the number of available observation for each individual category would be too small to make inferences. A proposed way to handle this is to bundle up the categories in an appropriate way, to create larger categories. However, it is not always possible to find a suitable way to bundle up categories due to the characteristics of the features. In those cases, a threshold for the number of occurrences that a category needs to have is specified. Observations that contain a category that falls below this threshold are removed.

One feature consists of manually entered text strings, which specifies the name of the manufacturer of the inspection object. There is a lack of convention on how these values are entered, giving rise to many different categories that represent the same manufacturer. A simple text processing algorithm can be employed to partially resolve this issue. The algorithm removes white space, certain keywords that are chosen from manual inspection of the data, case-sensitivity and special characters that are not numbers or letters. This greatly reduces the number of categories, but there might still be misspelled words that are not captured from this algorithm. These are however few and for simplicity, these observations are removed.

The data consists of many different groups of inspection objects and it is a reasonable hypothesis that there are differences in which features that are important between these groups. Therefore, the data set is split into groups that contain only one object group and each group is treated separately. For simplicity, only the two largest groups of objects will be treated further in this thesis. The largest group contains information from elevator inspections and the second largest group contains information about gate inspections.

After the preprocessing, the data set consists of the columns described in Table 4.1. The categories column shows the number of categories in the elevator and the gates group respectively. Note that feature 1, 8 and 9 are constants, since they are categorical variables with only one category after the preprocessing. These could be removed at this stage, as an additional preprocessing step. However, by including them, it is possible to test whether the feature selection algorithms manage to remove them.

Number	Feature name	Type	Categories
1	EmployeeCompanyId	Categorical	1/1
2	EmployeeSectionId	Categorical	7/6
3	InspectionObjectMonth	Categorical	12/12
4	InspectionObjectSiteId	Categorical	18/11
5	InspectionObjectStatus	Categorical	2/2
6	InspectionObjectIsCompanyObject	Categorical	2/2
7	InspectionObjectTypeId	Categorical	7/2
8	InspectionObjectGroupId	Categorical	1/1
9	InspectionObjectCategoryId	Categorical	1/1
10	InspectionObjectLocationCity	Categorical	15/10
11	Manufacturer	Categorical	5/5
12	Year of manufacturing	Numerical	-
13	ServiceDefintionId	Categorical	3/2
14	ServiceDefintionServiceInterval	Categorical	2/2
15	ServiceDefinitionIsRecurring	Categorical	2/2
16	ActivityResultTimeSpent	Numerical	-
17	HasInspectionObjectCareFirm	Categorical	2/2
18	DaysSinceEmployeeStartDate	Numerical	-
19	DaysSinceActivityResultCompleted	Numerical	-

Table 4.1: The features in the data set after preprocessing.

4.1.1 Class imbalance

In addition to preprocessing of the features, it is also necessary to consider the labels in the data set. Four different labels exist: approved, approved with remarks, not approved and not completed. Roughly 81% of the inspections in the raw data set results in approval, 14% in approved with remarks, 2.6% not approved and 2.4% not completed. This means that the data sets suffers from severe class imbalance. If a classifier would be trained on this data, cases that result in approval would dominate the training and it would be difficult for classifiers to capture the characteristics of the smaller classes due to their low prevalence, yielding poor performance. It is therefore necessary to handle this imbalance in order to create a classifier that does not only classify entries as approved. The most simple solution is to undersample the most frequent classes by removing a proportion of observations that yield approved and approved with remarks. This could however remove important observations from the data, especially in this case where the imbalance is so severe that it would require a majority of the observations to be discarded. Another possibility is to use weighted training, where entries from the smaller classes have more impact on the training. While this works in theory, problems still arise when the data is randomly split into training, validation and test sets. Each set may only contain a few entries of the smaller classes, which creates numerical problems.

In order to achieve as good results as possible for the data set at hand, the classification problem will be transformed into a binary classification problem, where everything that is not approval is bundled together in the same class. This is based on the hypothesis that objects which are not fully approved show similar characteristics. The approved class is then slightly undersampled, to create a data set where 60% of the labels are approvals and 40% are the other possibilities.

4.1.2 Ethical considerations

The raw data set contains information about individual employees in terms of their start date and an identification number. In order to not present results that could lead to consequences for these individuals, the information about their identification number is dropped from the analysis entirely. Information about their start date gives an idea of the experience of the employee, which could impact the results and provide useful information. It is thus possible to consider the experience of the employee as a feature without being able to directly identify individuals.

4.2 Designing the genetic algorithm

As was stated in Section 2.1.3.1, binary encoding is a suitable choice for GAs in the feature selection problem. In order to handle categorical data, the decoding procedure for the chromosomes uses a map data structure that takes the feature index as key and returns the corresponding columns in the one-hot encoded version of the data set, to make sure that all categories of a categorical feature are selected. This is an approach similar to what is used in the Group Lasso, where groups are pre-specified and always selected together. The decoding procedure is illustrated in Figure 4.2. When the fitness value of a chromosome is to be computed, all genes that carry the value 1 are used as keys in the map. A classifier is then trained and tested and the test score is used as fitness value. This is the only part of the algorithm that uses one-hot encoded data. The other components of the GA are implemented as described in Section 2.1.3 and the hyperparameters, presented in Table 4.2, are chosen based on recommendations in the literature [25].

Parameter	**Value**
Population size	50
Tournament selection probability	0.9
Tournament size	2
Mutation probability	0.05
Number of elitism copies	1
Number of generations	200
Number of runs	100

Table 4.2: Hyperparameters used for the GA.

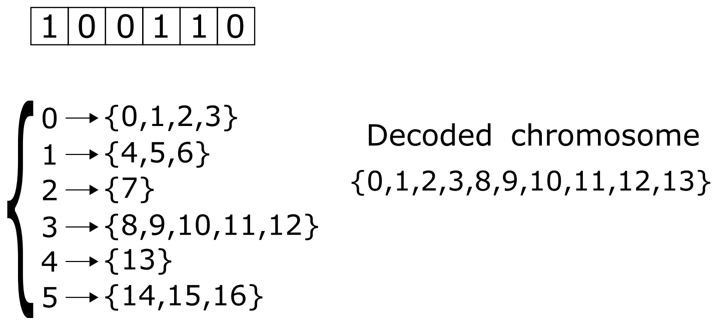


Figure 4.2: Example of the decoding procedure of a chromosome. The corresponding list of encoded column indices for each active gene in the chromosome is collected to yield a list of column indices to include in the computation of the fitness value.

4.3 Experimental procedure

This section describes how each of the feature selection algorithms described above is used to select features in the data set.

The SFS algorithm is executed with each of the five classifiers over the full range of features, making it possible to study how the test score evolves as more features are included. Since the classifiers require one-hot encoded data, a modification of the original SFS algorithm is necessary in order to handle categorical features. The same technique that is used for the GA is employed, where all one-hot encoded columns belonging to a categorical feature are tested for inclusion together.

The initialization of the chromosomes in the GA typically introduces a bias towards certain areas of the feature space. This can have a significant impact on the subsequent generations. To reduce this effect, the algorithm is executed 100 times and the selected feature subset from each execution is recorded. By counting the number of times that each feature is selected in the best chromosome at the end of the execution, the importance of each feature can be estimated. It is plausible to believe that a feature that is present in 90% of the cases is more important than a feature that is present in 70% of the cases, although both may be considered important for describing the model. To construct the feature subset, features are added sequentially in descending importance order and the corresponding classifier is trained and tested to estimate how much the dimensionality of the data set can be reduced.

The experimental procedure investigates the binary classification problem, which means that the original Relief algorithm may be used. It is however used with the L1 distance measure, as was described in Section 2.2.2. Relief is able to handle categorical features without the need of numerical encoding. In this case, the distances for the weight updates obtain value 0 if the categories are the same, and 1 otherwise. The number of weight updates m is chosen to be 1000.

LASSO requires numerical encoding of categorical features, which is achieved by one-hot encoding. In order to select all categories of a categorical feature, the Group LASSO algorithm is used. Although all categories are selected as a group during training, Group LASSO will return an importance value for each category which can differ slightly between categories. The common score for the categorical feature is calculated as the average score of the categories.

Random forests are able to handle categorical features without numerical encoding. The number of trees in the forest is specified to be 150, which is the standard choice in the `scikit-learn` implementation [37]. Further, a maximum of 20 splits is allowed for each tree and 10-fold cross validation is used to avoid overfitting the trees.

The features in the best subset obtained from each feature selection algorithm is extracted from the full data set and the classifiers are trained and tested using this data. The test scores are compared to the score obtained from the full data set from each classifier.

5
Results

This chapter presents the results obtained by the different feature selection algorithms on the two partitions of the data. First, the largest group of objects, elevators, will be presented. Thereafter follows a more brief presentation of the results for the second largest group of objects, gates. All features in this section are referred to by their index, which are explained above in Table 4.1.

5.1 Sequential forward selection results

Figure 5.1 shows the test score obtained against the number of features included in the data set for the SFS algorithm. The Naive Bayes classifier has a range of roughly optimal test scores for 3-10 features included, whereas the other classifiers have a roughly constant score from 3 features all the way to the full data set. The performance of the Naive Bayes classifier deteriorates when the number of features approach the full data set.

5.2 Genetic algorithm results

Figure 5.2 illustrates the evolution of solutions found by the GA using a Naive Bayes classifier. The thick green line shows the test score obtained by the best chromosome found at each generation and the thick blue line shows the average test score of all chromosomes at each generation. The plots are obtained by averaging over 100 individual runs of the algorithm. The faded lines show the test scores from each individual run. The variances of both the average and the best solution between individual runs are large, which illustrates the need of running the GA multiple times in order for the results to be reliable.

Rerunning the GA several times makes it possible to get an idea of the relative importances of the features by counting how frequently they occur in the best chromosome, which provides the information needed to select the best features. Figure 5.3 illustrates the proportion of occurrence for each feature in the data set using a Naive Bayes classifier to compute the fitness values. The feature enumeration is explained in Table 4.1. It is noted that the numerical features 12, 16, 18 and 19 are all important. The features 1, 8 and 9 are constants, as shown in Table 4.1, and are all given quite large importance, which is undesirable as they do not provide any qualitative information to aid the classification procedure.

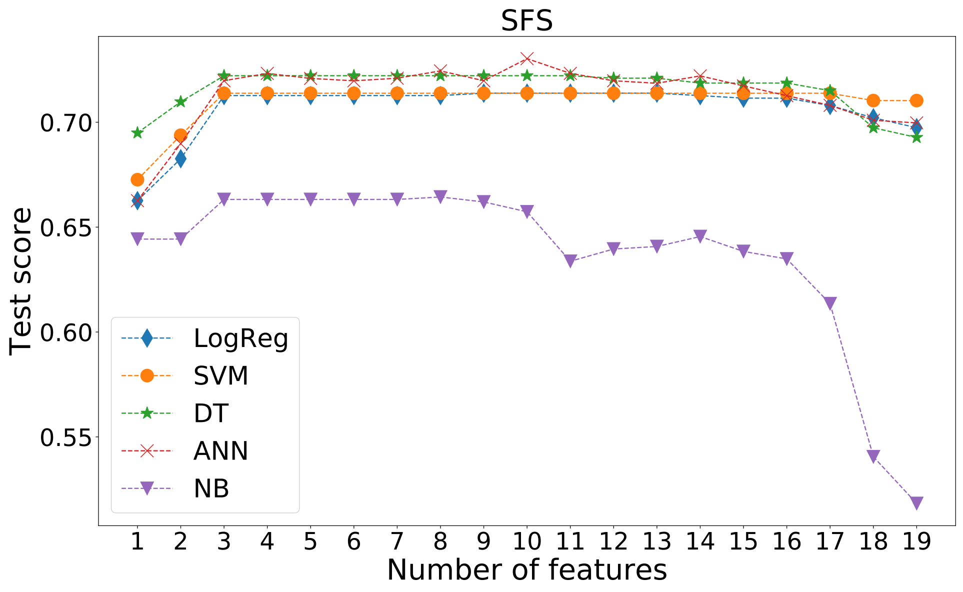


Figure 5.1: The test score obtained from feature subsets created by the SFS algorithm for the elevator data. The Naive Bayes classifier has a range of 3-10 features where it performs well, but when the number of features approach the full data set, the performance is significantly reduced. The other classifiers perform increase their test scores until 3 features are included in the set, after which the performance is roughly constant.

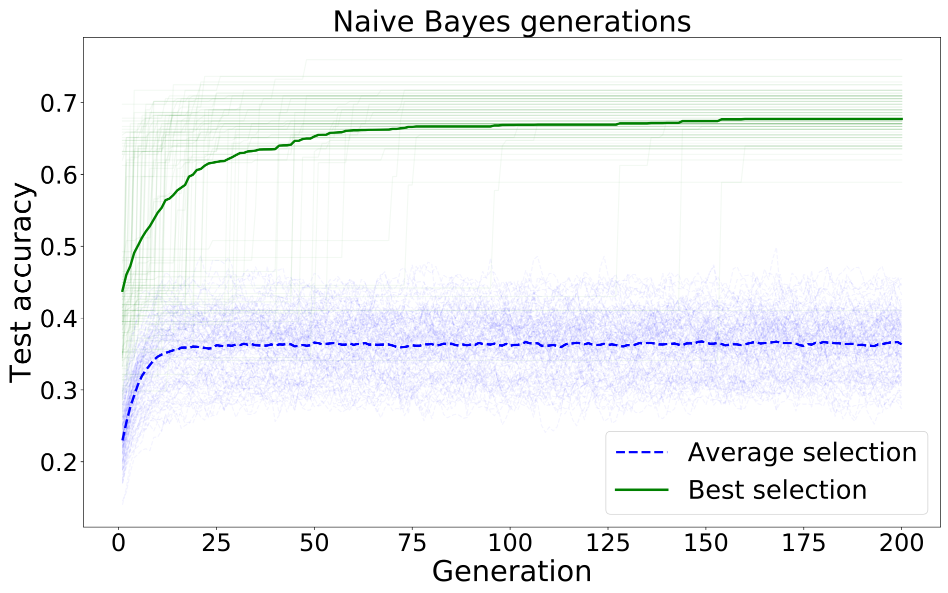


Figure 5.2: The evolution of the best chromosome (thick green) and of the average of all chromosomes (thick blue) as a function of the number of generations averaged over 100 runs of the GA, where a Naive Bayes classifier is used. The faded green and blue lines illustrate the results of each individual run. The variance between runs is large for both the average chromosome and the best chromosome.

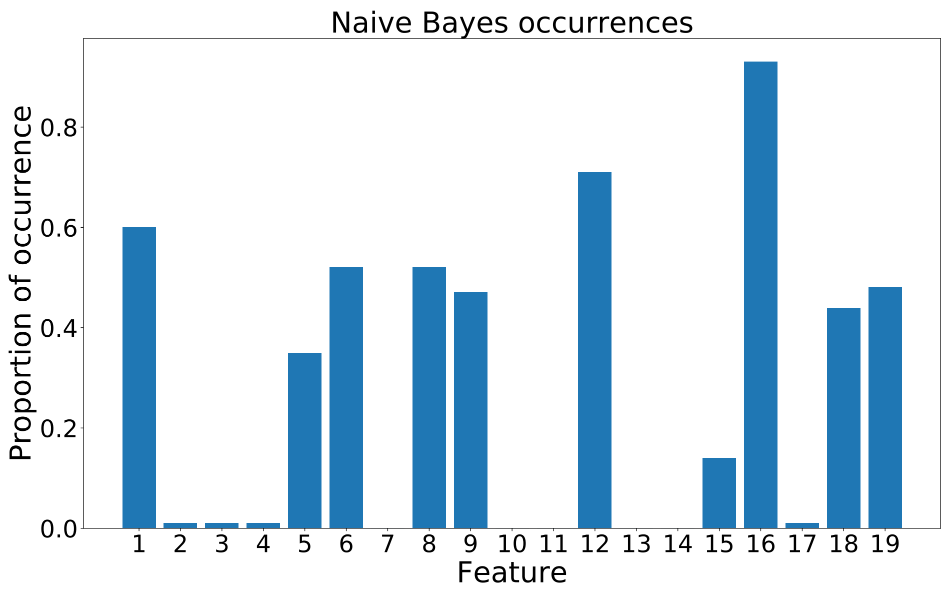


Figure 5.3: The proportion of occurrence in the best chromosome for each feature in the data set obtained by the GA with a Naive Bayes classifier for the elevator data.

The same experimental procedure is used with the other four classifiers. Figures 5.4, 5.5, 5.6 and 5.7 show the results obtained with a decision tree classifier, a logistic regression classifier, an ANN classifier and an SVM classifier respectively. Compared to the Naive Bayes results, there is more ambiguity in which features that are important for these classifiers. However, for the decision tree classifier in Figure 5.4, it is clear that features 12 and 19 are not of importance and in Figure 5.6, feature 11 is less important for the ANN. It is notable that the excluded features differ, which illustrates the tuning towards the classifier that is a general property for wrapper methods. As was the case for the Naive Bayes classifier, the constant features 1, 8 and 9 are given large relative importance for the four other classifiers.

5.3 Relief results

The weights obtained from the Relief algorithm provide a measure of relative importance and is presented in Figure 5.8. Note that this measure is different from the one that was used for the GAs and it is therefore not possible to compare the values on the y axis. The weights are normalized to give the most important feature the value 1.0. Contrary to the GA, the Relief algorithm manages to filter out the constant features 1, 8 and 9. The Relief algorithm gives large importance to the numerical features 12, 16, 18 and 19, and also primarily the categorical features 7 and 11. Overall, the distinction between relevant and irrelevant features is more evident for the Relief algorithm than in the GA, since many weights are 0 in this case.

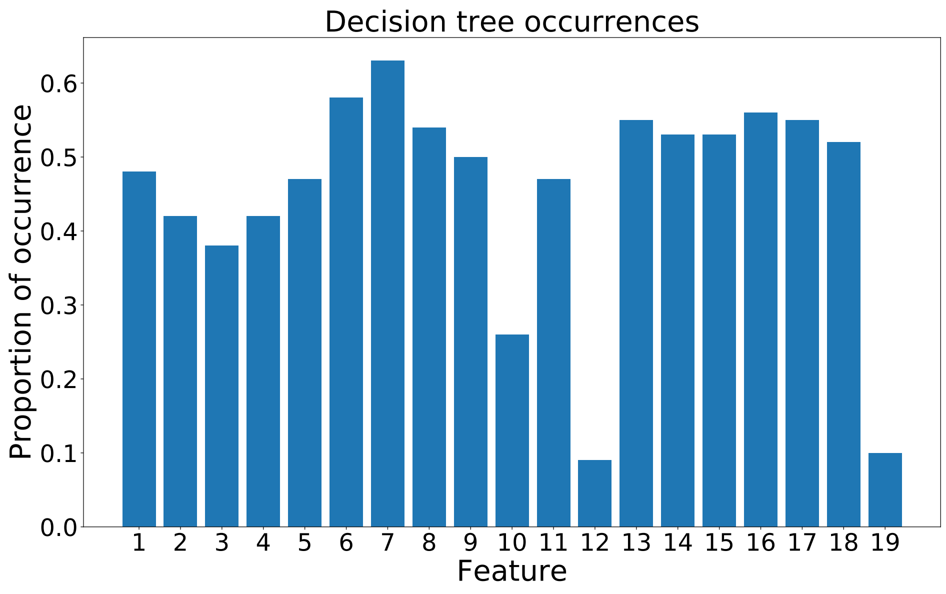


Figure 5.4: The proportion of occurrence in the best chromosome for each feature in the data set obtained by the GA with a decision tree classifier for the elevator data.

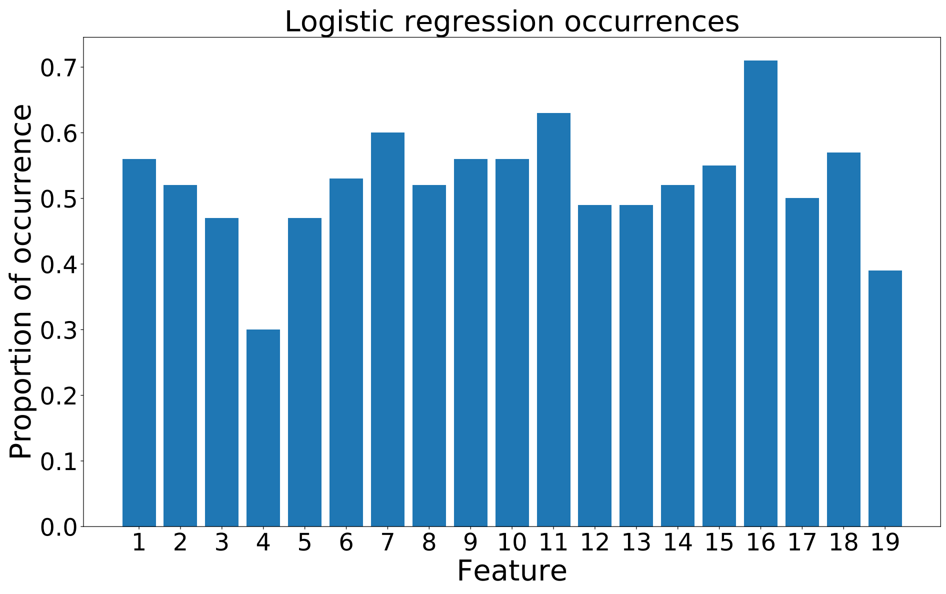


Figure 5.5: The proportion of occurrence in the best chromosome for each feature in the data set obtained by the GA with a logistic regression classifier for the elevator data.

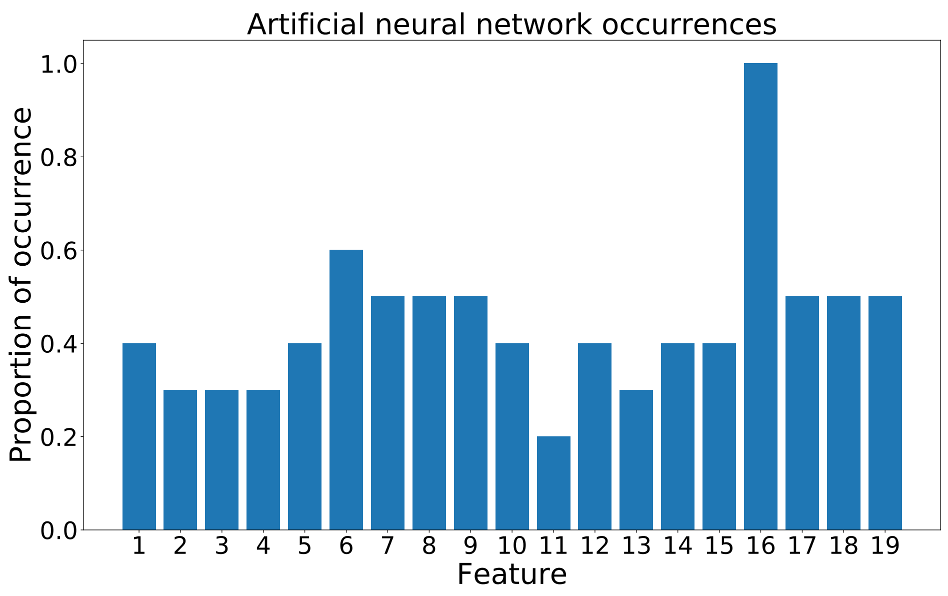


Figure 5.6: The proportion of occurrence in the best chromosome for each feature in the data set obtained by the GA with a multilayer perceptron neural network classifier for the elevator data.

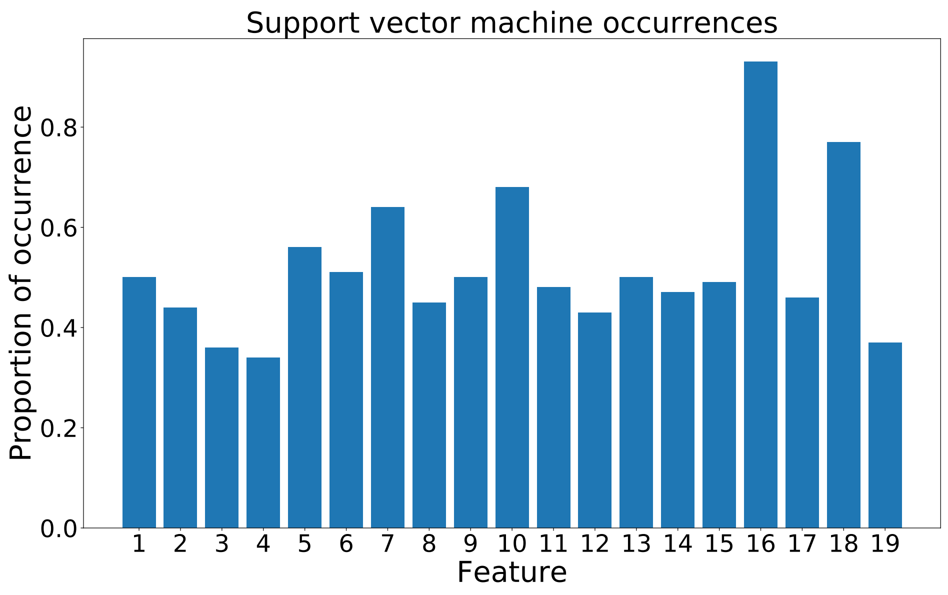


Figure 5.7: The proportion of occurrence in the best chromosome for each feature in the data set obtained by the GA with an SVM classifier for the elevator data.

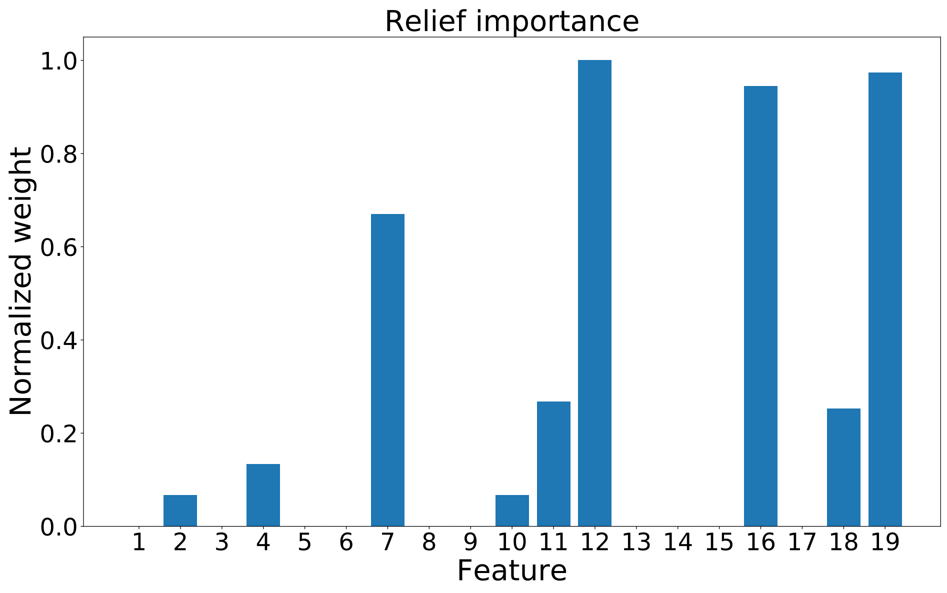


Figure 5.8: Feature importances obtained by the Relief algorithm for the elevator data.

5.4 Embedded results

Figure 5.9 illustrates the Group LASSO coefficients obtained. The values for the categorical features are calculated as the average of each category value, as was described in the Method section. The results are similar to the Relief algorithm in the sense that the numerical features are all of importance. However, the LASSO also includes some additional features that was not selected by Relief. Similar to Relief, the Group LASSO manages to filter out the constant features 1, 8 and 9. The results from the LASSO are also similar to the results obtained by the random forest, which are shown in Figure 5.10. The LASSO and the random forest consider the same subset of features to be important, but there are some differences in the relative importances of these features.

5.5 Evaluation of subset prediction performance

As seen by the bar charts for the GAs, it is not obvious which features to include in the subset selection by simply inspecting the charts. To make the subset selection less subjective, features are included one by one based on the relative importance in descending order. Whenever a feature is added, the corresponding classifier is trained and a test score is computed. The results from doing this with the GA rankings are shown in Figure 5.11. It is noted that the Naive Bayes classifier obtains a significantly better result with 6-10 features than when the full data set is used. For the other classifiers, the test scores increase to a roughly constant level already at three features, which opens up for a significant reduction in the number

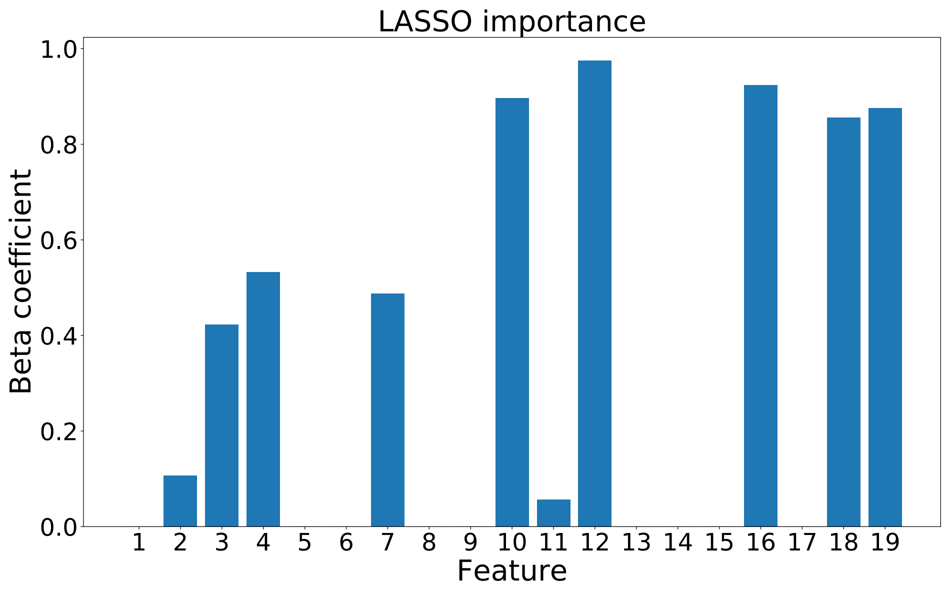


Figure 5.9: Feature importances obtained by the Group LASSO algorithm for the elevator data.

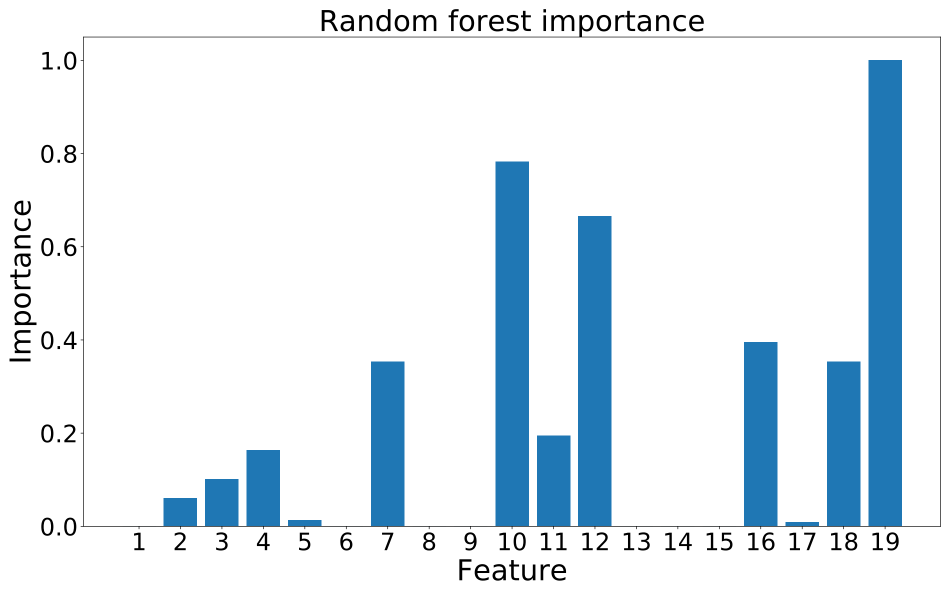


Figure 5.10: Feature importances obtained by the random forest algorithm for the elevator data.

of features needed.

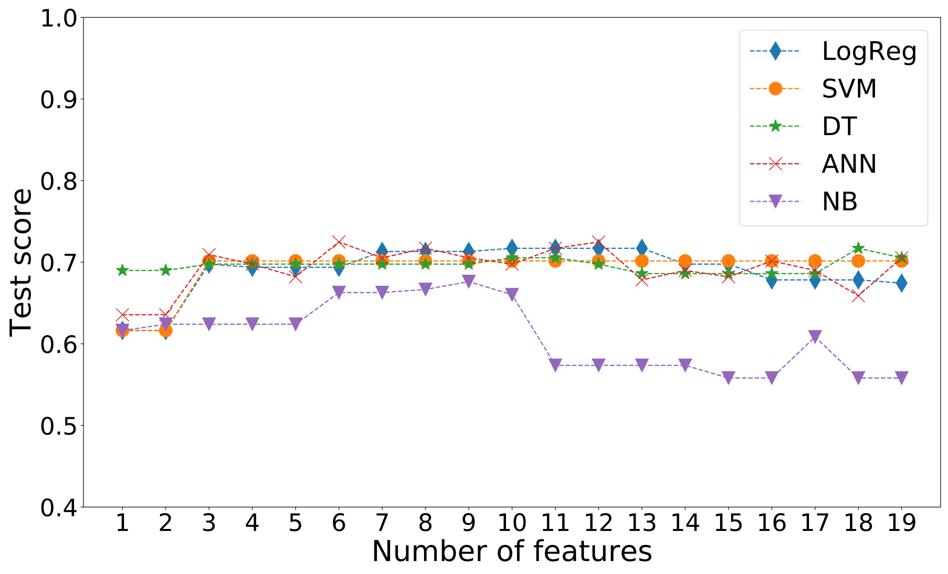


Figure 5.11: The evolution of test accuracy as features from the GA are included in descending relative importance order for the elevator data. Naive Bayes (purple) obtains significantly higher scores for 6-10 features. The decision tree (green) is approximately constant, with just a small difference between including one feature and all features. The other classifiers show a significant increase at 3 features.

Table 5.1 shows the classification score for the elevator data by using the best subset selection from each algorithm and the full data. The selections were obtained from Figure 5.1 for the SFS, Figure 5.11 for the GAs and by inspection of the bar charts for the other algorithms. The subsets have been tested with each of the five classifiers presented in Chapter 3. Each numerical value in the table shows the mean value over 10 runs together with a 95% confidence interval. In each row in the table, the same split into training and test sets is used to calculate both the reduced score and the full score. However, a new split is generated at a new row. Due to different splits in training and test data, the scores show a slight variation, as indicated in the full data column.

The subset selections obtained by the GAs and SFS appear to outperform classifiers trained on the full data set, however it is only statistically significant for the Naive Bayes classifier. The LASSO and random forest subsets for the Naive Bayes classifier also show a statistically significant improvement compared to the full data set. Notably, the subset obtained from the Relief algorithm for the Naive Bayes classifier performs significantly worse than the full data set. For the other classifiers, the results differ with some performing better and some performing worse than the full data set. However, none of these differences are statistically significant.

Algorithm	**Classifier**	**Reduced data score**	**Full data score**
GA	NB	0.650 ± 0.0240	0.524 ± 0.0253
GA	DT	0.699 ± 0.0186	0.688 ± 0.0229
GA	ANN	0.681 ± 0.0152	0.678 ± 0.0168
GA	SVM	0.718 ± 0.0156	0.705 ± 0.0141
GA	LogReg	0.693 ± 0.0153	0.688 ± 0.0157
SFS	NB	0.594 ± 0.0163	0.513 ± 0.0143
SFS	DT	0.699 ± 0.0126	0.690 ± 0.0128
SFS	ANN	0.687 ± 0.0178	0.678 ± 0.0202
SFS	SVM	0.698 ± 0.0231	0.686 ± 0.0249
SFS	LogReg	0.694 ± 0.0131	0.685 ± 0.0129
Relief	NB	0.456 ± 0.0239	0.516 ± 0.0323
Relief	DT	0.667 ± 0.0145	0.673 ± 0.0169
Relief	ANN	0.679 ± 0.0123	0.665 ± 0.0132
Relief	SVM	0.684 ± 0.0157	0.702 ± 0.0119
Relief	LogReg	0.702 ± 0.0155	0.704 ± 0.0131
LASSO	NB	0.559 ± 0.0336	0.504 ± 0.0187
LASSO	DT	0.686 ± 0.0179	0.685 ± 0.0213
LASSO	ANN	0.682 ± 0.0193	0.679 ± 0.0161
LASSO	SVM	0.698 ± 0.0204	0.698 ± 0.0205
LASSO	LogReg	0.698 ± 0.0155	0.693 ± 0.0200
RF	NB	0.562 ± 0.0304	0.505 ± 0.0199
RF	DT	0.693 ± 0.0243	0.693 ± 0.0254
RF	ANN	0.671 ± 0.0174	0.677 ± 0.0121
RF	SVM	0.709 ± 0.0159	0.697 ± 0.0167
RF	LogReg	0.700 ± 0.0184	0.689 ± 0.0166

Table 5.1: Test score for the reduced data sets compared to the full data set for the elevator data.

5.6 Second group of objects

The algorithms were run on a partition of the data set that contains inspection data on gates, which is the second largest group in the data after preprocessing. The test score plotted against the number of features for the SFS is shown in Figure 5.12 and for the GA in Figure 5.13. More detailed results about each feature selection algorithm are found in Appendix A. It is obvious from the plots that the classification accuracy is much higher for this group of objects compared to the previous. It is also remarkable that the classification performance for all classifiers is almost maximal when the most important feature is used solely. Similar to the largest group, the Naive Bayes classifier deteriorates when too many features are included and the worsening is more severe for this partition of the data. However, the range of features that performs well is larger for this partition of the data compared to the elevator group.

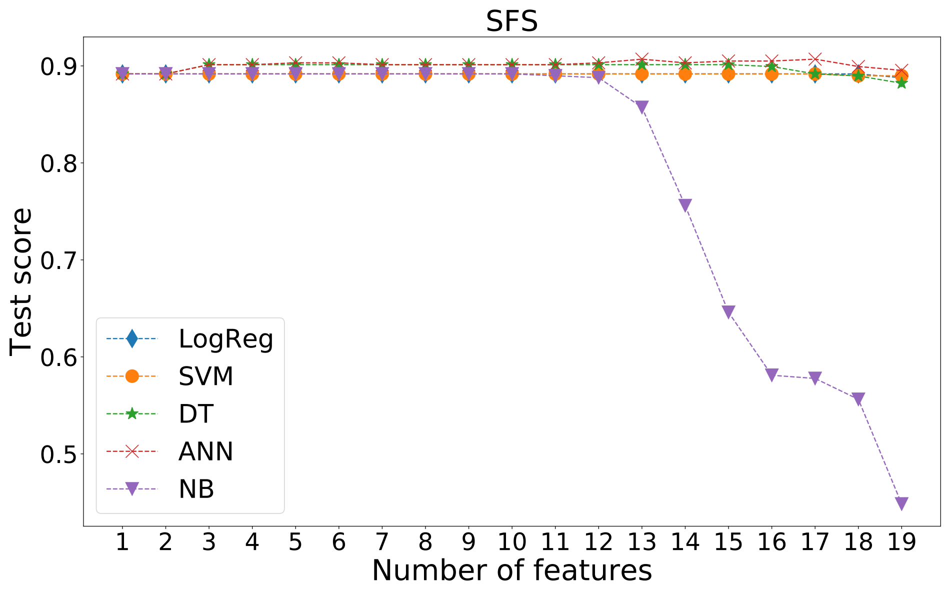


Figure 5.12: The evolution of test accuracy as a function of the number of features included by the SFS for the gates data. All classifiers achieve good performance using only the most importance feature. The Naive Bayes classifier (purple) performance is reduced when the number of features approach the full data set, with a cutting point at 12 features.

The subsets were trained and tested in an analogous fashion to the first group and these results are presented in Table 5.2. Similar to what was observed in the first group, the GA and SFS selections for the Naive Bayes classifier significantly improves the test score compared to the full data set. This is not achieved by the other algorithms. For the other classifiers, the differences are smaller. The confidence intervals for the Naive Bayes classifiers trained on the full data set are wide, which indicates that there is a large dependence on how the data is split into training and test sets.

Algorithm	**Classifier**	**Reduced data score**	**Full data score**
GA	NB	0.889 ± 0.0103	0.397 ± 0.0597
GA	DT	0.903 ± 0.0106	0.896 ± 0.00701
GA	ANN	0.899 ± 0.0192	0.893 ± 0.0193
GA	SVM	0.893 ± 0.0152	0.891 ± 0.0162
GA	LogReg	0.901 ± 0.0139	0.901 ± 0.0144
SFS	NB	0.882 ± 0.0163	0.349 ± 0.0872
SFS	DT	0.898 ± 0.0146	0.882 ± 0.0134
SFS	ANN	0.896 ± 0.0187	0.891 ± 0.0170
SFS	SVM	0.888 ± 0.0158	0.883 ± 0.0107
SFS	LogReg	0.893 ± 0.0151	0.893 ± 0.0151
Relief	NB	0.328 ± 0.0537	0.423 ± 0.105
Relief	DT	0.881 ± 0.0184	0.879 ± 0.0161
Relief	ANN	0.891 ± 0.0137	0.899 ± 0.0176
Relief	SVM	0.888 ± 0.0132	0.886 ± 0.0133
Relief	LogReg	0.893 ± 0.0184	0.892 ± 0.0171
LASSO	NB	0.390 ± 0.0659	0.486 ± 0.0508
LASSO	DT	0.872 ± 0.0118	0.874 ± 0.0149
LASSO	ANN	0.898 ± 0.00944	0.900 ± 0.00789
LASSO	SVM	0.880 ± 0.00754	0.879 ± 0.00761
LASSO	LogReg	0.901 ± 0.0124	0.901 ± 0.0126
RF	NB	0.447 ± 0.139	0.398 ± 0.0879
RF	DT	0.876 ± 0.0173	0.874 ± 0.0179
RF	ANN	0.899 ± 0.0112	0.898 ± 0.0127
RF	SVM	0.883 ± 0.0119	0.879 ± 0.0125
RF	LogReg	0.889 ± 0.0123	0.889 ± 0.0112

Table 5.2: Test score for the reduced data sets compared to the full data set for the gates data.

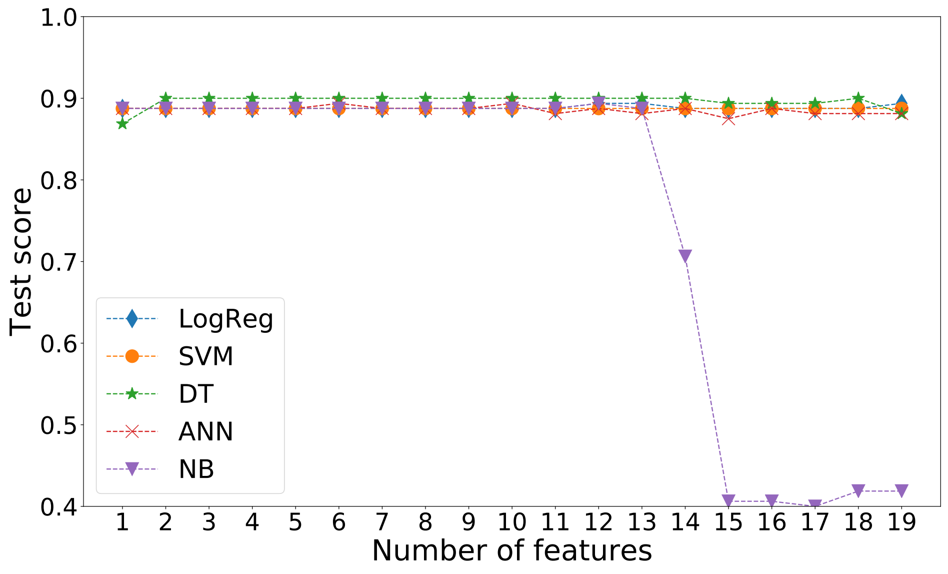


Figure 5.13: The evolution of test accuracy as a function of the number of features included by the GA for the gates data. All classifiers achieve good performance using only the most importance feature. The Naive Bayes classifier (purple) performance is reduced when the number of features approach the full data set, with a cutting point at 13 features.

6
Discussion

The following sections will analyze the differences in the results from the feature selection algorithms and in which situations each algorithm is preferred.

6.1 Differences in subset selections

As was argued in Section 2.2, filter methods only use general information about the data, such as correlations and distances, and are not impacted by any classification algorithm. A comparison between the subset selections obtained by Relief, LASSO and the random forest in Figures 5.8, 5.9 and 5.10 respectively reveals that the selection obtained by the Relief algorithm is a subset of the selections obtained by the other algorithms. This shows that the random forest and the LASSO algorithms capture the same general information about the data, but may also include additional features. It was further observed that the LASSO and the random forest included the same features in their subsets, however with different relative importances between the features. The GA selections, on the other hand, are different from the other selections in that they sometimes ignore features that are relevant in general. An example is found in the GA with decision tree as shown Figure 5.4 where feature 12 and 19 were excluded, but they are included in the Relief selection.

The differences in the results observed between different feature selection algorithms argue that one should not draw conclusions about important features from just one algorithm. Which selection that is most suitable depends on the goal of the feature selection. In terms of general interpretability, the Relief algorithm is likely to provide the best information since this algorithm does not make use of any classifier. Also, the embedded methods showed similar results when applied to the data set used in this thesis and are also likely to provide useful general information about the data. If the goal is to enhance the prediction performance of a classifier, GAs are a suitable choice as seen from the test data prediction results in Tables 5.1 and 5.2. The feature subsets that were constructed by the GA are however not likely to provide general information about the most important features, due to the fact that features found by the Relief are missing for some of the GA results. However, for classifiers that make explicit assumptions on the data, such as independence between features which is assumed by the Naive Bayes, the selection can be interpreted as the subset that best fulfills this criterion, which is a general statement about the data.

A drawback with the GAs used in the thesis is that they tend to include constant features, which was observed with all classifiers. This result is in contrast to the results obtained by Relief, LASSO and random forest who all manage to remove these features. The reason for GAs' tendency to include these features in the subset selection might be due to the fact that they do not provide any bad information since they are free from noise. The fitness function of the GA that was used does not provide any penalty regarding overly complex models, which is done in for example the LASSO. Thus, the GA selection solely relies on how the classifier test score is impacted by the features, and this score is not reduced by noise-free, constant features. One could remove features with low variance manually, as a preprocessing step, but as was observed, several feature selection algorithms manage to filter out such features and it is desirable to incorporate as much of the selection as possible in one algorithm. Possible improvements to the GA are discussed below in Section 6.6.

6.2 Differences in prediction performance

The classification performance on test data for the different subset selections in Tables 5.1 and 5.2 shows that the subset selections obtained by the GAs and SFS appear to outperform the full data set. The results are most significant for the Naive Bayes classifier. A comparison of the results from the different feature selection algorithms where the Naive Bayes classifier is used shows that the GA and SFS selections performs better than the subsets obtained from Relief, LASSO and the random forest, which verifies the hypothesis that wrapper methods lead to an enhanced prediction performance. The large differences for this classifier are likely due to the crude assumption about independence between features. The features in the full data set and in the subsets obtained by the other feature selection algorithms are likely to violate this assumption more than the GA and SFS subsets and thus obtain a lower prediction score. Since the Naive Bayes classifier makes the strongest assumptions about the data among the tested classifiers, the advantages of a tuned selection is greatest for this classifier.

Although the Naive Bayes classifier usually performs worse than other classifiers, it is still of importance in certain areas of applications. One such area is text classification which usually deals with a large number of features [38–41]. In this application, it is advantageous with a fast classifier that scales well with increasing number of features, which is offered by the Naive Bayes classifier. As the results in this thesis indicate, the feature subset selections obtained using a wrapper method such as a GA are necessary in order to reduce the violation of the independence assumption.

Finally, it is noted from the wide confidence intervals in Table 5.1 and especially in Table 5.2 that the results for the Naive Bayes classifier fluctuates more, when different splits into training and test data sets are used. Since this classifier deals explicitly with probabilities, even small changes in prevalences of a class in a set can have significant impact on the results. The reason why the second group has larger confidence intervals is likely due to that it contains fewer observations, which makes differences in class prevalence more significant. In order to narrow the intervals, more data points would be required which would make the prevalences more robust.

6.3 Computational time

The computational time required for the wrapper methods is much larger than for the filter and embedded methods due to the frequent invocations of the classifier. SFS requires $p(p+1)/2$ classifiers to be trained, where p is the number of features in the data set. The GAs will train $N \cdot n_g \cdot n_r$ classifiers, where N is the population size, n_g is the number of generations and n_r is the number of runs. These hyperparameters for the GA are likely to be increased slightly if a larger data set is used, however they should not differ too much. Assuming that these parameters always take the values in Table 4.2, the GA will require more classifiers than SFS up until roughly 1400 features, so for most data sets, the GA is more computationally expensive.

The choice of classifier greatly impacts the time, with neural networks, logistic regression, decision trees and support vector machines requiring significantly more time than Naive Bayes classifiers. For large data sets, this can make the wrapper approach unfeasible, at least with the best performing classifiers. As was argued above, it is important to not discard worse performing classifiers such as the Naive Bayes, since they might be more feasible in applications with extremely large data sets. For the most time consuming classifiers, the prediction results do not differ much between the different feature selection algorithms. Since the filter and embedded methods are more computationally efficient, they are likely to be a better choice in these situations.

6.4 Differences between inspection groups

The two largest groups of inspections were investigated in the results section. As shown by the GA results in Figure 5.11 and Figure 5.13, the classification accuracy is much higher for the gates group. A remarkable note for the gates group is that one achieves almost the same classification accuracy with just one feature as the maximal accuracy. For the elevator group, there is a significant increase when three features are included. This means that the dimensionality of both groups of data can be reduced greatly and still achieve reasonable classification accuracies.

For the Naive Bayes classifier, there is a range of local optima at 6-10 features for the elevator group as found by the GA. In the gates group, the Naive Bayes performs at a roughly constant level with 1-13 features included, after which the performance is significantly reduced. There are two probable reasons for the difference in range. First, the statistical dependence relationship between the features can be different for this partition of the data, which means that a larger subset of features might fulfill the independence assumption. Second, it is possible that the Naive Bayes classifier is less sensitive to violations of the assumptions for data that is easier to classify. It is, however, more important to find a subset of features in the well performing region for the gates group, since the decrease in performance is more severe. While this is easier to do for the second group since the range is greater, it is still not achieved by the subsets obtained from the Relief, LASSO and random forest algorithms, as indicated by the test scores in Table 5.2. Even for data sets that are simpler to classify, a wrapper method is required to capture a suitable feature subset to be used with the Naive Bayes classifier.

6.5 The importance of individual features

Some features appear in many selections, but the reason for a feature to be considered important varies. In this section, the features included by the Relief algorithm for the elevator data are discussed since they are believed to be of general importance for describing the data. They are summarized in Table 6.1.

One feature whose importance at first glance seems surprising is feature 19 (DaysSinceActivityCompleted), which would mean that the date of the inspection is important for describing the result. The importance is, however, an artefact of the way that the company works. By inspecting the raw data, one can note that it is not uncommon for multiple objects at the same site to achieve the same result. By convenience, an employee will inspect several objects at a site at the same date. For this reason, many objects may obtain the same label at the same date, which shows as importance in this feature. There is, of course, nothing fundamental with the inspection date that can impact the results. If the employee would inspect a single object at each site everyday and instead move between different sites, it is likely that this feature would not be considered important anymore, since the artificial pattern would no longer exist. Features 7 (InspectionObjectTypeId), 11 (Manufacturer) and 18 (DaysSinceEmployeeStartDate) are also related to specific sites and thus important for the same reason as described above. Most object at a site are of the same type, and often have the same manufacturer. For geographical reasons, an employee will visit a subset of all sites in the country. A natural question to ask is why feature 4 (InspectionObjectSiteId) is low ranked in the importance graphs, given that all of the above features are important because of this feature. The site ID feature is one of the categorical features with the largest number of categories and was such affected by heavy preprocessing steps that had to be taken in order for these features to be viable to work with. These steps removed much information and thus probably reduced the importance of that feature. This illustrates one of the difficulties with categorical feature selection.

On the contrary, the importances of feature 12 (YearOfManufacture) and feature 16 (ActivityResultTimeSpent) are explained by more fundamental reasons. The age of an object is important because older objects are more likely to be disapproved compared to newer objects and the time required for the inspection gives information about the result. The most obvious example for the latter is objects that have inspection time 0 reported, which are probable to obtain the result “Not completed”. These features are fundamentally important in the sense that their importance can not be removed by randomizing another feature, as is the case with the site ID feature for the artefact features.

Index	**Name**	**Type**
7	InspectionObjectTypeId	Artefact
11	Manufacturer	Artefact
12	Year of manufacture	Fundamental
16	ActivityResultTimeSpent	Fundamental
18	DaysSinceEmployeeStartDate	Artefact
19	DaysSinceActivityResultCompleted	Artefact

Table 6.1: Importance summarization for the features selected by the Relief algorithm.

6.6 Future work

The GA developed in this thesis uses a quite simple measure for evaluating the fitness values, as it only takes into account the test scores obtained by the proposed selection. As was observed, the subset selections typically include constant features. It would be of interest to incorporate a penalty for overly complex models in the fitness function of the GA, in order to combat the selection of constant features. The objective of this function would be to give a higher score to a selection that includes all relevant features but excludes constants, compared to one that includes all relevant features and the constants. In the GA in this thesis, these two cases would result in the same fitness value.

Although the computational time required for GAs is large, the formulation opens up for effective parallelization which could make them more scalable. A programming model known as MapReduce [42], which aims at facilitating the treatment of large data sets through parallelization, has been studied in the context of GAs [43–45]. Since computational speed is one of the main concerns with these types of algorithms, it would be beneficial to further study ways of making the implementation more efficient to increase their applicability to larger data sets.

7 Conclusions

GAs provide an elegant way of selecting features in the wrapper setting, even when the features are of categorical nature, since all treatment except for the computation of the fitness values are done with the chromosomes. It was demonstrated that it is possible to define a meaningful importance measure, to gain knowledge about how important each feature is for the GA for describing patterns in the data, which is not possible to achieve in stepwise selection methods.

GAs provide the advantages of wrapper methods in that the feature selection is tuned to a classifier, which generally leads to better performance than the feature selection obtained by filter and embedded methods. It was shown that the benefits of a tuned subset selection are greatest when a Naive Bayes classifier is used, where the filter and embedded methods produced worse performing feature subsets compared to the GA and the stepwise forward selection algorithm. For other classifiers, the differences were smaller. It was further found that one can reduce the dimensionality of the data significantly while still maintaining good results. For the elevator group, the data showed potential to be reduced down to 3 features, while the gates group showed good results with only the most important feature included in the data set.

However, it is a costly procedure to run GAs, as it is required to train a large number of classifiers. The time required is strongly dependent on what classifier is used. For the best performing classifiers, the time required is large which makes them unfeasible for larger data sets. More simple classifiers such as the Naive Bayes are still viable for larger data sets, and as was observed, can benefit greatly from feature selection via a GA. Since the differences between the feature selection algorithms were small for the other classifiers, it is advisable to choose a faster algorithm such as Relief, LASSO or random forest for those cases, if the data set is large.

Bibliography

[1] M. Chen, S. Mao and Y. Liu, “Big Data: A Survey”, *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171-209, April, 2014.

[2] X. Jin et al., “Significance and Challenges of Big Data Research”, *Big Data Research*, vol. 2, no. 2, pp. 59-64, February, 2015.

[3] P. Chen and C-Y Zhang, “Data-intensive applications, challenges, techniques and technologies: A survey on Big Data”, *Information Sciences*, vol. 275, pp. 314-347, August, 2014.

[4] E. Al Nuaimi et al., “Applications of big data to smart cities”, *Journal of Internet Services and Applications*, vol. 6, no. 25, August, 2015.

[5] H. M. Krumholz, “Big Data And New Knowledge In Medicine: The Thinking, Training, And Tools Needed For A Learning Health System”, *Health Affairs*, vol. 33, no. 7, pp. 1163-1170, July, 2014.

[6] J. Yan et al., “Effective and Efficient Dimensionality Reduction for Large-Scale and Streaming Data Preprocessing”, *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 320-333, March, 2006.

[7] M. Verleysen and D. François, “The Curse of Dimensionality in Data Mining and Time Series Prediction”, *Lecture Notes in Computer Science*, vol. 3512, pp. 758-770, March, 2005.

[8] L. Zhao, G. Zhuang and X. Xu, “Facial Expression Recognition Based on PCA and NMF”, In Proc. Proceedings of the 7th World Congress on Intelligent Control and Automation, 2007.

[9] M. Dash and H. Liu, “Feature Selection for Classification”, *Intelligent Data Analysis*, vol. 1, no. 1-4, pp. 131-156, March, 1997.

[10] G. Chandrashekar and F. Sahin, “A survey on feature selection methods”, *Computers and Electrical Engineering*, vol. 40, no. 1, pp. 16-28, January, 2014.

[11] Y. Saeys et al., “A review of feature selection techniques in bioinformatics”, *Bioinformatics*, vol. 23, no. 19, pp. 2507–2517, October, 2007.

[12] A. Blum and P. Langley, “Selection of relevant features and examples in machine learning”, *Artificial Intelligence*, vol 97, no. 1-2, pp. 245-271, December, 1997.

[13] S. Das, “Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection”, In Proc. Proceedings of the Eighteenth International Conference on Machine Learning, 2001, pp. 74-81.

[14] Z. Zhu, Y.-S. Ong and M. Dash, “Wrapper–Filter Feature Selection Algorithm Using a Memetic Framework”, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 70-76, January, 2007.

[15] A. W. Whitney, “A direct method of nonparametric measurement selection”, *IEEE Transactions on Computers*, vol. 20, no. 9, pp. 1100-1103, September, 1971.

[16] J. Reunanen, “Overfitting in making comparisons between variable selection methods”, *The Journal of Machine Learning Research*, vol. 3, pp. 1371-1382, March, 2003.

[17] P. Pudil, J. Novovičová and J. Kittler, “Floating search methods in feature selection”, *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119-1125, November, 1994.

[18] E.W. Steyerberg, M.J.C. Eijkemans and J.Dik F. Habbema, “Stepwise Selection in Small Data Sets: A Simulation Study of Bias in Logistic Regression Analysis”, *Journal of Clinical Epidemiology*, vol. 52, no. 10, pp. 935-942, October, 1999.

[19] M. Whittingham et al., “Why do we still use stepwise modelling in ecology and behaviour?”, *Journal of Animal Ecology*, vol. 75, no. 5, pp. 1182-1189, September, 2006.

[20] S. Derksen and H. J. Keselman, “Backward, forward and stepwise automated subset algorithms: Frequency of obtaining authentic and noise variables”, *British Journal of Mathematical and Statistical Psychology*, vol. 42, pp. 265-282, November, 1992.

[21] R. Mundry and C.L. Nunn, “Stepwise model fitting and statistical inference: turning noise into signal pollution”, *The American Naturalist*, vol. 173, no. 1, pp. 119-123, January, 2009.

[22] J. Yang and V. Honavar, “Feature Subset Selection Using A Genetic Algorithm”, *Feature extraction, construction and selection*, vol. 13, no.2, pp. 44-49, April 1998.

[23] W. Siedlecki and J. Sklansky, “A note on genetic algorithms for large-scale feature selection”, *Pattern Recognition Letters*, vol. 10, no.5, pp. 335-347, November, 1989.

[24] H. Uguz, “A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm”, *Knowledge-Based Systems*, vol. 24, no. 7, pp. 1024-1032, October, 2011.

[25] M. Wahde, “Biologically Inspired Optimization Methods”, Gothenburg, Sweden: WIT Press, 2008.

[26] J. Benesty et al., *Noise Reduction in Speech Processing*, Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2009.

[27] H. Theil, “On the Estimation of Relationships Involving Qualitative Variables”, *American Journal of Sociology*, vol. 76, no. 1, pp. 103-154, July, 1970.

[28] E. Plischke, “An adaptive correlation ratio method using the cumulative sum of the reordered output”, *Reliability Engineering & System Safety*, vol. 107, pp. 149-156, November, 2012.

[29] K. Kira and L. A. Rendell, “A Practical Approach to Feature Selection”, In Proc. Proceeding ML92 Proceedings of the ninth international workshop on Machine learning, 1992, pp. 249-256.

[30] I. Kononenko et al., “Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF”, *Applied Intelligence*, vol. 7, no. 1, pp. 39-55, January, 1997.

[31] R. Tibshirani, “Regression Shrinkage and Selection via the Lasso”, *Journal of the Royal Statistical Society*, vol. 58, no. 1, pp. 267-288, 1996.

[32] M. Yuan and Y. Lin, “Model selection and estimation in regression with grouped variables”, *Journal of the Royal Statistical Society Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49-67, Febraury, 2006.

[33] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Stanford, California, 2008.

[34] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press: 2016.

[35] C. Cortes and V. Vapnik, “Support-Vector Networks”, *Machine Learning*, vol. 20, no.3, pp. 273-297, September, 1995.

[36] I. Rish, “An Empirical Study of the Naïve Bayes Classifier”, In Proc. IJCAI 2001 workshop on empirical methods in artificial intelligence, 2001, pp. 41-46.

[37] F. Pedregosa et al., “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, November, 2011.

[38] G. Feng et al., “Feature subset selection using naive Bayes for text classification”, *Pattern Recognition Letters*, vol. 65, no. 1, pp. 109-115, November, 2015.

[39] D. M. Diab and K. M. El Hindi, “Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification”, *Applied Soft Computing*, vol. 54, pp.183-199, May, 2017.

[40] L. Jiang et al., “Deep feature weighting for naive Bayes and its application to text classification”, *Engineering Applications of Artificial Intelligence*, vol. 52, pp. 26-39, June, 2016.

[41] S. Xu, “Bayesian Naïve Bayes classifiers to text classification”, *Journal of Information Science*, vol. 44, no. 1, pp. 48-59, February, 2018.

[42] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, In Proc. Sixth Symposium on Operating System Design and Implementation, 2004, pp. 137-150.

[43] A. Verma et al., “Scaling Genetic Algorithms Using MapReduce”, In Proc. 2009 Ninth International Conference on Intelligent Systems Design and Applications, 2009, pp. 13-18.

[44] E-S. M. El-Alfy and M. A. Alshammari, “Towards scalable rough set based attribute subset selection for intrusion detection using parallel genetic algorithm in MapReduce”, *Simulation Modelling Practice and Theory*, vol. 64, pp. 18-29, May, 2016.

[45] F. Ferrucci et al., “A parallel genetic algorithms framework based on Hadoop MapReduce”, In Proc. Proceedings of the 30th Annual ACM Symposium on Applied Computing, 2015, pp. 1664-1667.

A

More details on the gates group

Bar charts of the importances for the gates group are presented in Figures A.4-A.8. It is notable that the Naive Bayes classifier in Figure A.1 includes more features for the gates group compared to the elevator group, which hints that there is a larger set of independent features for this partition of the data. It is also notable that the features that are of general importance and obtained by the Relief algorithm overlap the features in the elevator group, as illustrated in Figure A.6. The subset selections obtained from the Group LASSO and the random forest contain the same features for this group, which was also the case in the elevator group.

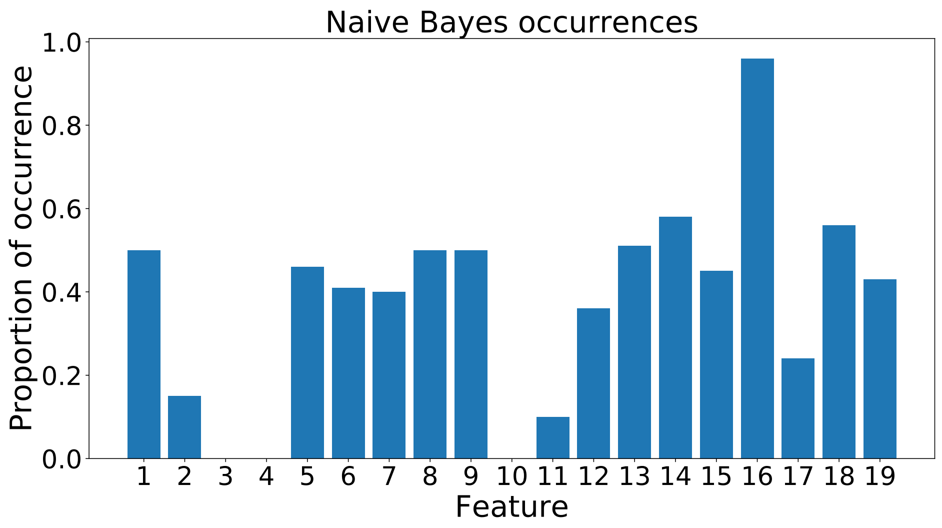


Figure A.1: The proportion of occurrence in the best chromosome for each feature for the gates group obtained by the GA with a Naive Bayes classifier.

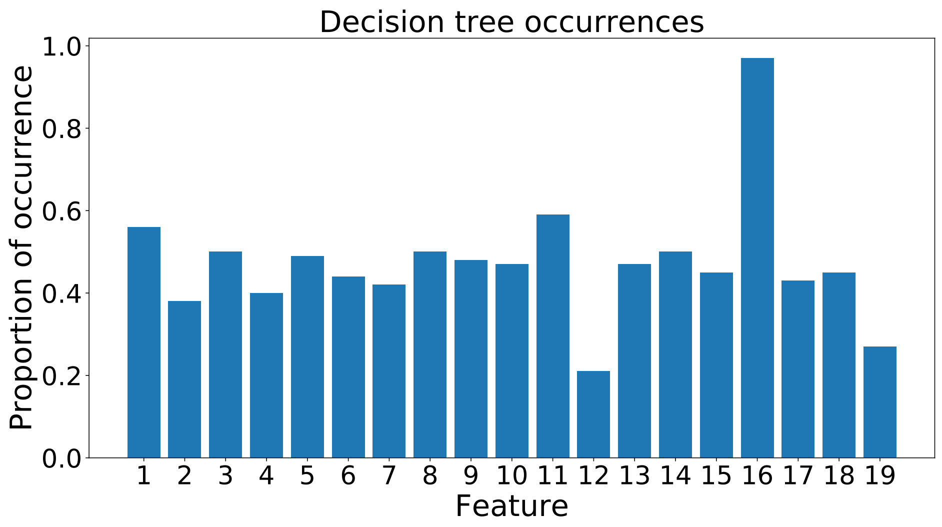


Figure A.2: The proportion of occurrence in the best chromosome for each feature for the gates group obtained by the GA with a decision tree classifier.

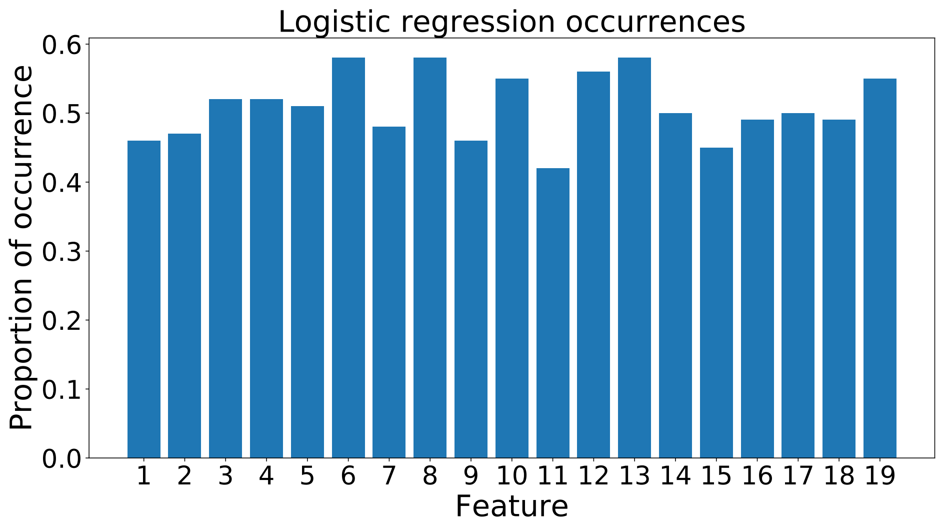


Figure A.3: The proportion of occurrence in the best chromosome for each feature for the gates group obtained by the GA with a logistic regression classifier.

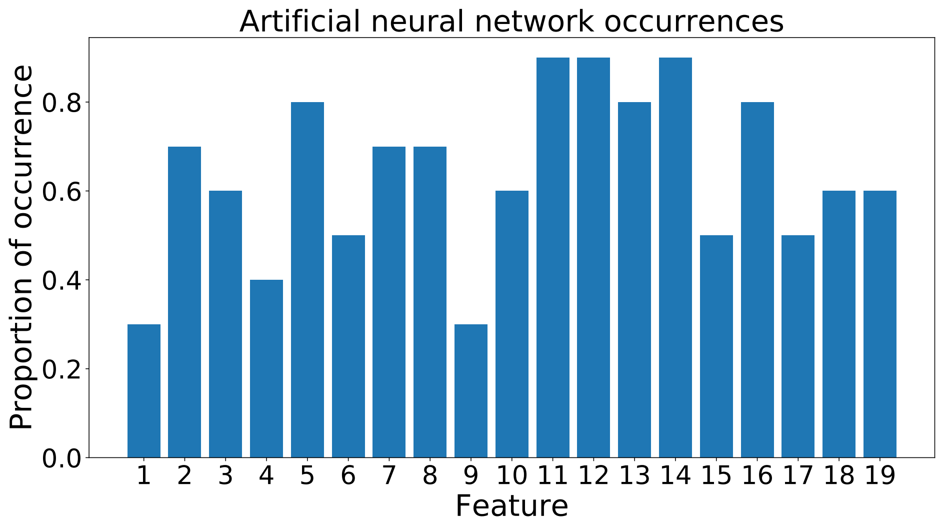


Figure A.4: The proportion of occurrence in the best chromosome for each feature for the gates group obtained by the GA with a multilayer perceptron neural network classifier.

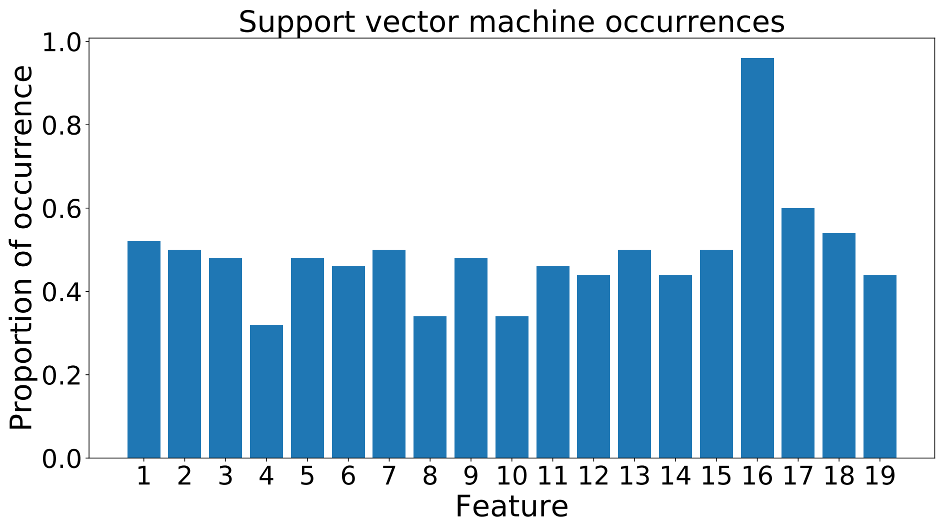


Figure A.5: The proportion of occurrence in the best chromosome for each feature for the gates group obtained by the GA with an SVM classifier.

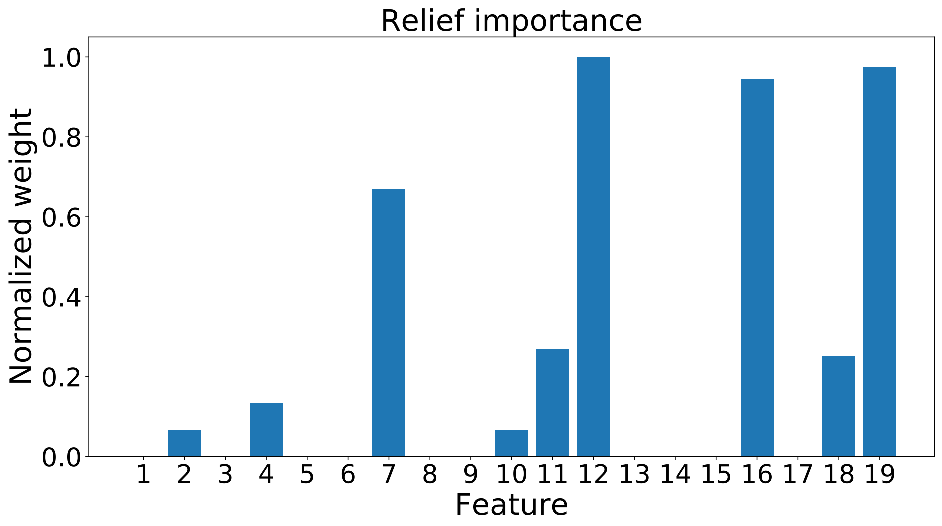


Figure A.6: Feature importances for the gates group obtained by the Relief algorithm.

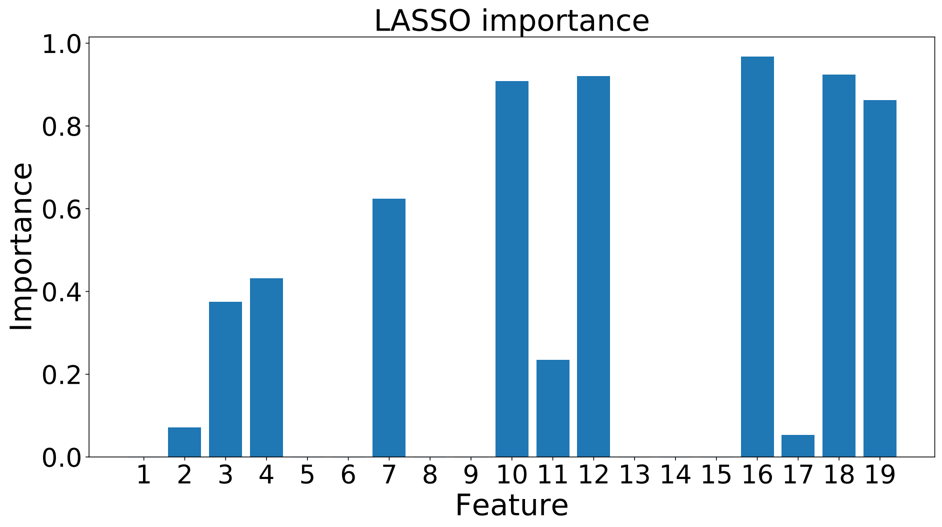


Figure A.7: Feature importances for the gates group obtained by the Group LASSO.

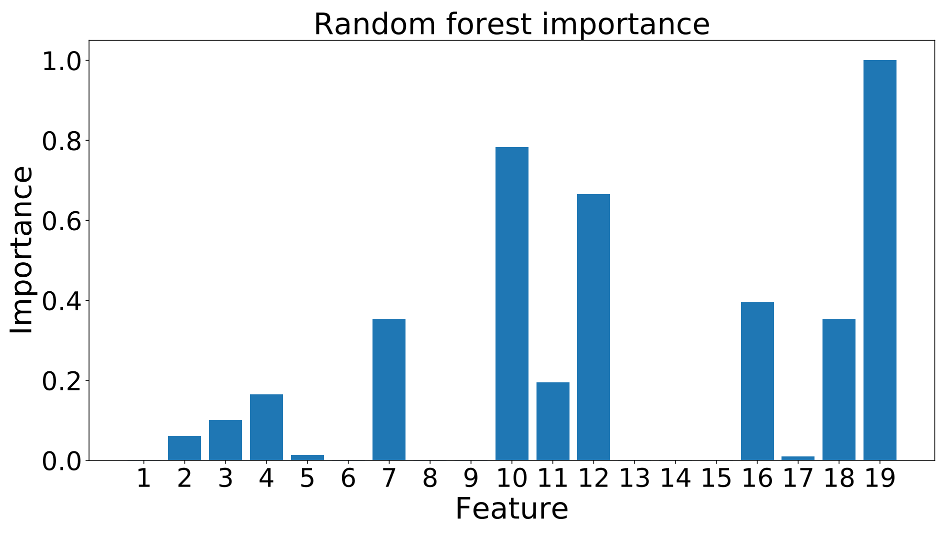


Figure A.8: Feature importances for the gates group obtained by the random forest.