Universiteit Leiden
Opleiding Informatica

Mythbusting data mining urban legends

through large scale experimentation

Name:	Martijn J. Post
Date:	25/8/2016
1st supervisor:	Peter van der Putten
2nd supervisor:	Jan N. van Rijn

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Mythbusting data mining urban legends through large scale experimentation

Martijn J. Post

Abstract

Data mining researchers and practitioners often use general rules of thumb or common data mining wisdom, those are so called data-mining myths. These myths are not always proven or sufficiently proven for general non-specific cases. Therefore, we will closely examine two data-mining myths in this thesis, we will give a small introduction about the myth and then try to prove or bust the myth using a large number of different data sets. The data used is from OpenML, a large user-shared data repository. The two myths that will be investigated are "Data preparation is more important than algorithm selection" and "Non-linear models perform better than linear models". The myths are investigated by setting up two types of experiments, an explorative experiment that uses all the data sets for insights about the myth, and a meta-learning experiment that uses the results of the explorative experiment to obtain further insights and learn more about specific meta-features. We show that while feature selection can improve or diminish the results of certain algorithms, feature selection is only a small part of data preparation and therefore we can not comment on the importance of data preparation in the myth using only feature selection and thus a more expansive experiment is needed. Also, we show that while non-linear models might be better for some data sets, this is not always true and linear models can be a lot better than non-linear models.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Data miners commonly use certain rules of thumb that may sound logical but have never truly been proven or only partially for specific cases. Our goal is to extensively research a number of these rules of thumb or "data-mining myths" using a large amount of diverse data sets. The large amount of diverse data sets allow our results to be representative for a range of real world problem domains. Therefore, the results can be used to prove or bust data-mining myths such that other data miners can use these sharpened rules of thumb knowing that they are validated. In this thesis two different data-mining myths are researched: "Data preparation is more important than algorithm selection" and "Non-linear models perform better than linear models". The thesis is therefore divided into two parts, each one discussing a single myth and a general related work, used methodology for both myths and a conclusion about the two myths. The myths will both be researched using just under 400 data sets which gives us a wide and varied amount of data sets to study the myths with.

The first myth "Data preparation is more important than algorithm selection" is discussed in Chapter 4, in which we focus on feature selection, a smaller part of data preparation, because of how broad the term data preparation is. Feature selection is the process of selecting a subset of features that are relevant to build a model. The goal is to remove irrelevant or redundant features, where real world data sets can be full of. This makes feature selection simplify the data sets increasing the interpretability, shorten training times because of a fewer amount of features and reduce overfitting that may be caused by additional features. As such feature selection has been the focus of much research discussing the benefits and the methodology of efficient feature selection [14]. To research this myth we will run twelve algorithms with and without feature selection over all the data sets. Afterwards we compare the results for a given data set of an algorithm combination with and without feature selection, this is done for all the data sets and algorithm combinations with the results being represented in the form of graphs. The results will be further used for a meta-learning experiment to predict when feature selection will be preferred.

The second myth "Non-linear models perform better than linear models" will be discussed in Chapter 5. A linear model is a learner that classifies data using a function that can be graphically represented as a straight line or plane that divides the data into two parts. A non-linear model however classifies data in a way that can not be graphically represented by a straight line or plane but can be represented as all kinds of non-linear functions like a parabola or a sphere. Therefore most linear models are said to be simple and non-linear models are a lot more complex, allowing non-linearly separable data to be classified. The simple linear models might be worse because the data is underfit by the linear function and non-linear models allow for a more complex function but that can lead to overfitting. To research this we will compare the performance of linear and non-linear models on every data set used in the experiment. The results will also be used for a meta-learning experiment to obtain new insights when to use a linear or non-linear model.

Chapter 2 discusses related work; chapter 3 discusses our approach to proving the myths; chapter 4 discusses the myth "Data preparation is more important than algorithm selection"; chapter 5 discusses the myth "Non-linear models perform better than linear models"; chapter 6 concludes.

Chapter 2

Related Work

In this chapter the related work will be discussed, first meta-learning will be discussed followed by a section about OpenML.

2.1 Meta-learning

Meta-learning, in the context of machine learning, is the process of learning to learn and applies learning algorithms on meta-data which is generated by machine learning and combining data sets in one. There are different approaches on meta-learning, for example discovering meta-knowledge by using data and algorithm characteristics. Another approach is stacked generalisation [34] which works by combining different learning algorithms. The primary goal of meta-learning is understanding the interaction between the mechanism of learning and the contexts in which that mechanism is applicable [13].
The Algorithm Selection Problem [28] is a common meta-learning task, where the problem is to find the best learning algorithm on a given data set. Using machine learning the objective is to find for a given instance and output the connection between features of the instance and the output. Through generalizing from the examples in the training set, the algorithm can also predict instances that are not trained on. Algorithms that can generalize well using the training set for a given data set vary on a problem per problem basis and no algorithm is the best for all the problems [35]. Therefore with meta-learning we try to find using past knowledge and knowledge about similar problems and features of the data sets the best or a ranking of the best algorithms for a given problem. The best learning algorithm can be found by training a meta-model on a so called meta-data set which contains the performance of different algorithms on different data sets, characterized by meta-features [5, 22, 24]. The meta-features can be categorized as simple (number of attributes, dimensionality), statistical (mean standard deviation of numeric attributes), information theoretic

(class entropy, noise to signal ratio) or landmarkers (performance of simple fast to run classifiers) [23]. The dimensionality is calculated by dividing the amount of attributes by the number of instances in the data set, the mean standard deviation of numeric attributes is calculated by computing the standard deviation of all the numeric attributes in the data set and calculating the mean of it. Table 2.1 shows some examples of traditional meta-features which are also used in the experiments that will be described further in Chapter 3.

Table 2.1: Standard Meta-features.

Category	Meta-features
Simple	# Instances, # Attributes, Dimensionality, Default Accuracy, % Observations With Missing Values, % Missing Values, # Numeric Attributes, # Nominal Attributes, # Binary Attributes
Statistical	Mean of Means of Numeric Attributes, Mean Standard Deviation of Numeric Attributes, Mean Kurtosis of Numeric Attributes, Mean Skewness of Numeric Attributes
Information Theoretic	Class Entropy, Mean Attribute Entropy, Mean Mutual Information, Equivalent Number Of Attributes, Noise to Signal Ratio
Landmarkers [23]	Accuracy of Decision Stump, Kappa of Decision Stump, Area under the ROC Curve of Decision Stump, Accuracy of Naive Bayes, Kappa of of Naive Bayes, Area under the ROC Curve of Naive Bayes, Accuracy of k-NN, Kappa of k-NN, Area under the ROC Curve of k-NN

2.2 OpenML

OpenML [31] is an online experiment database, where machine learning researchers can share data, results and studies and organize it to work more effectively to collaborate on a global scale. Networked science tools allow researchers and scientists to discuss complex ideas and share results online and should make it possible for anybody to contribute however they want. OpenML allows this by being open for everyone by being able to use every bit of data stored in the database freely and uploading your own contributions to support online discussions. This is the goal of OpenML, creating a place for machine learning researchers to collaborate, work more effectively and be more visible to each other. Other open data repositories like UCI [2] already exist, but UCI does not allow multiple researchers to constantly share data via a platform. Kaggle [6] is an other data mining platform but Kaggle is more competitively oriented instead of shifting the focus on collaboration between researchers like OpenML does.

We use OpenML for obtaining all the data sets that will be used in the experiments. Using OpenML we can easily look up our own results of classifiers and those of others to gather the data needed for our research. Users can download and share data on OpenML through different popular machine learning platforms like Weka [15], Moa [3], RapidMiner, R [17] and Python. The OpenML API for Java allows users to easily connect to the online experiment database and use query's to collect relevant data for their research. All the results of this thesis can be found on OpenML such that it is available for further research of others.[1]

[1] http://www.openml.org/s/20

Chapter 3

Methods

For both experiments the same methodology is used to research the data-mining myths. The experiments are split up into two parts, first we run all the chosen algorithms on the data sets used in the experiments for initial results. These results are used to empirically establish whether the myth was valid or not, this is followed up by an univariate analysis on the predictability of some meta-features of the data sets. In the second part we will use meta-learning to find additional information and try to see if it is possible for a machine learning algorithm to learn when to apply the myth using only meta-features about the data set.
The data sets used in the experiments are all data sets that are publicly available on OpenML. The data sets that we picked are all data sets with a binary target containing between the 10 and 200,000 instances such that the data sets are not small or extremely big. In OpenML there were in total 394 different data sets that matched these criteria. Data sets with a binary target were selected because the Area under the ROC Curve value will be measured for the scores. The Area under the ROC Curve score shows the performance of a model over a binary target by plotting the true positive rate against the false positive rate. This means that a random guesser will have a score of 0.5 and a good model which have a high true positive rate and a low false positive rate will score higher than 0.5 up to a maximum of 1.0. The Area under the ROC Curve score is used instead of the accuracy score because some data sets have a very skewed target class distribution where majority class vote models might score high on accuracy but are not indicative of the actual difficulty of the data set. All the data sets that are used in the experiments are tagged on OpenML with '$study\_20$'.
For the first part of the experiment, the exploration experiment, we first gather all the algorithms needed to research the myth and then run them on all the data sets. We then measure the Area under the ROC Curve score on a given data set for all the algorithms and calculate the difference between certain algorithm combinations dependent on what is researched to see which algorithm performed better on the given data set. The difference is measured on all the data sets and the results are shown in various graphs. An univariate analysis will be constructed on a few features of the data sets to investigate whether the feature has a direct

relation with the target class. With the univariate analysis we focus on a single variable or meta-feature of the data sets for example the amount of features or the dimensionality and show the relationship between the amount of features or the dimensionality of the data set and the percentage that a non-linear model was preferred for example.

In the second part, the meta-learning experiment, we use the results of the first part and use it to learn more about the relations of the meta-features and search for a learning algorithm that is able to build a model that can accurately predict when the myth should be applied. The meta-learning experiment will be set up by making a meta-data set formed out of the results of the previous experiment and the meta-features of the used data sets, for example the number of features and the dimensionality of the data sets. As an example the header of an ARFF file of one of the meta-data sets used in the experiments containing the simple meta-features is below.

```
@relation 'mythbusting-featureselection'

@attribute alg.name {J48,RandomForest,NaiveBayes,IBk,HoeffdingTree,SGD,
REPTree,JRip,SMO,MultilayerPerceptron,AdaBoostM1,Logistic}
@attribute alg.type {trees,rules,bayes,lazy,functions,meta}
@attribute locality {local,global}
@attribute DefaultAccuracy numeric
@attribute Dimensionality numeric
@attribute NumberOfBinaryFeatures numeric
@attribute NumberOfFeatures numeric
@attribute NumberOfInstances numeric
@attribute NumberOfNumericFeatures numeric
@attribute NumberOfSymbolicFeatures numeric
@attribute PercentageOfInstancesWithMissingValues numeric
@attribute PercentageOfMissingValues numeric
@attribute feature.select {yes,no}
```

Every instance in the meta-data set in the example is an algorithm that was ran on a data set including the meta-features of the data set with as target if the model with feature selection was better than the one without. Therefore is every instance in the meta-data set a result of the first part of the experiment. The models created by the learning algorithms on the meta-data set will show whether it is possible to determine when the myth is valid when given only the meta-features of the data sets and the used algorithm. Afterwards we will closely look at the meta-data set to find the important sets of meta-features that are influential for building an accurate model about the myth.

To complement the existing meta-features (shown in Table 2.1) we introduce an additional type of meta-features for use in the experiments, we will call these meta-features *feature selection landmarkers*. The new meta-features add to the information of the existing landmarkers by adding the same classifiers but now with feature selection. The classifiers used are all fast and inexpensive to run on the data sets and the added feature selection does not impact the speed of the classifiers much. The difference in score of the classifier with feature selection is subtracted by the classifier without feature selection to give the value of the new landmarker. These new *feature selection landmarkers* might prove useful for the data preparation myth and will also be used in the linear model myth.

Weka [15] is used as our machine learning platform of choice because the OpenML API supports Weka and we have prior experience in using Weka. Weka is a popular machine learning software which contains a large collection of different algorithms and tools for predictive modelling and data analysis. There are multiple data mining tasks that Weka is able to support like data preprocessing, clustering, classification, regression, visualization and feature selection provided the data set is inserted into Weka. The OpenML API enables us to choose the data sets that we want to use for the research and use the algorithms provided by Weka to create new models. All the algorithms used in the experiments are available in Weka without extra dependencies, except for the LibSVM algorithm.

Chapter 4

Data preparation is more important than algorithm selection

In this chapter we will show the results of the myth "Data preparation is more important than algorithm selection" focussed on feature selection. It is researched by setting up two large scale experiments that span just under 400 data sets.

This section contains the introduction; section 4.2 discusses related work; section 4.3 and section 4.4 evaluates our contributions; section 4.5 concludes.

4.1 Introduction

Data preparation is a vital part in machine learning, some say it is the most important step in machine learning, therefore most data mining practitioners tend to spend a lot of time on extensive data preparation. Data preparation however is a vast and expansive term, therefore the focus in the myth will be on feature selection.

Feature selection is an important aspect of data preparation and valuable for classification problems, because data sets, especially real world data sets, can be full of irrelevant features. These irrelevant features will be removed with feature selection to counteract the curse of dimensionality which is caused by a high amount of features. Feature selection also helps the interpretability of the data sets, speed up algorithms that are reliant on the amount of features and help against overfitting.

However algorithm selection is also important and is a big process in machine learning because it depends on the size, quality and type of data. Also due to time constraints using all sorts of different algorithms is

not always feasible. Therefore a good algorithm selection process is vital for efficient machine learning.
Data preparation because of the large amount of approaches to prepare and clean the data seems to be more important than extensive algorithm selection therefore the myth "Data preparation is more important than algorithm selection" was born. In this chapter we will investigative over just under 400 data sets for every data set and a given algorithm whether or not feature selection will improve the results. The large scale of used data sets give a broad and varied set of results will give us the general effect of feature selection on data sets. Other approaches could be to pick only one real world data set and a huge amount of algorithms to test the results of feature selection on the data set, or set up two methods for approaches on the data sets, one with an extensive data preparation process and a small algorithm selection process and the other method the other way around and measure the difference in performance of both methods.
To research this two large scale experiments were done with 394 data sets, in the first experiment it is researched per algorithm if it is better to use feature selection or not by comparing the Area under the ROC Curve values of a given algorithm data set combination with each other. The second experiment is a meta-learning experiment, the results of the first experiment are used to create a meta-data set that allows us to build a model that learns when feature selection is preferred and find the optimal set of meta-features for building such a model. A paper on the experiments in this chapter has appeared in [25].

4.2 Related Work

In this section we will show some related work concerning the myth "Data preparation is more important than algorithm selection", starting with related work about feature selection followed by algorithm selection.

4.2.1 Feature Selection

Feature selection is a small part of data preparation and is beneficial for data because of better readability of the data sets through less features, generalization of the data which can better the results of some algorithms that tend to overfit and the learning speed goes up of certain algorithms that are reliant on the amount of features.
There are multiple extensive papers written about the benefits and methods of feature selection [4,7,9,14,20]. The goal of feature selection is to create a more accurate and therefore better model using a smaller subset of the total amount of features. Feature selection also alleviates some of the problems that the curse of dimensionality brings. The curse of dimensionality is directly linked to the amount of features, algorithms that work fine in low dimensions, a low amount of features, may overfit the data or suffer from large variance error in high dimensions [10]. In higher dimensions it is much harder to find similarities between features

because there might be a lot of noise created by irrelevant features and even if all the features are relevant then there is the problem of increasing the likelihood of high correlations across features. To find a smaller optimal subset of features a search method needs to be created that can be evaluated such that it can be measured against other subsets of features.

A simple approach to feature selection would be an exhaustive search algorithm that iteratively evaluates all sub sets of features, but this is not feasible for most data sets because of the quantity of features so other approaches are needed. A different approach would be to pick the features with the highest amount of predictive power like for example the information gain of the feature. However this is not optimal because this approach does not take in account that some features with low predictive power might have good synergy with other features and in turn have a lot higher predictive power, or some features that have a high predictive power correlate to other selected features and using both features is not adding value over using one feature. Therefore, a subset evaluator might be preferred instead of single feature ranker [14, 16]. Some algorithms have a built in form of feature selection for example algorithms that build decision trees and rules, which select the best features to split on, a few of these algorithms will also be tested in the experiments.

4.2.2 Algorithm Selection

Algorithm selection is important for building a good model on the data set or any other problem that needs to be solved. Although finding an effective, good or the best algorithm for a given problem is a large task and not easy considering the differing amounts of algorithms with highly varying run-times and algorithm performance varies wildly for every problem, therefore the Algorithm Selection Problem [28] arises.

A commonly used simple approach for algorithm selection is to run a certain amount of algorithms on the problem, then compare the performance of the algorithms and choose the best one. But it can be very time consuming to run all the algorithms on the problem and is very dependent on the set of algorithms chosen to run on the problem. There are also selection methods that do not search for the algorithm by just comparing their performance with other algorithms, like boosting or portfolio approaches [21, 30]. Meta-learning is an alternative method and uses automatic algorithm selection by constructing a training meta-data set using the meta-features of data sets and choosing a set of base-level learning algorithms which the meta-learner can choose from [13]. After an algorithm is selected then the next step is tuning the parameters of the chosen algorithm, tuning too much however may result in overfitting or underfitting.

4.3 Explorative experiment

In this section the experiment and the results of the explorative experiment will be explained and concluded with a discussion about the results.

4.3.1 Experiment

The algorithms that are chosen for the experiments are shown in table 4.1. These 12 algorithms are picked because they are widely used, are basic algorithms and give a coverage of most types of algorithms applied in machine learning. The algorithms are evaluated over the data sets using 10-fold cross-validation, with and without feature selection therefore we have 24 different results in total. The difference in performance between the algorithms with and without feature selection is measured in Area under the ROC Curve score. The feature selection algorithm that is used is Correlation-based Feature Subset Selection (Cfs-SubSetEval) [16].

Table 4.1: Algorithms used in the experiments of the first myth

(a) All algorithms are as implemented in Weka 3.7.13 [15] run with default parameter settings, unless stated different

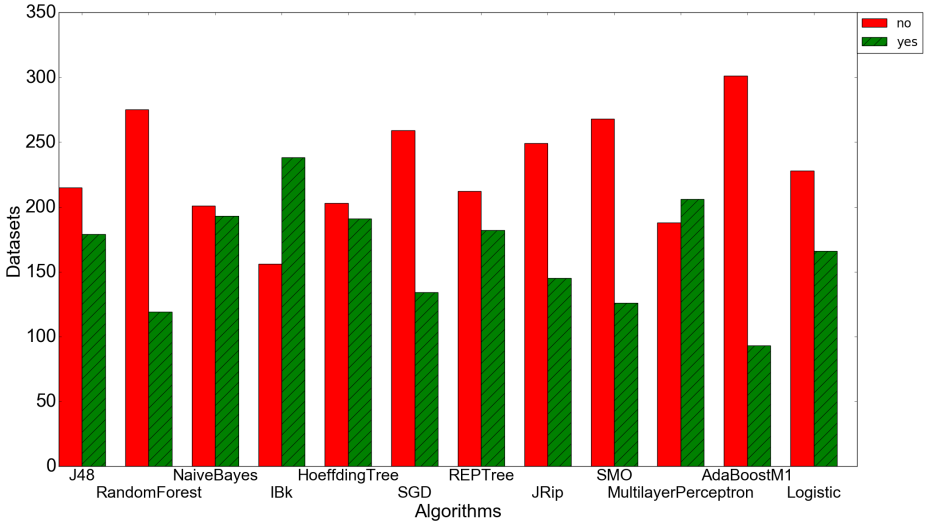
Algorithm	Model type	Parameter settings
Naive Bayes	Frequentist	
IBk	k-NN	$k = 1$
Stochastic Gradient Descent (SGD)	SVM	
Sequential Minimal Optimization (SMO)	SVM	Polynomial kernel
Logistic	Logistic	ridge $= 0.00000001$
Multilayer Perceptron	Neural Network	1 hidden layer
JRip	Rules	
J48	Decision Tree	
Hoeffding Tree	Decision Tree	
REP Tree	Decision Tree	
RandomForest	Bagging	100 trees
AdaBoost	Boosting	100 iterations

Cfs-SubSetEval works by evaluating the worth of a sub set by considering the predictive power of each feature along with the degree of redundancy between them. Sub sets that are highly correlated with the target class and have a low correlation with other features are preferred. We also experimented with other feature selection methods like GainRatio and InfoGain rankers, but the differences in performance compared to no feature selection were marginal at best. The differences in performance when using Cfs-SubSetEval are a lot higher, also Cfs-SubSetEval is used because sub set evaluators are generally considered to be better than rankers, therefore we chose to use Cfs-SubsetEval for both the experiments.

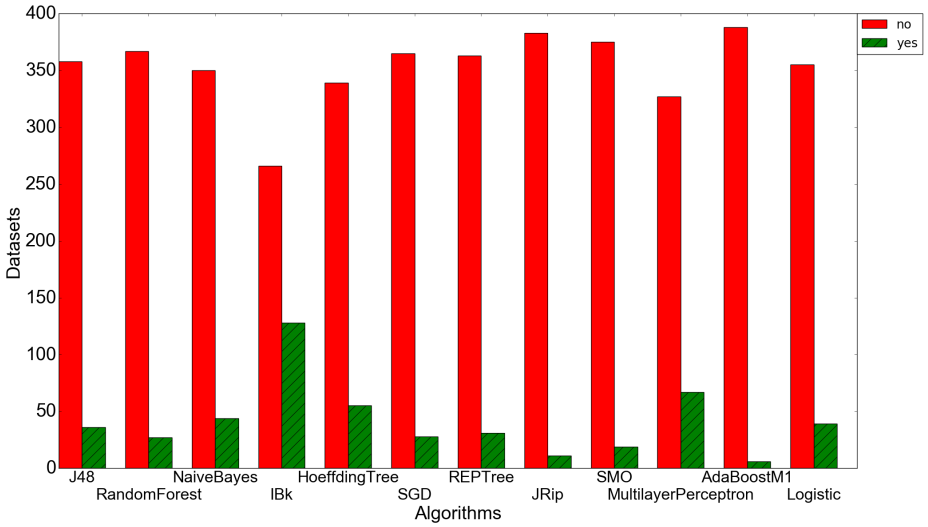
4.3.2 Results

Figure 4.1a shows per algorithm in how many cases feature selection yields better results for a given data set, Figure 4.1b shows when the difference is statistically significant with a double tailed T-test of 0.05. If the difference in score was not statistically better for feature selection, then it was counted as 'no feature selection' in Figure 4.1b.

In total 42 percent of algorithm data set combinations did feature selection improve the results, but only in 10 percent of the cases was this improvement statistically significant.



(a) Results for different algorithms on the data sets without a significance test

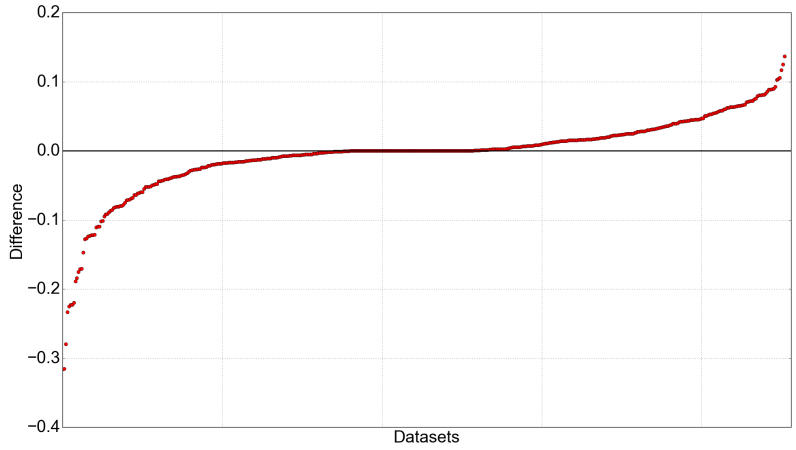


(b) Results for different algorithms on the data sets with a significance test applied

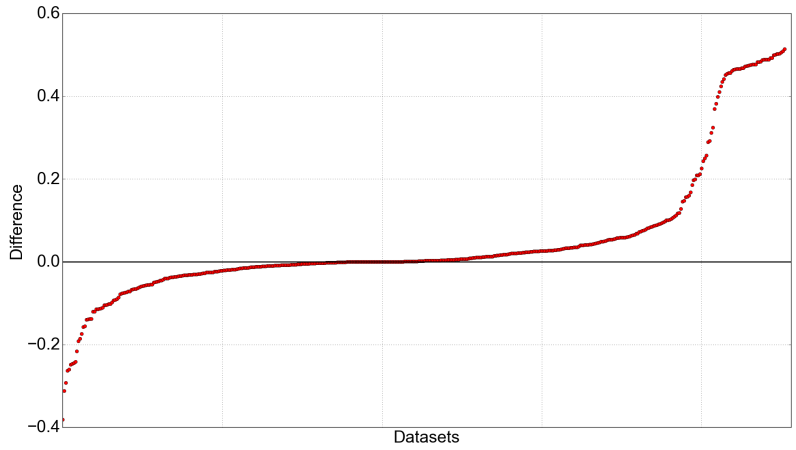
Figure 4.1: Number of data sets sorted by algorithm for which features selection improved the results in green (yes) and vice versa in red (no)

In Figure 4.1a we can see that for most algorithms the majority of the data sets are better with no feature selection, except for IBk and Multilayer Perceptron. For Hoeffding Trees and Naive Bayes there are barely more data sets where feature selection is worse. For J48 there are a large amount of data sets, 179 data sets to be precise, where separate feature selection is better, despite that it is a tree building algorithm and therefore has a form of feature selection embedded in the algorithm [26]. In Figure 4.1b however it is seen that none of the algorithms are significantly better with feature selection for the majority of the data sets. What can also be seen in Figure 4.1b is that the IBk algorithm has the most data sets where feature selection performs better in comparison with the other algorithms just like in Figure 4.1a.

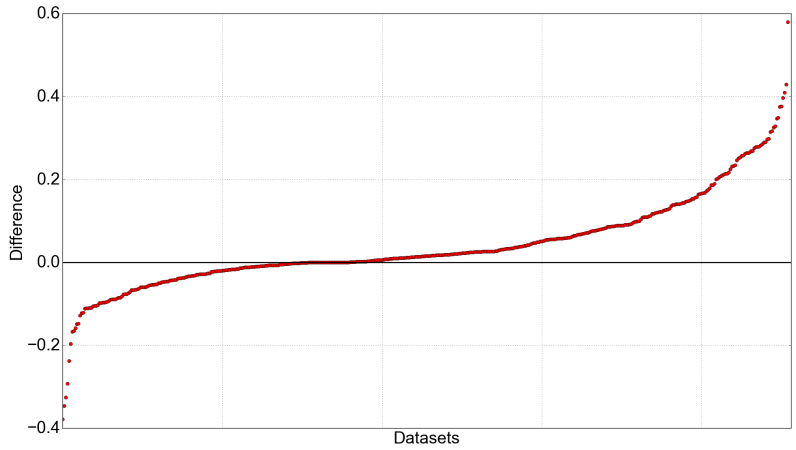
In Figure 4.2 the effect of feature selection is shown per data set. The y-axis shows the difference in Area



(a) J48



(b) Multilayer Perceptron



(c) IBk

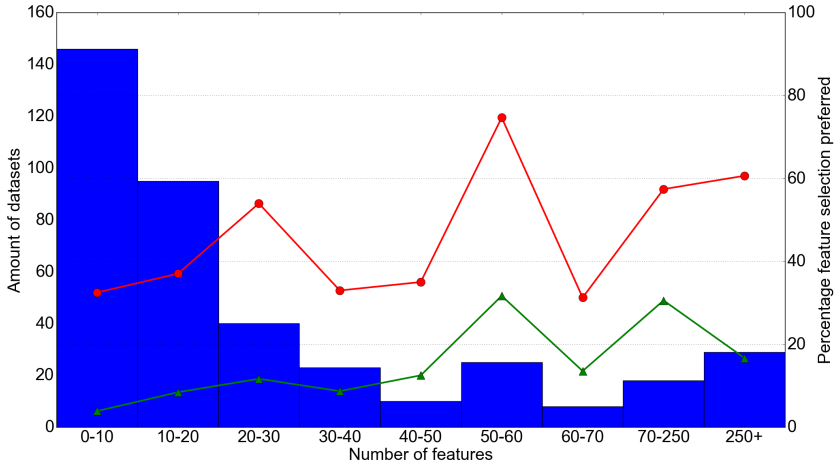
Figure 4.2: The effect of feature selection per data set on the x-axis for a given algorithm, sorted by difference in Area under the ROC Curve as the y-axis. When the difference is positive, the algorithm performed better after feature selection.

under the ROC Curve value of a given algorithm model on a data set, the difference is calculated as follows, the Area under the ROC Curve value of the model with feature selection is subtracted by the Area under the ROC curve value of the model without feature selection. The x-axis of the graph is the 394 different data sets used in the experiment and they are sorted by the difference in Area under the ROC Curve value to create a visual curve. A value above the 0-line means that feature selection resulted in a better Area under the ROC Curve value on the data set.
Figure 4.2a shows the effect of feature selection on the J48 algorithm, most of the other algorithms tested also look like this one. There are a large amount of data sets with no significant difference and a small amount where feature selection is better and in comparison there is a slightly bigger amount of data sets under the 0-line.
The Multilayer Perceptron algorithm results shown in Figure 4.2b are almost the same as the J48 curve but there is a large cluster of data sets where feature selection is a lot better. In figure 4.2c the results of the IBk algorithm on the data sets are shown, you can see here that by far most points are above the 0-line as would be expected from Figure 4.1a.

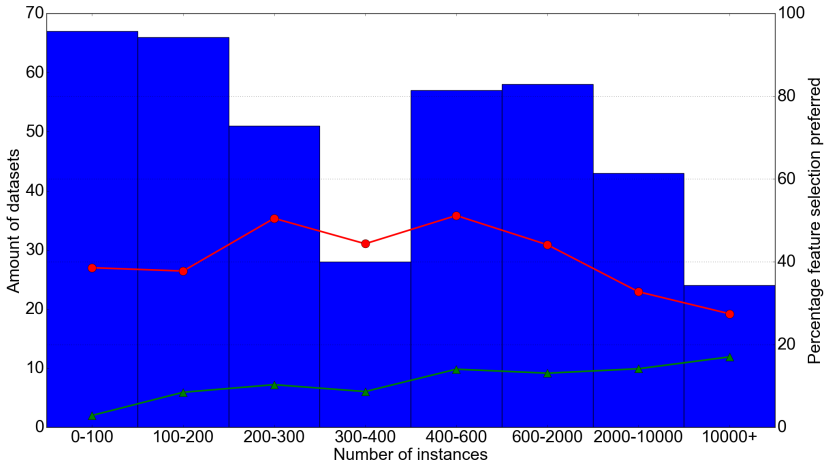
Figure 4.3 and Figure 4.4 show an univariate analysis on some features of the data sets and our own meta-feature. The features in the analysis are: number of features, number of instances, dimensionality (Number of features divided by the number of instances) and the *feature selection landmarker* of the kNN error rate.
The number of features do not seem to give a clear picture when feature selection is preferred. Figure 4.3b shows that the percentage of whenever feature selection is better that it is significantly better grows for a higher number of instances. In Figure 4.3c it is observed that a higher dimensionality, calculated by dividing the number of features by the number of instances, positively influences the percentage of preferred feature selection. The results of the kNN error rate feature selected landmarker in Figure 4.4 show that a higher error rate of the feature selected landmarker lowers the percentage that feature selection is preferred and vice versa. This is expected because a higher error rate of the feature selection landmarker shows that the feature selection model of the landmarker had more errors and that therefore feature selection was detrimental. A more in-depth analysis of the attributes of the data sets will be done in the meta-learning experiment in Section 4.4.

4.3.3 Discussion

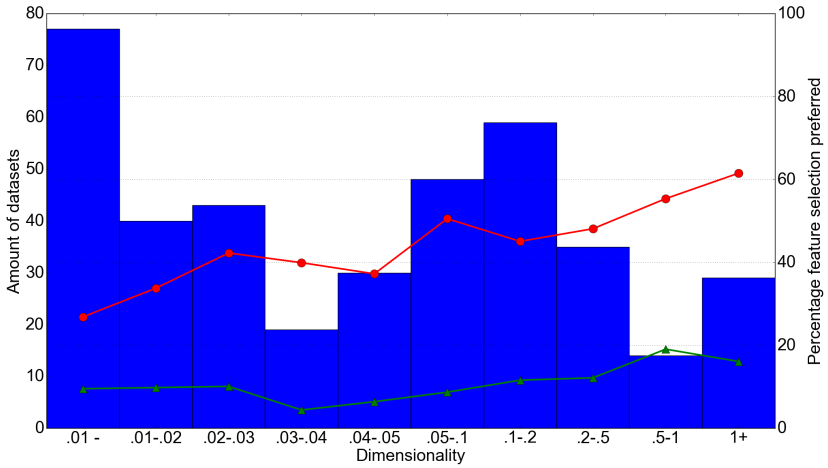
The results of the explorative experiment show interesting patterns and expected behaviour of the used algorithms. The IBk and Naive Bayes algorithms are two algorithms where feature selection was beneficial more than average as seen in Figure 4.1a and Figure 4.2c. This is expected because IBk suffers from the curse of dimensionality which means that the algorithm works unfavourable if there are a lot of attributes [27]



(a) Number of Features



(b) Number of Instances



(c) Dimensionality

Figure 4.3: The percentage that feature selection is preferred is plotted (the red line has no significance test and the green does) over some simple meta-features of the data sets in bar-graph form

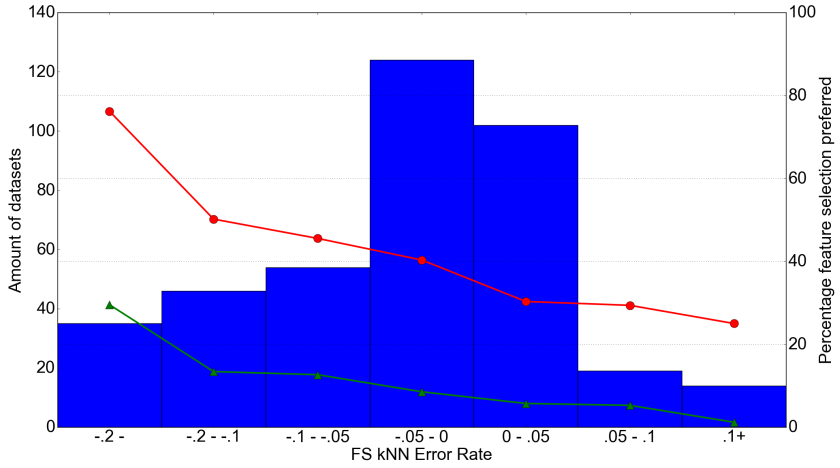


Figure 4.4: The percentage that feature selection is preferred is plotted (the red line has no significance test and the green does) over the feature selected kNN Error Rate of the data sets in bar-graph form

and Naive Bayes is negatively influenced by correlated features because a Bayesian model assumes that no features are correlated [18].

The results in Figure 4.1a and Figure 4.2a show that for an algorithm like J48 which has built-in protection against irrelevant features [26] still benefits from feature selection for a large portion of the data sets.

Multilayer Perceptron algorithms should learn on their own which features are relevant in the model [33], but Figure 4.1a shows that Multilayer Perceptron performs better with feature selection in quite a few data sets and in Figure 4.2b there is large cluster of data sets where the difference in Area under the ROC Value with feature selection is very high. The results also show the importance of the used algorithm when using feature selection.

Overall feature selection was better for a certain amount of data sets but the amount of data sets where the difference was statistically significant are sparse. It is possible that the data sets on OpenML have a bias for more cleaned up data sets with pre-selected features, what the amount of data sets with a low amount of number of features in Figure 4.3a may hint at. Therefore there might be a difference in results if more real world data sets would be used.

4.4 Meta-learning experiment

In this section the experiment and the results of the meta-learning experiment will be explained and concluded with a discussion about the results.

4.4.1 Experiment

In the next experiment we use meta-learning to learn if it is possible to consistently predict if feature selection for a given data set and algorithm will result in a better Area under the ROC Curve score. To achieve this we made a meta-data set using the results of the first experiment.
Every instance in the meta-data set are two 10-fold cross validation runs on a algorithm, one run with and one run without feature selection, and the target is whether the run with feature selection had a better performance.
We made 13 data sets, one for every algorithm and one where all other data sets are combined. These data sets are split further into 5 different sets with differing amounts of attributes. The attributes of the data sets are the different kinds meta-features as shown in Table 2.1. To add to this we also created some of our own attributes, *feature selection landmarkers*, that might be beneficial for the experiment. These new landmarkers complement the other landmarkers without feature selection by giving additional information about the data set by adding a simple classifier with feature selection. The result of the classifier with feature selection is subtracted by the one without showing the effect of feature selection on the classifier. The 5 different sub sets of the data sets are as follows :
The *simple* set contains only the simple meta-features (see Table 2.1), which are 10 attributes in total.
The *no landmarkers* set contains all simple, statistical and information theoretic meta-features (see Table 2.1), giving a total amount of 19 attributes.
The *default landmarkers* set contains all meta-features as seen in Table 2.1 without our feature selection landmarkers, raising the amount of attributes to 28.
The *feature selection landmarkers* set contains all meta-features except the standard landmarkers as seen in Table 2.1 but do contain our own feature selection landmarkers, this set also consists of 28 attributes.
The *all landmarkers* set contains all meta-features and both kinds of landmarkers, rounding up to 37 attributes.
Unlike the algorithm data set, the combined data set has three extra features: the algorithm name, the algorithm type and the locality of the algorithm. These are added for more information because they might positively influence the score.
As meta-algorithm we use Weka's RandomForest classifier with 100 trees.

4.4.2 Results

The results of the meta-algorithm over the meta-data sets is shown in Table 4.2. Every row in the table is a different algorithm and the bottom row is the combined data set with the extra algorithm meta-features. The columns represent the different sets of the data set. The results are expressed in Area under the ROC curve

Table 4.2: Area under the ROC Curve scores for various sets of meta-features on different sub sets of the meta-data set

Data set	Simple	No LM	Default LM	FS LM	All LM
J48	0.705	0.703	*0.737*	0.733	0.731
IBk	0.680	0.700	0.750	0.768	*0.783*
Multilayer Perceptron	*0.734*	0.704	0.708	0.711	0.710
Logistic	0.623	0.625	*0.711*	0.676	0.695
SMO	0.642	0.632	0.695	*0.713*	0.704
SGD	0.705	0.698	0.736	*0.746*	0.733
Hoeffding Tree	0.612	0.617	*0.679*	0.647	0.670
REP Tree	0.593	0.573	0.614	0.591	*0.621*
Naive Bayes	0.620	0.660	0.714	0.708	*0.721*
JRip	0.590	0.581	0.595	0.616	*0.639*
AdaBoost	0.623	0.634	0.638	0.649	*0.668*
RandomForest	0.712	0.722	0.764	0.774	*0.784*
Total data set	**0.704**	**0.728**	**0.765**	**0.768**	**0.773**

scores. The sets with landmarkers consistently score higher than the ones without. The *default landmarkers* set and the *feature selection landmarkers* set have similar scores over the data sets with only small differences with one set having a slightly better score depending on the data set. The *all landmarkers* set performs the best on most of the data sets.

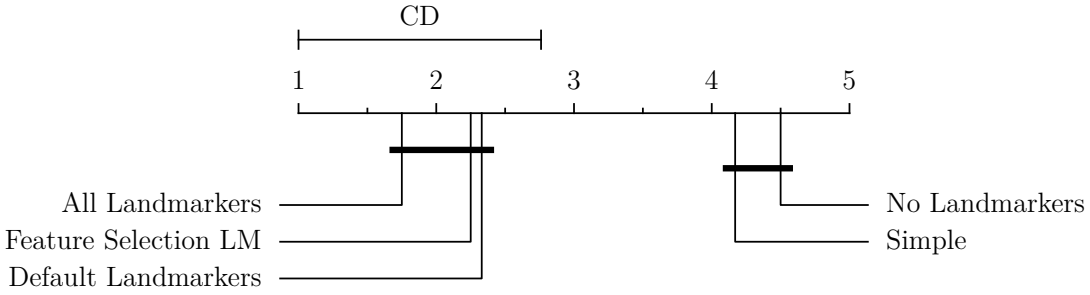
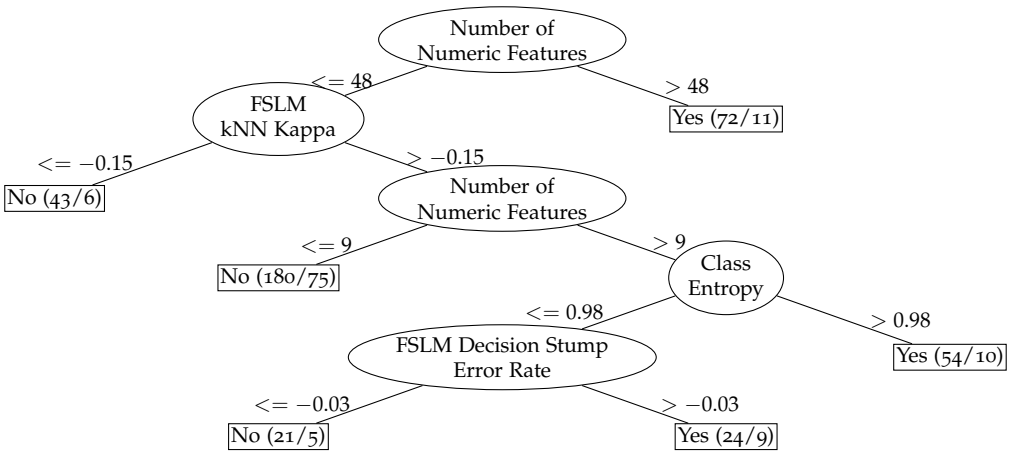


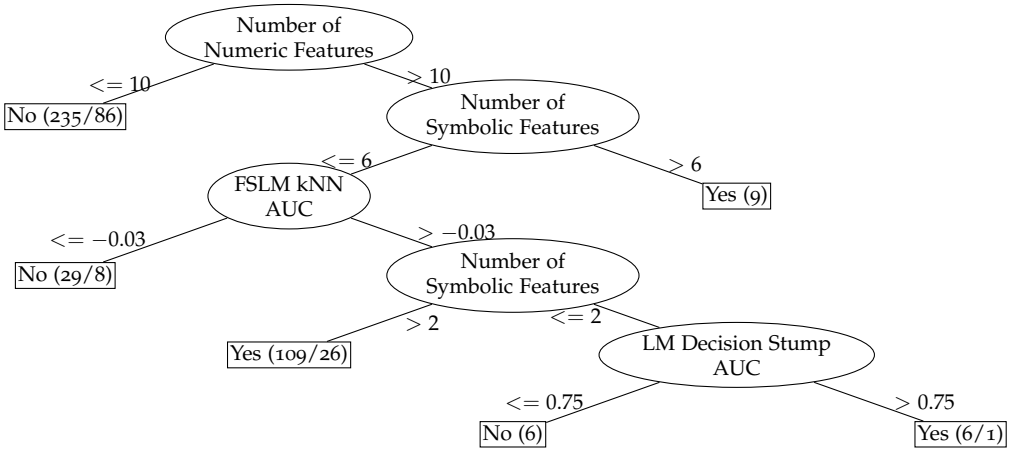
Figure 4.5: Results of Nemenyi test. Sets of meta-features are sorted by their average rank (lower is better). Classifiers that are connected by a horizontal line are statistically equivalent

To investigate whether the differences in performance is significant between the different sets and to clearly see how each set performs we made a statistical test as shown in Figure 4.5.
As can be seen the landmarkers sets are statistically better than the *simple* and *no landmarkers* set. This gives evidence that landmarkers do positively influence the meta-classifier. It is also visible that the *all landmarkers* set performs better than the separate landmarker sets. Also to note is that the *simple* set performs better than the *no landmarkers* set which has more meta-features.

Figure 4.6 shows two pruned J48 decision trees over the Multilayer Perceptron and Hoeffding Tree data sets to determine when to use feature selection. 'Yes' meaning use feature selection and 'No' means that no feature selection is better.



(a) Multilayer Perceptron



(b) Hoeffding Tree

Figure 4.6: Decision trees determining whether to use feature selection with a Multilayer Perceptron and Hoeffding Tree. Each leaf node contains the amount of correctly classified instances and the amount of misclassified instances

It is seen that both trees first split on the Number of Numeric Features and this attribute even appears twice in the Multilayer Perceptron tree. Number of Symbolic Features also appears twice in the Hoeffding Tree tree this suggests that the simple meta-features are important features. Both types of landmarkers also appear in the tree further proving that the landmarkers are beneficial for building a model with high predictive power.

4.4.3 Discussion

As can be seen from the results it is possible for our meta-data set to find out whether feature selection is beneficial or not. Because compared to the majority class prediction, which always has a Area under the ROC

Curve value of 0.5, our meta-data sets always score higher, even the *simple* and *no landmarkers* set as seen in Table 4.2.

The benefits of landmarkers for determining whether to use feature selection are also high as seen in the Nemenyi test (Figure 4.5) and the decision trees (Figure 4.6). These landmarkers are also best used if they are combined with the feature selection landmarkers and the existing ones. The most important deciding factors for feature selection are the number of attributes, the relative difficulty of the task as measured by landmarkers and the used algorithm.

The used algorithm is an important meta-feature for predicting when to use feature selection, for certain algorithms like IBk it will likely be more beneficial to use feature selection with a relative lower amount of features in the data set compared to the SGD algorithm for example.

4.5 Conclusions

The myth that we investigated in this chapter was "Data preparation is more important than algorithm selection" with a focus on feature selection specifically. The explorative experiment shows that feature selection is only better in 42% of the data set algorithm combinations of which only 10% is significant. Even though, feature selection may positively influence the run-time of the algorithms and the Area under the ROC Curve score quite significantly as seen in Figure 4.2b. There are two algorithms where feature selection is preferred for the majority of the data set those are IBk with 60% of the data sets and Multilayer Perceptron with 52%, all the other algorithms have a majority of data sets where no feature selection is preferred. The two algorithms with the least amount of data sets with feature selection preferred are RandomForest with 30% and AdaBoostM1 with 24%. Therefore we conclude that for most data sets feature selection is worse or only marginally better which is very dependent on the used algorithm.

With the results from the explorative experiment we investigated further with a meta-learning experiment and found using a RandomForest meta-algorithm on the total meta-data set that it is possible to predict whether feature selection should be used for a given data set, even though with an Area under the ROC Curve score of 0.773 with the best meta-algorithm it is not always predicted correctly. The meta-data set was split up into different sub sets to determine the best set of meta-features, out of the results we saw that the sub set with all the meta-features (Table 2.1) including our own *feature selection landmarkers* was the best followed by the sub set with all the standard meta-features with just our own landmarkers and the sub set without our landmarkers but with the standard landmarkers. Out of our meta-learning experiment we found that to determine when feature selection should be used one should look at the number of features, the relative difficulty of the task as measured by landmarkers and the algorithm that will be used. This improves the importance of a good algorithm selection compared to pre-determining whether to use feature selection or not.

In conclusion feature selection is not as beneficial as we thought, it is even detrimental for the majority of the data sets for most of the tested algorithms. This might be because the data sets used in the experiments are mainly UCI [2] data sets which often have been cleaned by domain experts already. The importance of data preparation as a whole however can not be described using only feature selection. Therefore, we can not show the validity of the myth using only this set of experiments and more experiments should be set up including more aspects of data preparation.

Chapter 5

Non-linear models perform better than linear models

In this chapter the results of the myth "Non-linear models perform better than linear models" will be shown. This myth is researched by setting up two large scale experiments, one explorative and one meta-learning experiment, that span just under 400 data sets.

This section contains the introduction; section 5.2 discusses related work; section 5.3 discusses the explorative experiment; section 5.4 discusses the meta-learning experiment; section 5.5 concludes.

5.1 Introduction

Linear models are said to be too simple for complex problems and therefore non-linear models are preferred over linear models. This forms the basis for the myth "Non-linear models perform better than linear models". Even though non-linear models might seem to perform better they also have their own complications, because non-linear models tend to overfit and are generally more computationally expensive than linear models which might result in longer run-times. In contrary linear models are relatively simple but are prone to underfitting. The bias and variance trade-off is a way to deconstruct the type of error that is made by a given model [10]. The bias error is the tendency that model consistently learns the same wrong thing. While the variance error is the tendency for different models built on samples from the same population to give different results for a particular test instance. Simple linear models tend to have a high bias error because they consistently classify the target wrong if it can not be induced in a linear way. While complex non-linear models have a high

variance error which is caused by overfitting the data set. However we will not use the bias and variance trade-off in the experiments and we will just focus on the performance of the linear and non-linear models. In this chapter these two different types of models will be put to the test and we will compare their results over slightly less than 400 different data sets. The algorithms that are used to research this are Multilayer Perceptron and LibSVM, both of the algorithms can be tuned to create linear or non-linear models. As discussed in Chapter 3 we first create an explorative experiment after which the obtained knowledge is applied to a meta-learning experiment to try and learn when a linear or a non-linear model should be used and research which features of a data set will be beneficial for learning this.

5.2 Related Work

In this section we will show some related work concerning the myth "Non-linear models perform better than linear models", starting with related work about LibSVM followed by Multilayer Perceptron.

5.2.1 LibSVM

One of the two algorithms used in the experiments is a SVM algorithm [8]. LibSVM is a library for Support Vector Machines (SVMs) [32] and is one of the most widely used SVM software libraries. Support Vector Machines are used for supervised machine learning models to analyse data for classification and regression problems.

Support Vector Machines classify instances based on where they are located in a p-dimensional grid with p being the amount of features. The SVM first calculates every data point in the training data as a p-dimensional vector. Afterwards the SVM calculates a $(p-1)$-dimensional hyperplane that splits the data points into two categories where the distance of each data point per category is maximized, this way it can categorize new data points using the hyperplane. The resulting model is a binary linear classifier but the $(p-1)$-dimensional hyperplane is only calculable for linearly separable data.

However SVM models can also be made for non-linearly separable data and can be used to create non-linear models. This is done applying the kernel trick [1] to the hyperplanes. Instead of measuring the difference between the data points with a dot product, it is replaced by a non-linear kernel function. The resulting values of the data points with the non-linear kernel function allow a hyperplane to classify non-linearly separable data. This transformation by using the kernel trick can be non-linear and therefore non-linear models can also be made using Support Vector Machines. In Figure 5.1 is an example of non-linear separable data that can be classified by a hyperplane through applying the kernel trick.

Because LibSVM contains a library of different SVM models with Weka integration it allows us to easily use different types of SVM models for our research.

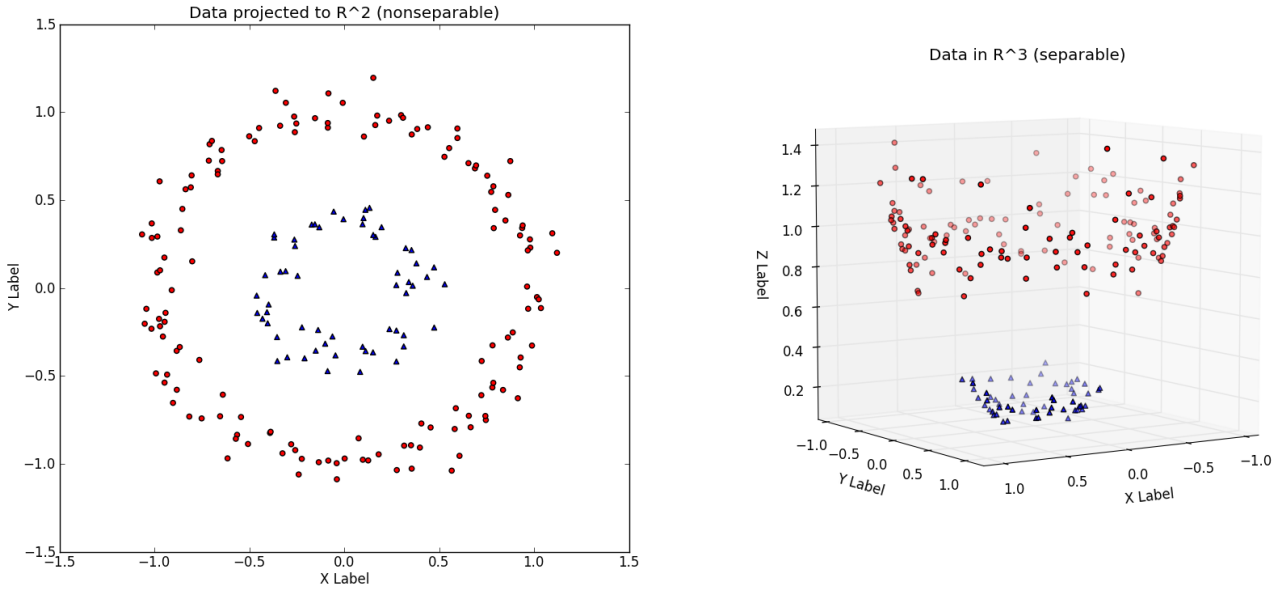


Figure 5.1: Left is a non-linearly separable data set. Right is the same data set transformed by using a kernel trick. Image taken from Eric Kim [19]

5.2.2 Multilayer Perceptron

Artificial neural networks are inspired by biological neural networks and are widely used because of their ability to learn using a set of inputs, this can also be used for building machine learning models. One type of neural network used in machine learning is the Multilayer Perceptron [12]. The Multilayer Perceptron algorithm used in the experiment is set to be a feed-forward network that uses backpropagation [29] to classify instances.

Feed-forward networks are layers of nodes in a directed graph without cycles, with each layer connected to the next layer. All the nodes, except for the input nodes, have an activation function which depends on the nodes before it and a given weight, the result of the model depends on the results of the activation functions on the output nodes. Backpropagation allows a neural network to learn and train itself by changing the weights and therefore the activation functions of the nodes in the network depending on the results of the output nodes.

The Multilayer Perceptron implemented in Weka is made out nodes with a sigmoidal activation function and allows us to change the size of the hidden layer, a hidden layer is a layer of nodes that is not the input or output layer. A neural network with no hidden layers is called a perceptron, the input nodes are directly connected to the output layer and therefore it is linearly separable.

Due to the flexibility in the hidden layer we can create a linear model by not having any hidden layer and create a non-linear model if we add a hidden layer. By being able to create a linear and non-linear model of the Multilayer Perceptron algorithm with ease, we decided to use the algorithm for our research.

5.3 Explorative experiment

In this section the experiment and the results of the explorative experiment will be explained and concluded with a discussion about the results.

5.3.1 Experiment

In this explorative experiment the same data sets are used as described in Chapter 3. As such the experiment was run over 394 different binary classification data sets widely varying in amount of features and instances.

Table 5.1: Algorithms used in the experiments of the second myth

(a) All algorithms are as implemented in Weka 3.7.13 [15] run with default parameter settings, unless stated different

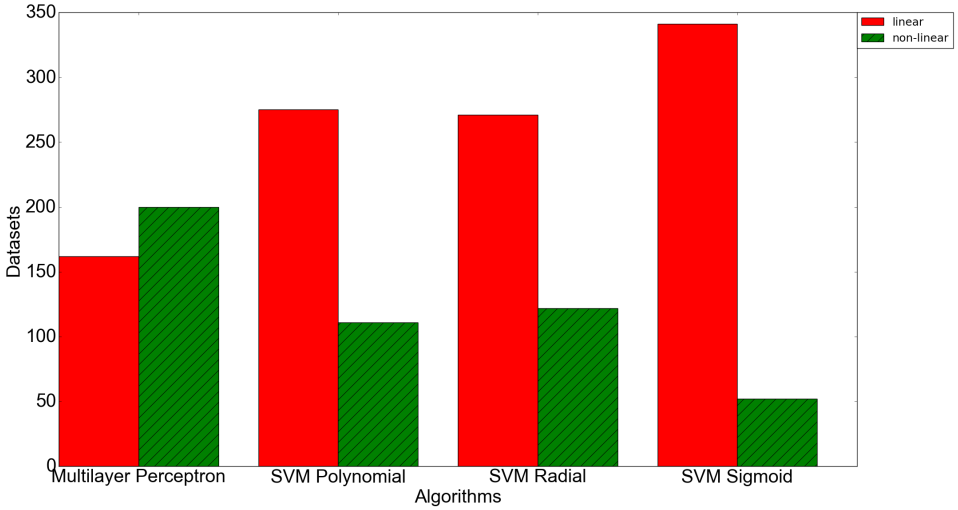
Algorithm	Parameter Settings	Linearity
Multilayer Perceptron	$h = a$ $n = 500$ ($a = (attribs + classes)/2$)	Non-linear
Multilayer Perceptron	$h = 0$ $n = 500$	Linear
LibSVM	Linear $k = 0$	Linear
LibSVM	Polynomial $k = 1$	Non-linear
LibSVM	Radial basis function $k = 2$	Non-linear
LibSVM	Sigmoid $k = 3$	Non-linear

The used algorithms with their respective parameters are shown in Table 5.1. For Multilayer Perceptron we use a single hidden layer with a size of $'a'$ for the non-linear model and a layer size of zero for the linear model, such that there are no hidden layers. An epoch is one iteration over the neural network while training after which the weights of the nodes can be adjusted depending on the output. An epoch size of 500 for the Multilayer Perceptron algorithm is chosen to reduce the run-time of the algorithm on some extremely large data sets due to time constraints, while still having a big enough epoch size such that the amount of training of the model is not too small. The LiBSVM algorithm has four pre-built kernel types, one of which being linear and the other three being non-linear.

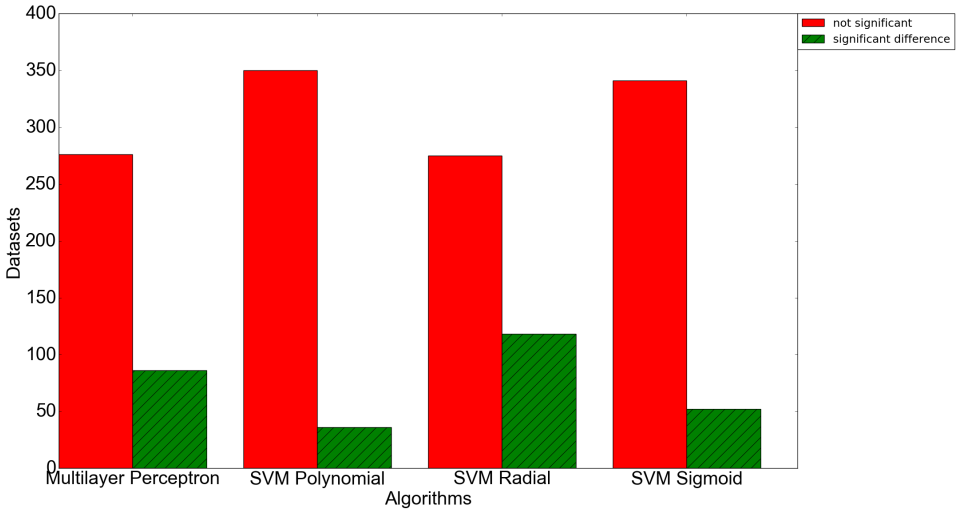
We will run all the different algorithms on all the 394 data sets using 10-fold cross-validation. Afterwards the Area under the ROC Curve score of the linear and corresponding non-linear model will be measured for all the combinations. The four different combinations are as follows:

The $h = a$ Multilayer Perceptron will be compared with the $h = 0$ Multilayer Perceptron with 500 epochs.

Every LibSVM algorithm that is non-linear will be compared to the linear LibSVM algorithm.



(a) Results of the algorithms on the data sets



(b) Results of the algorithms on the data sets with a significance test

Figure 5.2: Amount of data sets where 'linear' in red is better and 'non-linear' in green is better

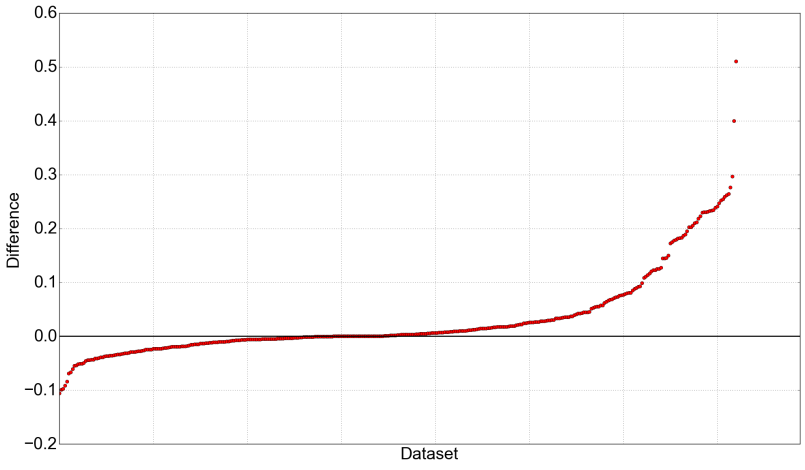
5.3.2 Results

In Figure 5.2a are the results of all the algorithm combination runs on all the data sets without a significance test and in Figure 5.2b are the same results but with a significance test (double tailed T-test of 0.05). In Figure 5.2b if the difference in performance was significantly better in favour of the non-linear model it is counted as 'non-linear' otherwise it is counted as 'linear'. Due to time constraints we were not able to run all the algorithms on all the data sets, resulting in only 362 data sets for the Multilayer Perceptron algorithm combination and 386, 393 and 393 data sets for the polynomial, radial and sigmoid algorithm combinations respectively

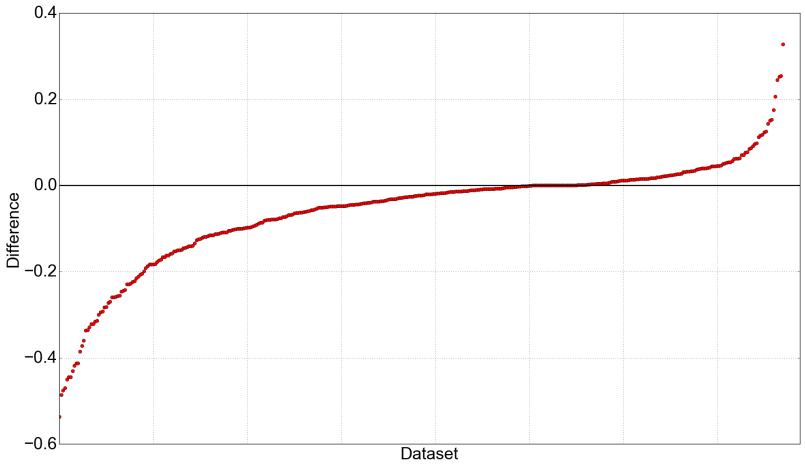
Without a significance test the Multilayer Perceptron non-linear model is the only model that is better on

more data sets than the linear model, to be more exact 200 data sets are better using a non-linear model and 168 data sets using a linear model. However all the LibSVM non-linear models score worse than the linear model over most data sets, especially for the sigmoid function where the non-linear model is better for only 52 of the data sets. After a significance test, where the results are seen in Figure 5.2b, it is seen that the majority of data sets are not significantly better with a non-linear model.

An interesting result of the significance test however, is that the radial and sigmoid kernels of LibSVM have about the same amount of data sets where the non-linear model is better compared to the results with no significance test (Figure 5.2). For the radial function there is a difference of only 4 data sets where the non-linear model was better but not significantly and for the sigmoid function the non-linear model was significantly better for all 52 data sets that it was better.

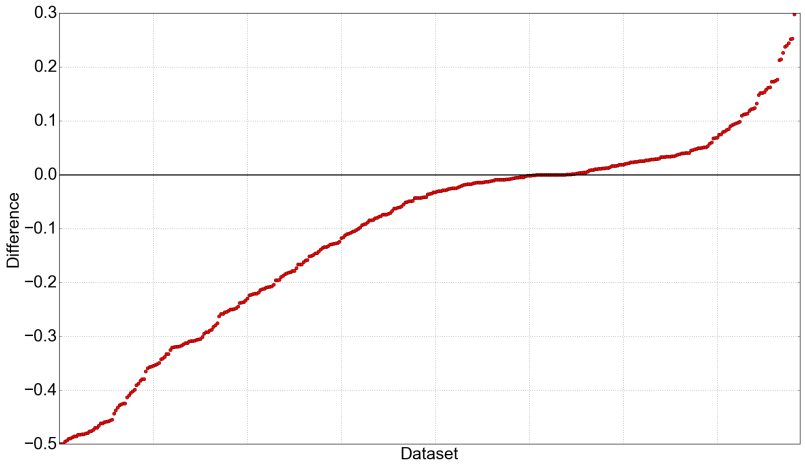


(a) Multilayer Perceptron with one hidden layer with a size of $'a'$

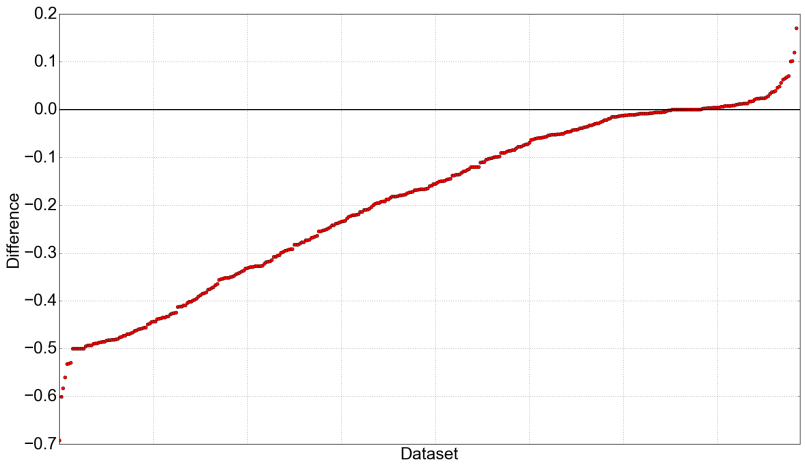


(b) SVM Polynomial

Figure 5.3: The effect of a non-linear model per data set on the x-axis for a given algorithm, sorted by difference in Area under the ROC Curve as the y-axis. When the difference is positive, the algorithm performed better with a non-linear model



(c) SVM Radial



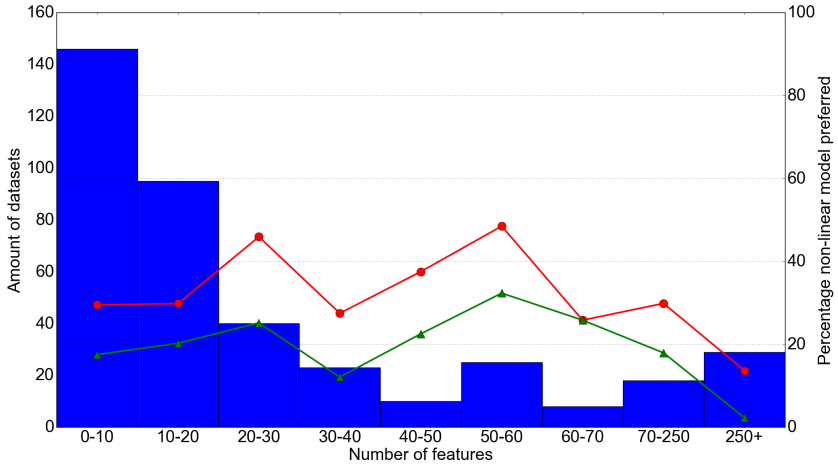
(d) SVM Sigmoid

The effect of the non-linear model per data set is shown in Figure 5.3. The Multilayer Perceptron graph in Figure 5.3a shows that the linear model is only slightly better when it has a higher Area under the ROC Curve value compared to the non-linear model because it is seen that the highest difference is only 0.1 when the linear model was preferred.

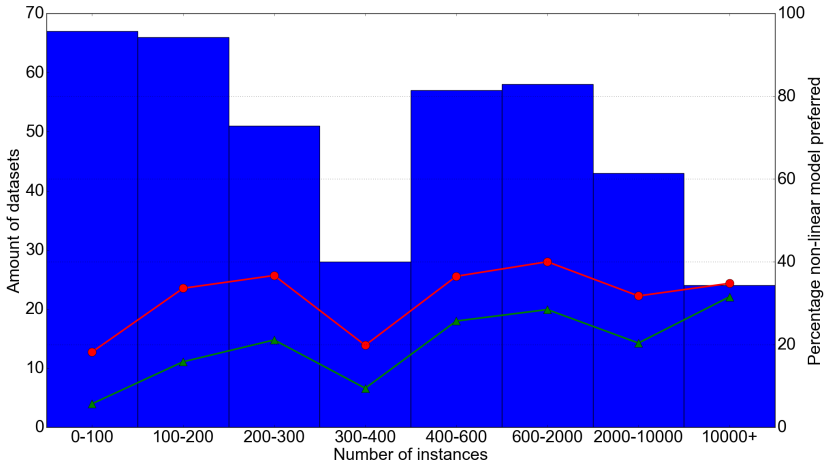
In the LibSVM graphs it is seen that there is a big difference in favour of the linear model for a large portion of the data sets, especially for the radial and sigmoid model in Figure 5.3c and Figure 5.3d respectively.

Figure 5.4 shows an univariate analysis on a few simple meta-features of the data sets from which we get an insight of how they might affect the models.

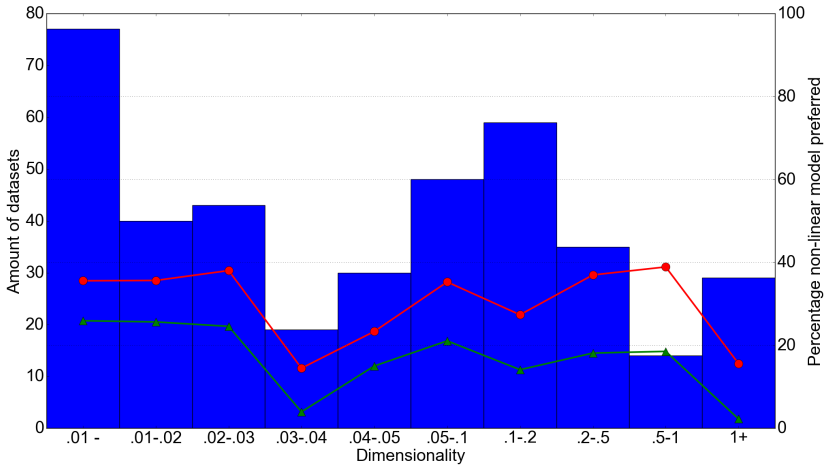
It is seen in the figures that the green line with a significance test closely resembles the red line without a significance test and the difference between the two lines is not big. This suggests that a lot of the data sets that do prefer a non-linear model are also significantly better compared to using a linear model on the data set. Another observation is that non-linear models tend to do worse with a higher number of features, dimensionality and number of numeric features because the percentage of data sets where a non-linear model



(a) Number of features

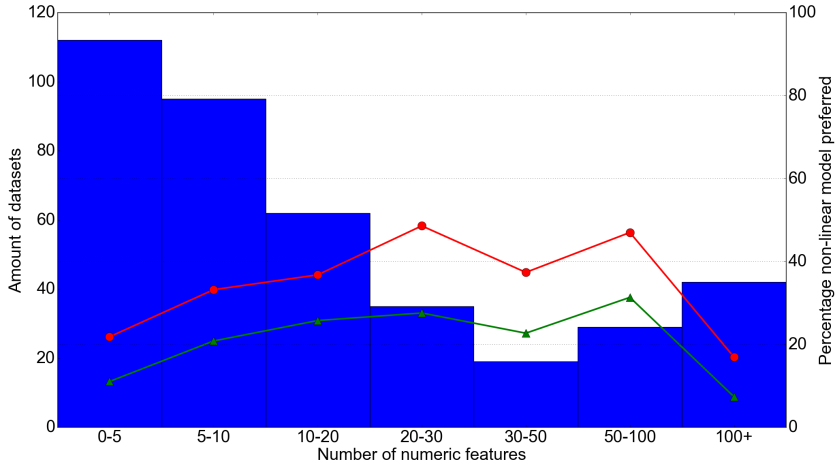


(b) Number of instances

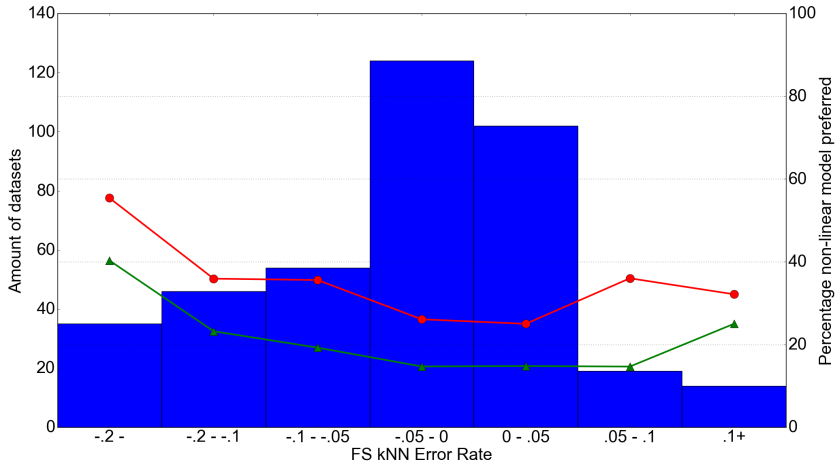


(c) Dimensionality

Figure 5.4: The percentage a non-linear model is preferred is plotted (the red line has no significance test and the green does) over some ranges of statistics about the data sets in bar-graph form



(d) Number of numeric features



(e) kNN error rate

is preferred goes down with higher values.

In the kNN error rate graph in Figure 5.4e there is a small effect, a lower error rate means that non-linear models tend to perform a little bit better.

All in all there is no clear case using a single meta-feature when non-linear models are always preferred.

5.3.3 Discussion

The results show some interesting patterns of the non-linear models. For example the Multilayer Perceptron non-linear model performs better than the linear model on most of the data sets while weirdly enough all the non-linear models of the LibSVM algorithm fail to exceed the linear model in Area under the ROC Curve score for the majority of the data sets as seen in Figure 5.2.

However when the LibSVM non-linear models perform better in Area under the ROC Curve score than the linear models, then there is likely a significant difference in performance, especially for the radial and sigmoid

kernel types for LibSVM.

A significant difference in performance of the linear and non-linear model for the LibSVM models is also observed in Figure 5.3, most of the data sets are under the 0-line and they even go as low as a -0.5 difference in Area under the ROC Curve score with the non-linear model in favour of the linear model implying that the differences in performance of the linear and non-linear models are large for most of the data sets.

This is not the case for the Multilayer Perceptron models however, if the non-linear model performs worse than the linear model in Area under the ROC Curve score then there is only a small difference upwards to just -0.1.

In Figure 5.4 it can be seen that it is not possible to accurately predict if a non-linear model is preferred using only single meta-features therefore we will set up an experiment in the following section to learn if this is possible with more features.

5.4 Meta-learning experiment

In this section the experiment and the results of the meta-learning experiment will be explained and concluded with a discussion about the results.

5.4.1 Experiment

In this experiment we will set up a meta-data set using the data from the previous experiment to learn the important meta-features need for predicting when to use a non-linear model and we will try to create a model using the meta-features that can predict when to use a linear or non-linear model for a given data set and algorithm.

For every combination of non-linear and corresponding linear model on a data set we compared the Area under the ROC Curve score and classified them as $'linear'$ or $'non-linear'$ depending on which model performed better. For the attributes of the meta-data set we used the meta-features of Table 2.1. The *feature selection landmarkers* that were created for the first myth in Chapter 4 are also added as attributes for the meta-data set, as they might be beneficial for the model.

The full meta-data set is also split in four smaller data sets for every combination of models (Multilayer Perceptron, LibSVM Polynomial, LibSVM Radial, LibSVM Sigmoid), the full meta-data set also has two extra attributes namely the name of the algorithm and the kernel type of LibSVM (null in case of Multilayer Perceptron instance).

Table 5.2: Area under the ROC Curve scores of a few meta-algorithms on the meta-data sets

Meta-data set	RandomForest
Multilayer Perceptron	0.751
LibSVM Polynomial	0.696
LibSVM Radial	0.801
LibSVM Sigmoid	0.799
Total data set	**0.851**

5.4.2 Results

The results of the RandomForest meta-algorithm over the meta-data sets is shown in Table 5.2. It is seen that the meta-data set can efficiently predict whether a linear or non-linear model should be used going as high as a Area under the ROC Curve score of 0.851 over the total data set. The lowest Area under the ROC Curve score was on the LibSVM Polynomial meta-data set with a score of 0.696, however this is better than a random guesser which would have a score of 0.5.

Table 5.3: Top 5 ranked attributes of the LibSVM Polynomial and Radial meta-data sets

Meta-data set	Polynomial	Radial
#1	Number of Symbolic Features	FS LM Decision Stump AUC
#2	FS LM Naive Bayes Error Rate	FS LM Naive Bayes AUC
#3	FS LM Naive Bayes Kappa	FS LM Decision Stump Error Rate
#4	FS LM Naive Bayes AUC	LM Decision Stump Error Rate
#5	FS LM Decision Stump Error Rate	LM Decision Stump AUC

Table 5.4: Top 5 ranked attributes of the LibSVM Sigmoid meta-data sets

Meta-data set	Sigmoid
#1	Number of Numeric Features
#2	LM kNN AUC
#3	LM kNN Error Rate
#4	Number of Features
#5	LM kNN Kappa

Table 5.5: Top 5 ranked attributes of the Multilayer Perceptron and total meta-data sets

Meta-data set	Multilayer Perceptron	**Total data set**
#1	FS LM Decision Stump AUC	Number of Numeric Features
#2	FS LM Decision Stump Kappa	LibSVM Kernel Type
#3	FS LM Decision Stump Error Rate	Number of Features
#4	Number of Instances	LM Decision Stump Kappa
#5	LM Decision Stump AUC	Algorithm Name

The top five attributes with the highest information gain are shown in Table 5.3, Table 5.4 and Table 5.5. Apparent is that the landmarkers are very influential especially our own *feature selection landmarkers*. The simple meta-features like Number of Symbolic or Numeric Features also score high in the rankings.

5.4.3 Discussion

From the results it can be seen that the meta-data set can actually be efficiently used to determine if a linear model on a data set is preferred. The results of the RandomForest meta-algorithm are promising, on the total data set it gives a score of 0.851 and it also scores higher than the majority class vote for the separate data sets as seen in Table 5.2. A reason why the total data set performs better is that the algorithm name is one of the attributes and thus can be generalised easier than the other meta-data sets without this attribute. Also to note is that the *feature selection landmarkers*, normal landmarkers and simple meta-features are the attributes with the highest info gain for deciding when a linear model is beneficial as seen in Table 5.3 and Table 5.5. It is interesting that the *feature selection landmarkers* have a high info gain because they were not specifically made for predicting whether to use a linear or non-linear model. It might be that whenever feature selection is preferred it means that the data set is more complex which is harder for a linear model to correctly classify.

5.5 Conclusions

The results of the explorative experiment show that a linear model is preferred for 68% of the data sets over all the algorithms as seen in Figure 5.2. The myth "Non-linear models perform better than linear models" is seen to hold for some non-linear models. The non-linear model of the Multilayer Perceptron algorithm for example performed better on the majority of the data sets without a significance test. However there was no majority with a significance test applied and even though there was no time to run the non-linear model on all the data sets for the experiment, the amount of data sets missing is not large enough for a majority in data sets with a significance test when using the non-linear model. Therefore there are non-linear models that do perform better than their linear counterpart, but as shown in the results of the non-linear models of the LibSVM which all score lower on a majority of the data sets by the linear model, it shows that the myth is not always true and the validity of the myth varies per non-linear model used.

The difference on how good a non-linear model performs compared to a linear model varies wildly per data set and per used algorithm as seen in the difference graphs in Figure 5.3. Even though when a certain model is preferred there is a high chance that the difference is quite significant, this is seen especially for the models created by the LibSVM algorithm, which might be because how much linear models generalize and non-linear models overfit depending on the data set.

Further observations that non-linear models tend to do worse if they overfit the data can be seen in the univariate analysis of the number of features in Figure 5.4a and the number of numeric features in Figure 5.4d where the percentage that non-linear models are preferred goes down with more features, suggesting that the algorithm is using all the features to create a non-linear model resulting in an overfitted model. It is found that overfitting is detrimental for the performance of the model also shows in the meta-learning experiment

where the simple meta-features like number of features and number of numeric features were one of the best attributes of the meta-data set for determining when a linear or non-linear model is preferred on a data set. We also found that a meta-data set can be efficiently used to determine whether or not a non-linear model is preferred given that the meta-algorithm of RandomForest gave an Area under the ROC Curve score of 0.851 in Table 5.2.

For further research a bias-variance error [11] analysis might be noteworthy in determining if a non-linear model is preferred, because linear models tend to have a high bias error and non-linear models a high variance error.

Chapter 6

Conclusions

In this thesis we tackled on two different myths, "Data preparation is more important than algorithm selection" and "Non-linear models perform better than linear models".
The first myth "Data preparation is more important than algorithm selection" was discussed in Chapter 4 with a focus on the feature selection aspect of data preparation. To set up an experiment between algorithm selection and feature selection we first gathered a large amount of differing data sets using OpenML [31]. We ended up with 394 data sets that would be used for all experiments. After running 12 different algorithms with and without feature selection on the 394 data sets we observe that the algorithms with feature selection performed better in 42% of the data set algorithms combinations. Whether feature selection is preferred for a given data set is seen to be very dependent on the used algorithm, for example feature selection is preferred for the IBk algorithm for 64% of the data sets compared to the AdaBoostM1 algorithm where only 24% of the data sets had a better Area under the ROC Curve value with feature selection.
To investigate the relation between the used algorithm and feature selection further a meta-learning experiment was done. Our own *feature selection landmarkers* were shown to be beneficial for deciding when to use feature selection especially when coupled with other landmarkers, the best model on the full meta-data set using all the landmarkers had a score of 0.773. The number of features and the used algorithm turned out to be important attributes in determining when to use feature selection. In conclusion feature selection turned out be detrimental for the majority of the data sets for most of the tested algorithms. The effect of feature selection on a data set is seen to be influenced by the used algorithm, but the results are not complete enough to compare algorithm selection with data preparation as a whole and thus can not be used to prove or bust the myth.

The second myth "Non-linear models perform better than linear models" was researched in Chapter 5. We picked four complex non-linear models in the form of a Multilayer Perceptron and LibSVM non-linear kernels

and two simpler linear models created of the Multilayer Perceptron and LibSVM linear kernel algorithms.
After all the algorithms were ran on almost all the 394 data sets we saw that in 68% of the cases the linear model was actually better than the more complex non-linear model. Especially the linear model of the LibSVM algorithm performed very well going up to an increase in Area under the ROC Curve score of 0.5 compared to the non-linear model on some data sets. The data sets however where the non-linear LibSVM models performed better than the linear models did almost all have a significant difference in performance compared to the linear model, especially for the radial and sigmoid kernel types. The non-linear Multilayer Perceptron model however did outperform the linear model in most of the data sets but the majority was not significant. The results clearly show that non-linear models do not always perform better than linear models and the performance of each model varies per data set.
To further research if it was possible to predict when to use a non-linear model using the meta-features of a data set we also performed a meta-learning experiment on the results. The experiment showed that it could be quite efficiently predicted when a non-linear model would be preferred with an Area under the ROC Curve score of 0.851 using a RandomForest meta-algorithm. Our results show that one of the best features of the meta-data sets for predicting which type of model is preferred are the landmarkers including our self-made *feature selection landmarkers* and the simple meta-features like the number of numeric features and number of features. To conclude the myth "Non-linear models perform better than linear models" is not true for all non-linear models because the LibSVM non-linear models are outclassed by the linear model on most data sets. Even though some non-linear models do perform better than linear models as is seen by the models created by the Multilayer Perceptron.

In conclusion our method of using a large diverse amount of data sets and using meta-learning for further insight proved to be an efficient way of researching the two myths. For further research the data preparation myth can be researched more extensively by adding more aspects of data preparation for more complete results. For the second myth more linear and non-linear models can be added and lastly more data-mining myths can be further researched. All the used data sets, data sets created for the experiments and algorithms can be found on OpenML with the tag '*study_20*' for further use by anyone.[1]

[1] http://www.openml.org/s/20

Bibliography

[1] A Aizerman, Emmanuel M Braverman, and LI Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

[2] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[3] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[4] Avrim L Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artificial intelligence*, 97(1):245–271, 1997.

[5] Pavel Brazdil, João Gama, and Bob Henery. Characterizing the applicability of classification algorithms using meta-level learning. In *European conference on machine learning*, pages 83–102. Springer, 1994.

[6] Jennifer Carpenter. May the best analyst win. *Science*, 331(6018):698–699, 2011.

[7] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[8] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011.

[9] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[10] Pedro Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87, 2012.

[11] Jerome H Friedman. On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data mining and knowledge discovery*, 1(1):55–77, 1997.

[12] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14):2627–2636, 1998.

[13] Christophe Giraud-Carrier. Metalearning-a tutorial. In *Tutorial at the 7th international conference on machine learning and applications (ICMLA), San Diego, California, USA*, 2008.

[14] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.

[15] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[16] Mark A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

[17] Ross Ihaka and Robert Gentleman. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[18] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[19] Eric Kim. The kernel trick. `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`, 2013.

[20] Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

[21] Kevin Leyton-Brown, Eugene Nudelman, Galen Andrew, Jim McFadden, and Yoav Shoham. A portfolio approach to algorithm selection. In *IJCAI*, volume 1543, page 2003, 2003.

[22] Yonghong Peng, Peter A Flach, Carlos Soares, and Pavel Brazdil. Improved dataset characterisation for meta-learning. In *International Conference on Discovery Science*, pages 141–152. Springer, 2002.

[23] Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In *Proceedings of the 17th international conference on machine learning*, pages 743–750, 2000.

[24] Fábio Pinto, Carlos Soares, and Joao Mendes-Moreira. A framework to decompose and develop metafeatures. In *MetaSel@ ECAI*, pages 32–36, 2014.

[25] Martijn J. Post, Peter van der Putten, and Jan N. van Rijn. Does Feature Selection Improve Classification? A Large Scale Experiment in OpenML. In *Advances in Intelligent Data Analysis XV*. Springer, 2016.

[26] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[27] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531, 2010.

[28] John R Rice. The algorithm selection problem. *Advances in computers*, 15:65–118, 1976.

[29] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961.

[30] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[31] Joaquin Vanschoren, Jan N Van Rijn, Bernd Bischl, and Luis Torgo. Openml: networked science in machine learning. *ACM SIGKDD Explorations Newsletter*, 15(2):49–60, 2014.

[32] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[33] Antanas Verikas and Marija Bacauskiene. Feature selection with neural networks. *Pattern Recognition Letters*, 23(11):1323–1335, 2002.

[34] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[35] David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.