FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Automated Feature Engineering for Classification Problems

Guilherme Felipe do Nascimento Reis

U. PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Mestrado em Engenharia de Software

Supervisor: Carlos Manuel Milheiro de Oliveira Pinto Soares

July 29, 2019

Automated Feature Engineering for Classification Problems

Guilherme Felipe do Nascimento Reis

Mestrado em Engenharia de Software

July 29, 2019

Abstract

The study on feature generation has grown over the last years.It is entirely dependent on domain knowledge and done manually, so it is time consuming and not scalable. In turn, meta-learning helps to learn through different domains and can bring benefits to this area.

We present two meta-learning approaches to support feature selection. The first predicts whether a given automated feature engineering approach will improve the results. The second predicts if a single feature will improve the results.

We test these approaches on 100 data sets from different domains. Our experiments showed that it is possible to use meta-learning in the feature selection process, and can inform us whether or not we should generate features with a the given method.

On the other hand, the results of the second approach, which predicts the usefulness of each individual feature are not positive. The results show that meta-learning can be used to aid the generation and selection of features. However, our approach can still be improved, being more precise in the predictions at the meta-level and better results at the base level.

Our code is available at *https://github.com/guifeliper/automated-feature-engineering*

Keywords: data mining, machine learning, metalearning, KDD, feature engineering, automated feature engineering

Resumo

O estudo sobre geração de features tem aumentado conforme os anos. Totalmente dependente de conhecimento de domínio e feito de forma manual, por isso consome muito tempo e não é escalável. Por sua vez, meta-learning auxilia o aprendizado através diferentes domínios.

Nos apresentamos duas abordagens de meta-learning para auxílio na seleção de features. A primeira prevê se uma determinada abordagem de geração automática de features melhorará os resultados. O segundo prevê se um único recurso melhorará os resultados.

Nós testamos essas abordagens em 100 conjuntos de dados de diferentes domínios. Nossos experimentos mostraram que é possível usar o meta-learning no processo de seleção de recursos, e pode nos informar se devemos ou não gerar recursos com um determinado método.

Por outro lado, os resultados da segunda abordagem, o qual prevê a utilidade de cada característica individual, não são positivos. Os resultados mostram que o meta-learning pode ser usada para auxiliar na geração e seleção de recursos. No entanto, nossa abordagem ainda pode ser melhorada, sendo mais precisa nas previsões no meta-level e melhores resultados no base level.

Nosso código esta disponível em *https://github.com/guifeliper/automated-feature-engineering*.

Keywords: data mining, machine learning, metalearning, KDD, feature engineering, automated feature engineering

Acknowledgements

First, I would like to thank my family and wife for all the support and advice throughout my journey. Definitely without them, I would not be the person I am today.

I also want to thank all the mentors who have appeared in my life, all of them helped me to become the professional that I am today. Mainly, Prof. João Rodrigues, who unknowingly introduced me to engineering when I was still in high school. The Prof. Tiago Henrique, who improved me during my engineering career when I was still in Bachelor, and my Professors Carlos Soares and Rui Quintino who helped and guided me a lot in this work.

For all those who helped me and encouraged me in my path, I thank you from my heart, and I will certainly always remember you.

Guilherme Reis

Contents

List of Figures

List of Tables

Abbreviations

AutoML	Automation Machine Learning
DFS	Deep Feature Synthesis
CRISP-DM	Cross Industry Standard Process for Data Mining
NA	Not Available
NHST	Null Hypothesis Significance Test
ML	Machine Learning

Chapter 1

Introduction

One of the factors for a successful application of a machine learning approach is a well prepared data set. The data needs to be cleaned and prepared for the algorithm to induce a good model. Preparing a data set correctly usually takes a significant amount of time, and it requires solid knowledge about the domain. One particularly important task is feature engineering, which consists of creating new variables that make it easier for the algorithm to obtain a good model. There are two approaches for feature engineering: manual and automated. In the manual approach, one feature is built at a time. This is time-consuming, challenging and error-prone process and depends on domain knowledge. The automated approach can extracts many features from a data set in a systematic way which can be done without human intervention. Additionally, it can be applied to any problem and the building process is quicker and not dependent on domain knowledge.

Several automation strategies have been developed recently, to decrease the effort devoted to featuring engineering. However, these methods commented on in this thesis and our method do not guarantee that the set of features generated will lead to better models [6, 11, 8].

> "Automating feature engineering optimizes the process of building and deploying accurate machine learning models by handling necessary but tedious tasks so data scientists can focus more on other important steps." [6]

1.1 Motivation

On feature engineering, it is possible to automate part of the process which does not depend on domain knowledge.

A subarea of machine learning that deals with the problem of extracting knowledge about learning problems which is independent of the domain are meta-learning. Meta-learning consists of learning about the behaviour of machine learning (ML) algorithms by collecting (meta)data about instances of those processes and applying ML algorithms to that metadata [15]. However, meta-learning approaches have never been applied to the problem of feature engineering.

1.2 Objectives

Given a method to automatically generate features, our goal is to use a meta-learning approach to decide whether it will lead to better models compared to the original data set.

Our preliminary research question is the following:

- Does the systematic combination of features using simple operations lead to better predictive models than the original data set?

After, we specifically address the following questions:

- Can a meta-learning approach predict when a method for the systematic generation of features leads to better predictive models?
- Can a meta-learning approach predict when a method for automatic generation of features leads to better predictive models?
- Can a meta-learning approach predict when an individual feature leads to a better predictive model?

The contributions of this work are:

- Application of a meta-learning approach to the problem of deciding whether a method for automatic feature generation should be applied and which features should be used for improvement of the model;
- Development of a set of meta-features that are specific for the problem addressed here;
- Empirical evaluation of the approach proposed using a simple method for automatic feature generation.

In section 2, we discuss the background and related work; In section 3, we explain the systematic feature generation and the exploratory data analysis. For the sections 4 and 5 we discuss the approaches proposed and the experimental setup used and results. Finally, in section 6, we present the conclusion and future work.

Chapter 2

Background and Related Work

Over the last years, the demands for data mining have been increasing and also the growing interest in automated machine learning, see the figure 2.1 [1]. Data mining uses a considerable number of data analysis techniques for the process of identifying specific patterns in a massive amount of data.

In this chapter, we discuss an analysis of the data mining process and mining techniques 2.1, the actual state of automated machine learning 2.4, meta-learning and automated feature engineering 2.3.

2.1 Data Mining

Data mining can use machine learning techniques, but it is possible to use data management techniques (e.g. relational database management system) . Nevertheless, we are going to talk only about machine learning techniques.

There are two primary type of tasks for data mining, **predictive** and **descriptive**. **predictive** has the intent to predict based on the historical data, and the most common tasks are, classification and regression. *Regression* is used to predict a numeric value. For example, a new car model is added to the database, and the system needs to suggest the price for a car with five doors, electric direction, from a specific brand, with 100.000 Kms. Regression should be used for that case. *Classification* is used to determine a class for a data point. For example, a specific entry for the flowers database does not have a classification, and the system needs to identify what flower it is to complete the data. Classification should be used for that case.

Descriptive data mining is used to organise the data and understand better the information that the database has; it is also divided into two techniques, association and clustering. *Clustering* identifies groups of similar data points. Data points in a cluster are more similar among themselves than to data points in other clusters. *Association* analyses the association between items. For

[1]Available at `https://trends.google.pt/trends/explore?q=AutoML`

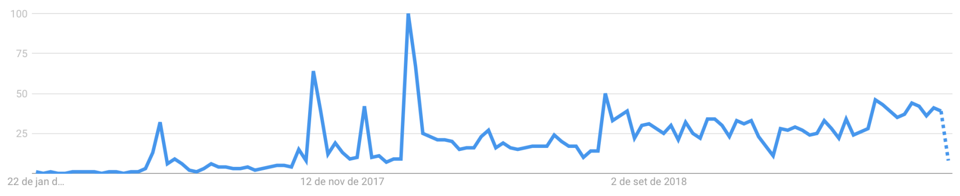

Figure 2.1: Google Trends about AutoML

.

example, in a commercial database association can identify products that are generally purchased together, a famous story analysed a market database and discovered that usually clients who buy diapers also buy beer.

The process of data mining is interactive and challenging. The most common process model involves six phases, where each one has specifics deliveries for the next phase. The process model is also known as Cross Industry Standard Process for Data Mining (CRISP-DM), see life cycle on figure 2.2.

Business understanding - The goal of this phase is understanding the business perspective and then translating it to a data mining problem.

Data understanding - The focus of this phase is to collect and have initial contact with the data. Analyse data quality and identify possible problems.

Data preparation - The goal of this phase is to clean the data, generate more features, select the features created to feed the algorithms for the next phase. This phase is the centre of our work.

Modelling - The algorithms are selected, and their parameters tuned for an optimal result.

Evaluation - Before the final deployment, this step will assess whether the business goals have been met.

Deployment - The goal of this phase is to plan deployment, produce a final report and review the project to evaluate what went right and what went wrong.

All of this information and more detailed process of CRISP-DM can be found in the book CRISP-DM 1.0 [14].

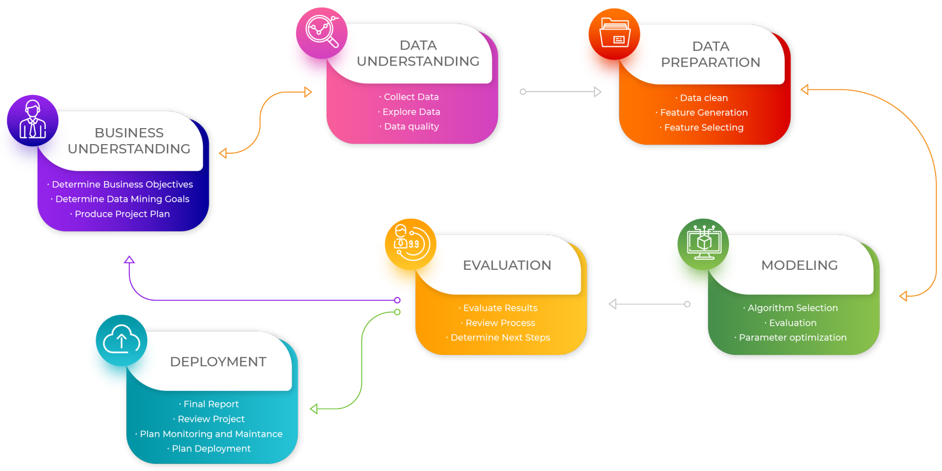

Figure 2.2: Phases of CRISP-DM model [14]

2.2 Automated Machine learning

Machine learning is time-consuming and needs many resources to get the task done. Therefore, companies like IBM have been investing in Automated Machine Learning research [11]. Automated machine learning (AutoML) goal is to change this scenario, developing accessible machine learning.

Currently, AutoML focuses on data preparation and modelling. In data preparation, it is often used to automate the process of data preprocessing, feature generation, and selection. For model training, it is used to automate model selection and hyper-parameter optimisation.

2.2.1 Data preprocessing

The data collected is not always suitable to run a machine learning process. Therefore, it is necessary to transform the data. Preprocessing involves many different tasks such as scaling, missing value imputation, outlier detection, binning and categorical data transformation [2].

Scaling - Scaling techniques ensure that the numerical features are weighted proportionately by the algorithms. The terms used for that technique are standardisation and normalisation.

Missing values - Missing values often occurs in data sets. This can happen because of a device bug, typing errors or even the user choosing not to provide some information.

1. *Use a constant*: It is possible to fill all the missing values with a constant. The constant can be a disproportionate number or a not available (NA) to separate those for the rest of the data set.

2. *Use a mean/median value to fill*: Numerical values can be replaced with mean or median and categorical values can be replaced by the mode.

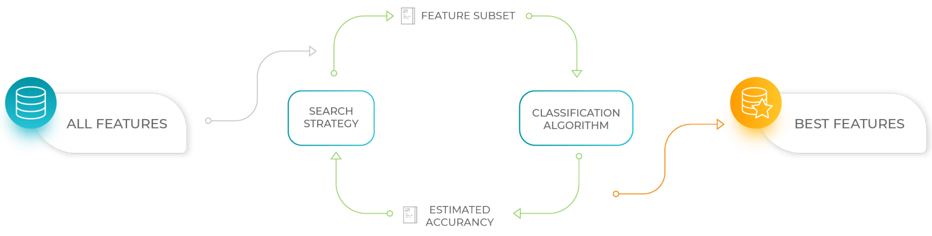

Figure 2.3: Feature selection process [18]

3. *Remove/Delete Data*: This technique is recommended when only a few data are missing, the elimination can be applied for a specific row or column. However, when too much data are missing, it may reduce the amount of available data too much, affecting the quality of the models generated.

4. *Use data mining to predict the missing value*: It is possible to use machine learning algorithms to predict the most probable value for missing data. However, when used for too much missing data it can cause overfitting on the final model because the data used to fill the empty data was the same to train and test.

5. *Add flag*: Indicate the missing data with a flag. Insert a flag (e.g.: 'missing value' string or outlier number) in the missing data can identify where the data was missed, aiding to identify the lack of information and assess the data.

Outlier detection - Outliers are those values "that are so different from the others that distort the data distribution. Those values can lead to low performance models. Outliers have two types, univariate and multivariate. A univariate outlier, as the name implies, is an outlier based on a single variable. A multivariate outlier is a mixture of values on two or more variables.

Binning - Binning goal is to discretizes continuous values, it is a process to group a continuous number in smaller interval, for example, dividing the age of a survey into small intervals. This is useful for algorithms which have difficulties in analysing numerical variables, such as naive Bayes. Two common binning methods are:

- *Equiwidth binning*: The bins are divided into n intervals of equal size.
- *Equifrequency binning*: The bins are divided into n groups, each containing the same number of values.

All of this information and more detail can be found [2].

2.2.2 Feature selection

Feature selection is used to remove unnecessary and redundant attributes from the data because some attributes decrease performance instead of increasing it. Furthermore, the complexity of the

Table 2.1: Example of feature generation extracting information from a date attribute

User ID	Penultimate login	Last Login	Diff Login	Last Login is Weekend?
865	12/12/16	03/01/17	22	FALSE
1954	18/03/16	23/07/16	127	TRUE
1307	25/11/17	11/03/18	106	TRUE
1966	29/03/16	26/04/16	28	FALSE
51	07/01/16	23/02/16	47	FALSE

model can increase with more attributes, not helping the data scientist to understand or explain the model. So, feature selection can help to have a better performance with fewer features and less complexity in the model [4] (Figure 2.3).

Three types of methods can be used: Filter methods, wrapper methods and embedded methods.

Filter Methods These methods analyse each feature independently and assign a score to each. These scores are used to rank the features and the best ones are selected. The scores can be based on the information gain measure.

Wrapper Methods These methods evaluate combinations of features. Different combinations are created and compared by learning and evaluating a model. The performance of the model is used to guide the search for the best set of features. Wrapper methods include recursive feature elimination, deleting and adding features, or using a random algorithm to help the search.

Embedded Methods These are often regularisation mechanisms embedded in the algorithms. It consists of adding a penalty for complex models, helping to reduce overfitting and variance. However, it adds bias to the model. An example of this method is Ridge Regression.

2.2.3 Feature generation

The feature generation, also known as feature construction, adds more features into the raw data and it can result in more information that leads to models with better accuracy. However, adding features increases the dimensionality of the data set and increases the chance of overfitting, known as the curse of dimensionality. To minimise the effect of feature creation, one approach is feature selection, which was described earlier.

To illustrate the feature engineering dilemma, consider Table 2.1. Here we have five columns showing the behaviour of access of some users, and the first three columns are the raw data, the last two columns are the new features generated in the process of feature engineering. We can see that the users who access the platform on weekends also stay longer without using it. When studying only the original data, we do not see this patterns. New features can help to show those patterns and help to build better models.

Numerical - Many mathematical operations can be applied to transform single and groups of variables. For example, log, ratio, addition, variance of two numbers or any mathematical operation. Pairwise feature creation is the name given for the process of numerical computing attributes.

Temporal - The temporal features can have two most common transformations, extraction of partial information (e.g. day, month and weekday) and combination of values (e.g. Difference between two dates). Temporal features are complex data types, because of that the combination of different elements aids to extraction. In the table 2.1, it is possible to see an example of temporal transformation, the column "Last login is weekend" is a transformation from the column "Last login", other feature would be the difference of the columns "Penultimate login" and "Last login".

Categorical - This transformation has fewer possibilities than others. Usually, categorical values are represented by a string, frequency of value or intervals is commonly used in categorical data transformation. For example, the age interval, from a user data set containing the user's birthday, it is possible to create some intervals from the user's birthday, e.g. 18-25, 26-32, 33-45.

2.2.4 Machine learning algorithms

There are many algorithms for the most common prediction tasks, such as classification and regression. Many of those algorithms apply to both tasks, with minor changes. For example, the Support-vector machine and K-Nearest Neighbors algorithms can be applied to classification or regression problems. However, the description of these algorithms is beyond the goals of this thesis [2].

However, selecting an algorithm should not be a personal choice. Different algorithms lead to different models, which may have very different performance. Thus, this choice may result in unsatisfactory results. Additionally, algorithm selection should also consider computational complexity, differences in training and scoring time, linearity versus non-linearity, algorithm-specific feature transformations. AutoML approaches aim to take these issues into account, saving the data scientist time and avoiding poor choices.

2.3 Meta-Learning

Meta-learning extends the concept of data mining, adding methods to extract knowledge about the behaviour of data mining algorithms. Meta-learning has been used for the automation of machine learning. It is based on new concepts, meta-features and meta-knowledge. It aims to learn not only how to solve a given problem, but learn the structure of the corresponding task [16]. For example, in a process of classifying if a patient has a determined disease or not, instead of only learning how to classify that disease, meta-learning will learn about the process of learning a model for that disease so it can later adapt it for other diseases. This technique is well known for learning to learn.

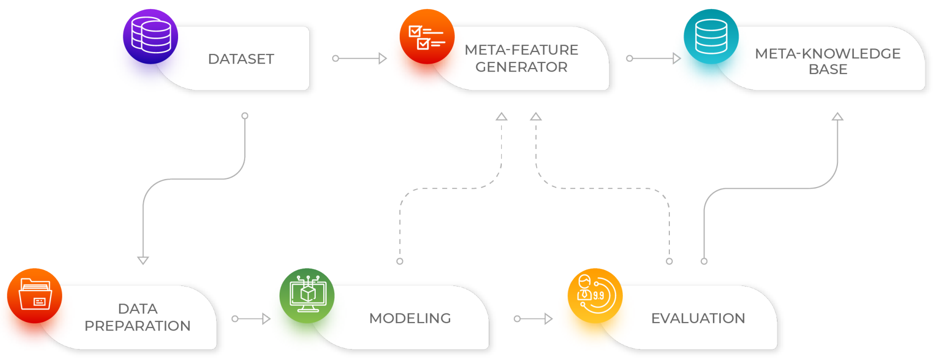

Figure 2.4: Meta-Learning [17]

We can see a clear difference between base-learning and meta-learning [13]. In the case of meta-learning, the goal is to accumulate experience from many different applications from the same nature. Base-level learning on other hand is only interested in learn how to execute the specific task.

For meta-learning approaches, there is a need to use meta-features. Meta-features are the characteristics of the data sets. Many of those characteristics are statistical and information-theoretic parameters. They include the number of classes, number of features, correlation between features, number of examples and others [16, 17]. There are also others data set characterisations such as model-based and landmarking [15].

To illustrate the meta-learning process see the figure 2.4. Following the workflow, from the new data set we generate meta-features. In parallel, we apply the standard workflow for base-learning, including data preparation, model selection and evaluation. All the information generated is stored in the meta-database so that we can use in the following learning process [17]. The meta-model is learned by applying a learning algorithm on the meta-data. Given a new data set, the meta-model is applied to support the selection of the best algorithm for that problem.

By obtaining the predictions from the Meta-level, we can use this result to apply it to a data set. The prediction obtained by the meta-model can be evaluated at the meta-level and base-level. At the meta-level, the evaluation compares the prediction with the label in the meta-data. For instance, if a classification approach is used to select the best algorithm for a data set, the meta-level accuracy is the proportion of times the meta-model predicts the best algorithm. At the base-level, the evaluation compares the effect of following the recommendation provided by the meta-model. Given the same scenario as above, the base-level evaluation estimates the loss in base-level performance of executing the algorithm predicted by the meta-model when compared to executing the best algorithm. If the meta-model recommends the best algorithm correctly, then there is no effect. However, if the algorithm predicted is not the best, there will be a loss in the performance obtained which may or may not be large. To illustrate this, see figure 2.5.

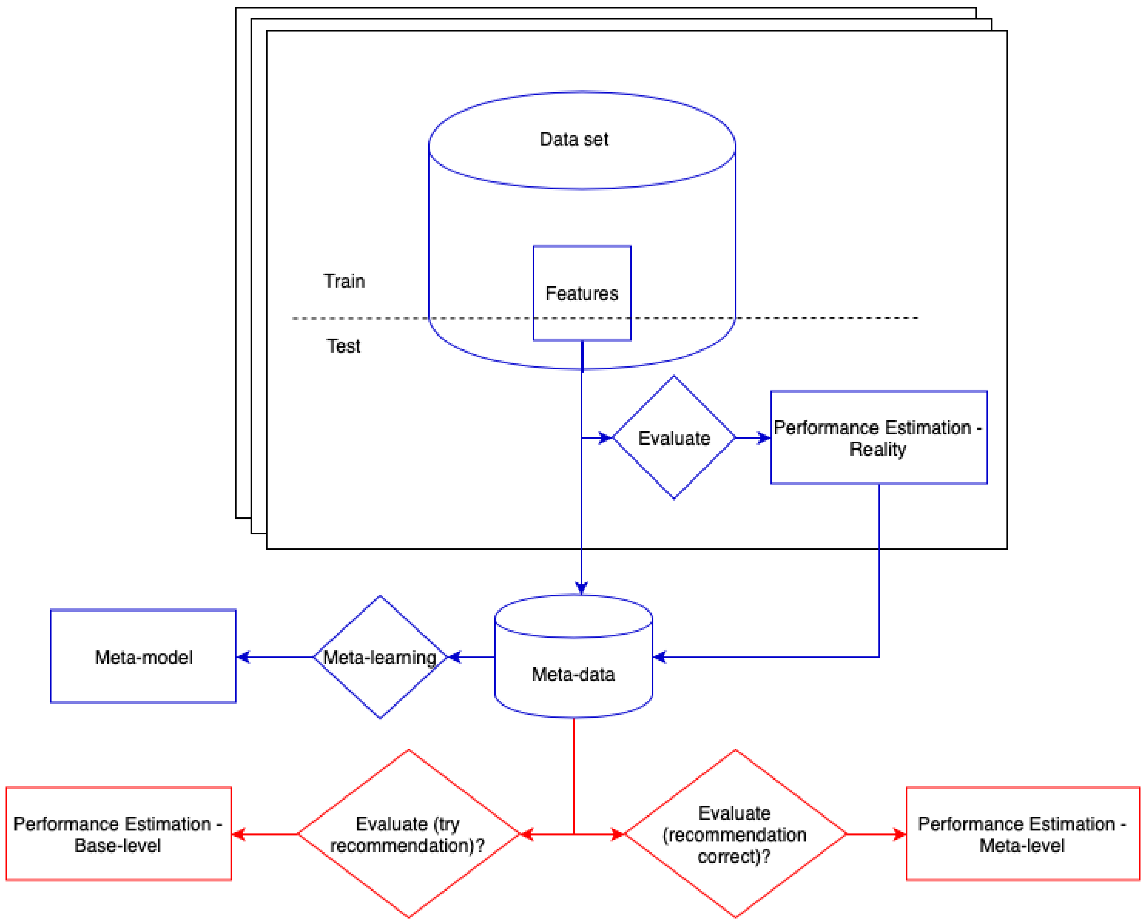
Data set
Train
Features
Test
Evaluate
Performance Estimation - Reality
Meta-model
Meta-learning
Meta-data
Performance Estimation - Base-level
Evaluate (try recommendation)?
Evaluate (recommendation correct)?
Performance Estimation - Meta-level

Figure 2.5: Meta-Level and Base-Level diagram

2.3.1 Evaluation

Given that the accuracy values are stochastic, it is necessary to assess whether differences between different methods are statistically significant. For that purpose, we can use multiple tests. A recently proposed test is the Bayesian Correlated t-test [1]. The authors discourage the use of null hypothesis significance tests (NHST) in to compare performance between ML algorithms. Because it does not estimate probabilities of hypotheses, ignores magnitude and uncertainty and others, they suggest Bayesian analysis instead of NHST.

Additionally, Bayesian analysis can show the probability of two algorithms being practically equivalent. According to the authors, it is reasonable to affirm that a difference of less than 1% between two classifiers can be considered as practically equivalent. Therefore the interval of [-0.01, 0.01] is defined as a rope or region of practical equivalence [1].

2.4 Automated feature engineering

.

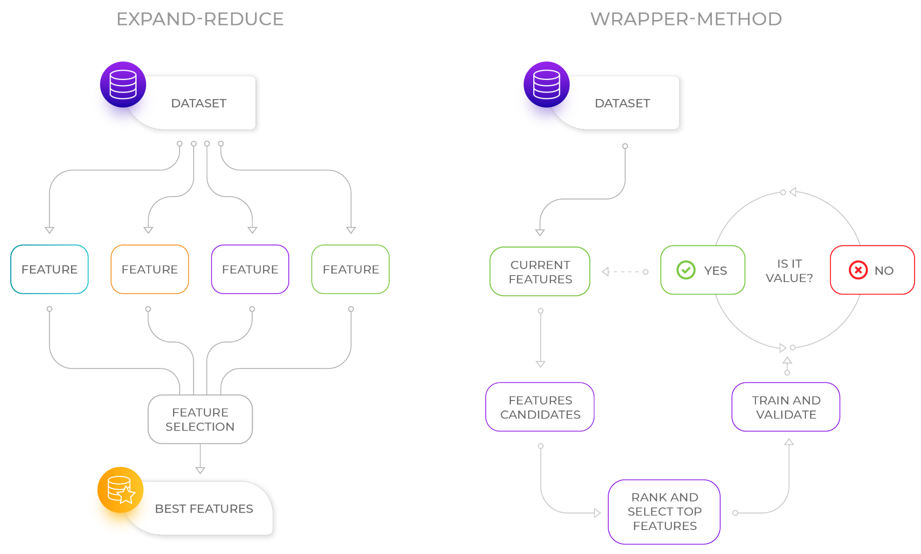

Figure 2.6: Expand-reduce and Wrapper method workflow

In this section, we discuss the related work on the topic of automated feature engineering, which was introduced in Section 2.2.3. There are not many relevant papers. We discuss them according to different characteristics: Approaches, Feature transformations, Evaluation, Tasks, Data, Algorithms, Results analysis, Programming languages.

2.4.1 Approaches

Automated Feature Engineering has two approaches to feature extraction, called expand-reduce and wrapper-method. The expand-reduce is the most common method used for feature engineering as we can see the Deep Feature Synthesis [6], ExploreKit [8], AutoLearn [9], One button machine [11] using this approach. The **expand-reduce** approach tend to create a very large number of features in minimal time. Resulting in the curse of dimensionality which may lead to overfitting, as explained earlier 2.2.3. Thus, these approaches require some kind of feature selection (Section 2.2.2). Depending on the technique used, this may incur into significant computational costs.

The Cognito [10] and the Automatic Feature Generation for Machine Learning Based Optimising Compilation [12] uses another approach called **wrapper-method**. This approach is based on creating a feature at a time. After creating a feature, a process is applied to evaluate and decide if the new feature adds any useful information to the data set. The approach will run until a time defined runs out. The advantage is the curse of dimensionality does not happen in this approach, so it is no need for feature selection after the process. However, the process can be slower than expand-reduce. Both methods are illustrated in Figure 2.6.

Table 2.2: Example of binary feature generation

ID	**Quantity**	**ProductPrice**	**Total Price**
1	3	123	**369**
2	9	10	**90**
3	33	15	**495**
4	12	324	**3888**
...	...	...	**...**

The meta-learning is not well defined in any paper, only in Deep Feature Synthesis [6] and ExploreKit [8] speaks about meta-data or meta-features.

2.4.2 Feature transformations

Transformations are divided into three main operators: *unary, binary and higher order*. The following paragraphs will introduce the main types of each one of them.

Unary operators discretise and normalise one feature. *Discretizers* are good for some classification algorithms which were not developed for numerical data, such as naive bayes. Table 2.1 illustrates well what is a discretise example. The last column *"Last Login is Weekend?"* is a popular discretisation of dates, categorising a date feature in only two options, true or false. The discretization operation has been used in many automated feature engineering approaches [6, 11, 10, 8, 9].

Normalisation is a well-known operator. The normalisation tries to approximate the distribution of the variable to a normal, with mean 0 and standard deviation 1. In addition, we are also considering a unary transformation those functions that need a unique input as *logarithm; square; square root; exponential; rounding; absolute* value as Cognito [10] mention on its paper.

Binary operators can be applied to pairs of features. These include basic arithmetic operations such as +,- ,× , ÷, comparison operations such as ==, !=, >,<, >=,<=, and logical operations such as AND, OR.

A binary operator is illustrated in Table 2.2. The simple arithmetic operation multiplication is applied to two features, Quantity × ProductPrice. We note that all papers analysed use binary operators [6, 11, 10, 8, 9, 12].

Higher order operators can use multiple features for the extraction of new features. Operators that are commonly used include [6, 11, 10, 8, 12]: *SUM; MIN; MAX; COUNT; Distinct count; GroupByThenMax; GroupByThenMin; GroupByThenAvg; GroupByThenStdev; GroupByThenCount; TimeWindowEntityAgragateByEntity; TimeWindowPeriodAgragateByEntity; AggregatebySpatial; AggragatebySpatialTime.*

For example, given a data set with numeric values for each quarter, we could use a transformation for sum all the quarters, e.g. SUM(Q1, Q2, Q3, Q4). Furthermore, we can notice that SQL functions might easily manage most of those functions.

2.4.3 Evaluation

Evaluation is important to understand the behaviour of algorithms. For the analysed projects, the measure used was classification accuracy [6, 12, 11, 8, 10, 9]. The accuracy is simple to calculate; it is the ratio of the correct predictions to the total number of input samples. A important issue needs to be considered when using accuracy on classification problems. This evaluation method can be misleading if the data set is not well balanced. In some cases the data set can have 90% of class A and 10% of class B, being harder to classify B than A, and the problem is that an accuracy which would be very good in balanced data sets (e.g. 80%) is very bad in this example. To address this issue, in Explore Kit [8] the percentage of error reduction is also analysed.

2.4.4 Data

Different data sets from different sources have been used in the related work. Those sources are:

UCI Repository - David Aha started the UCI repository in 1987. It is commonly used for students, educators and researchers. Cognito [10] and AutoLearn [9] use UCI data sets.

Kaggle - This is a data science community, owned by Google that is known for the competition to solve challenges. Deep Feature Synthesis [6] and One button machine [11] used the KDD cup 2014 challenge to evaluate their approach.

OpenML - A platform for open science in ML, it contains a large number of data sets. Data sets from that platform was used by ExploreKit [8].

Others - Some papers used specific data sets, including company and compiler data sets. The One button machine [11] used Outbrain and Bimbo group company's data set. Automatic Feature Generation for Machine Learning-Based Optimising Compilation [12] used data concerning GNU Compiler Collection.

All the related work uses supervised learning [6, 11, 10, 8, 9, 12].

2.4.5 Algorithm

There are several algorithms today for being used, however, we notice a preference between Random forest and Decision tree on all projects studied [6] [11] [10] [8] [9] [12]. Decision trees are relevant when it is necessary to explain a particular decision. The decision tree is classic for visual explanation and easy to understand; it solves classification and regression problems. For example, in the figure 2.7, we demonstrate the result of the algorithm decision tree run in a data set, which contains only the size of an array as a feature and the model classifies the item between two algorithms heap sort and merge sort. For Decision tree the node means a feature, the branches show a decision and the leaves are the results [2].

Random forests are also a popular method used to solve classification and regressions problems. The model of Random Forest creates an entire forest of random and uncorrelated decision

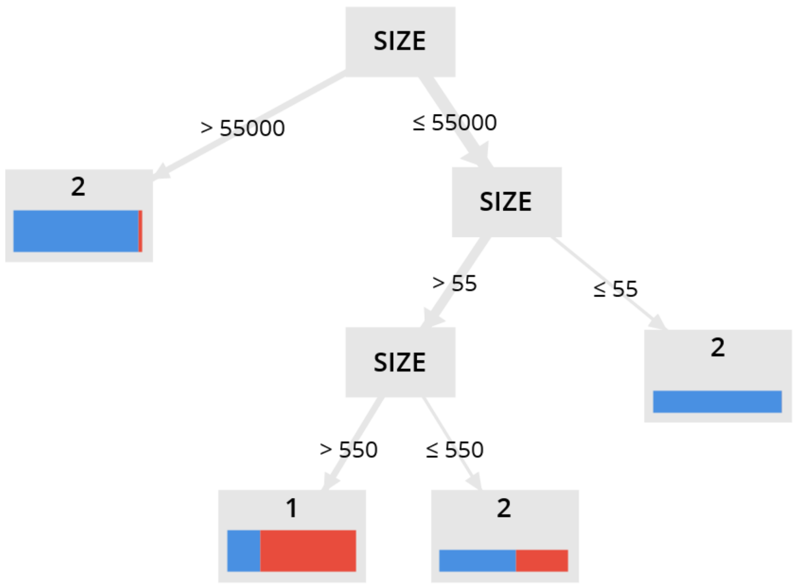

Figure 2.7: Example of Decision tree aids the decision between two algorithms

trees to present a better predictive result. The Decision tree in most of the cases have a high variance, and it happens when we extract different training and test sets from the same data set, resulting in different results.

Sadly, the Decision tree can cause poor performance for a new data set. However, Random forest is eager to solve the weakness of Decision tree reducing the variance and bias errors, by choosing a subsample of the feature space for each split. [3]

2.4.6 Results analysis

In this section, we summarise the results of the related papers. They are discussed separately because they are not all relatable.

ExploreKIT [8] - The authors built four approaches and the best approach was the ML_full, obtaining an average error reduction of 17.4%–29.3%.

Deep Feature Synthesis [6] - The Data Science Machine, also known as Deep Feature Synthesis, was one of the first projects with successful results on automated feature generation. The algorithm has been tested in Kaggle competitions and it achieved 90% of the best score achieved by any competitor in two out of three competitions. The best performance was on KDD15 challenge beating 86% of other competitors.

OneButton Machine [11] - The One Button Machine uses two approaches. It was tested in different databases, one of them was the KDD 2014 cup for the comparison with the Deep feature synthesis project. The One button machine compare their results with the results of data scientists in the KDD 14 Challenge, and their results are in the top 17% while the DFS was in the top 30%. The results show that these approaches are competitive with a large number of experienced data scientists.

AutoLearn [9] - The Autolearn algorithm aims to use fewer features for better accuracy. It achieved 13.28% and 5.87% improvement in terms of accuracy, against the original features. It also obtained competitive results when compared with ExploreKit and others. The algorithm was tested on 25 data sets.

Cognito [10]- It obtained improvements between 7% - 40% compared with the original features on different data sets.

Automatic feature generation for machine learning based optimising compilation [12] - The goal of this project is to use AutoML for detecting the best loop unrolling in GCC 4.3.1. The novel algorithm applied in this project was able to achieve 75% of the maximum velocity of the compiler on average. The benchmarks show that the algorithm improved 35% of the speed.

In this thesis will be used one of the Bayesian tests presented in the article, called Bayesian correlated t-test that will be used to compare performance between the results obtained by 10-fold cross-validation and the baseline of each fold.

2.4.7 Programming languages

The programming languages used in the related work were Python, Java and C/C++. There is no motivation for the selection of the programming language. Thus, the discussion aboutthe better language for Automated Machine Learning is open.

The choice of programming language is not relevant for this thesis. However, the choices in the related work are well divided. Two papers use Python, Deep Feature Synthesis [6] and AutoLearn [9], two papers use Java, Cognito [10] and ExploreKit [8], and the only one that uses C/C++ is Automatic Feature Generation for Machine Learning Based Optimizing Compilation [12]. There is another one which that did not identify which programming language was used [11].

The Vision Mobile company released a report in April 2017[2], showing the popularity of each language for machine learning. However, it can change depending on the background of the individual in charge of the development process. For example, Python was the prefered language for machine learning on 38% of answers and used mostly by data scientists and junior professionals. Front-end developers use javascript with 16%, Electronics engineers prefer C/C++ (8%), data analysts and statistician prioritise R (14%). In conclusion, there is no correct language.

[2]Available at https://visionmobile.com/reports/state-developer-nation-q1-2017

Chapter 3

Systematic feature generation

In this chapter, we present our approach for systematic feature engineering. This method will be used as feature generation method for the meta-learning approaches used in the following chapters. We also present an exploratory data analysis of the empirical results obtained with this method. The approach was implemented in Python[1], using some libraries to assist the development, such as Itertools [2], Scikit Learn [3] and Pandas [4]. The ML algorithm used was Random Forest with 100 estimators and all default Scikit-learn parameters . The evaluation metric used was accuracy and it was estimated with stratified cross-validation.

3.1 Experimental Setup

We analysed 100 supervised classification data sets collected from the OpenML platform [5]. We selected data sets with 30 or fewer classes (Figures 3.1 and 3.2). Since we have decided to focus on operations on numerical variables, we decided to remove all variables which are not numerical 3.2. For example, considering we have a data set with 20 attributes on the total, but only 5 of them are numerical attributes, our original data set will contain only the 5 numerical attributes. The data sets were collected from the OpenML platform [5]. The full list of data sets can be found in appendix A.

3.2 Approach

Our approach can be applied to variables of any nature. However, for simplicity, in this project, we will focus on numerical variables, without loss of generality.

[1] `https://www.python.org/`
[2] `https://docs.python.org/3/library/itertools.html`
[3] `https://scikit-learn.org/stable/` - version 0.19.2
[4] `https://pandas.pydata.org/`

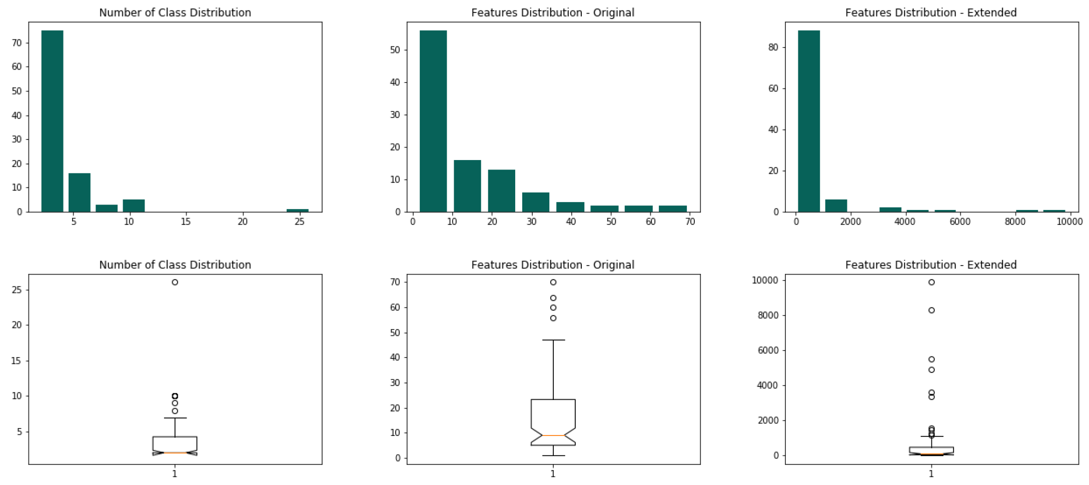

Figure 3.1: Analysis of data sets without eliminating highly correlated features.

Formally, given a data set $D := (X,Y)$, with a target variable Y, and a set of variables $X := (x_1, x_2, \dots, x_m) \subseteq \mathbb{R}^m$, where m is the number of variables. Additionally, given a binary function $\tau_o : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ our approach generates a set of new features E_x containing all possible combinations of the original variables, without loss of generality, using that function:

$$E_x = \left\{ f_{(o,i,j)} | f_{(o,i,j)} = \tau_o(x_i, x_j), \forall_{i \neq j \in \{1,\dots,m\},o} \right\} \tag{3.1}$$

After the generation of new features, the approach combine the new features to the original features, $X^+ = (X, E_x)$, then for each original data set we have $D^+ = (X^+, Y)$. This approach is expected to generate a very large number of variables, many of which may be redundant. For instance, if the function used to combine variables is commutative (e.g. product), then $E_x^{o,i,j} = r_o(x_i, x_j) = r_o(x_j, x_i) = E_x^{o,i,j}$, which means that these two new features are redundant. Thus, we apply a filter feature selection method to eliminate redundant features in E_x [7]:

$$X^{x,filt} = filter(X^+) \tag{3.2}$$

3.3 Exploratory data analysis

Figure 3.1 shows the distribution of the number of features in the data sets, before and after the addition of new features. The first plot shows the distribution of classes, the majority of the data sets has between 2 and 5 classes, but there is an outlier, counting 26 classes. In the same figure, it is possible to observe the difference between the features on the original and the extended data sets.

The extraction of features does not take too much time. The algorithm to create new features and evaluate the original and the new data set for the 100 data sets, took 4 hours. The process generated 132.206 features, which would be impossible for a human to perform in such a short

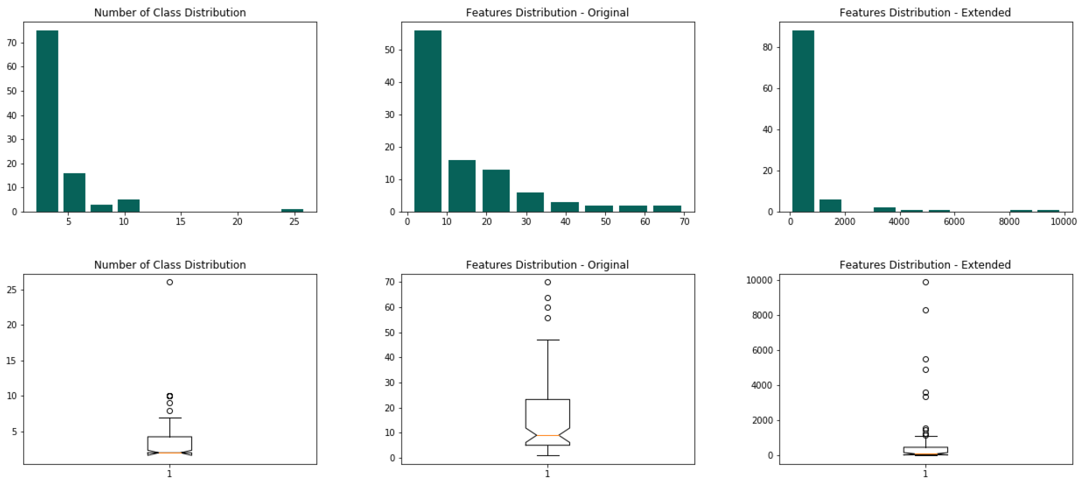

Figure 3.2: Analysis of the data sets with highly correlated features removed.

time. However, a large number of new features is expected. For each binary function, in a data set with *m* variables, a combination of *m*, two-by-two, will be generated. For example, the original hill-valley data set has 100 numeric features. The extension data set we generated has 19,900 features in total.

Figure 3.2 shows the distribution of the number of features in the data sets, before and after the extension after applying a filter-based feature selection method, as described in Section 2.2.2. In this case, the original hill-valley data set has 100 numeric features, and after the extension and filtering process, it results in 9902 features on the total. The elimination of highly correlated features can reduce the computational cost of our model.

3.3.1 Base-level performance

We analysed the base-level performance on the two versions of the extended data sets, with and without feature selection. The data was classified with two options 1 and 0, which 1 is when the systematic feature generation approaches have a better performance than the original data set and 0 when the original data set has a better performance than the extended data set. For all 100 data sets analysed we have a balanced results on both versions. For the filtered strategy, we see 52 improvements from the original data set (class 1) and 48, which the original is better than the extended features (class 0). For the non-filtered strategy, we see 51 improvements from the original data set (class 1) and 49, which the original is better than the extended features (class 0). We see a slight difference between them. More details are available in Appendix A.

However, we can see a more significant distinction in the difference between the accuracy of the original data set and extended data set. In the filtered data set, we have a maximum improvement on the accuracy of 48.5 percentage points and a maximum reduction of 9 points. For the non-filtered strategy, we had a maximum improvement on the accuracy of 40.5 points and a maximum reduction of 15.5 points.

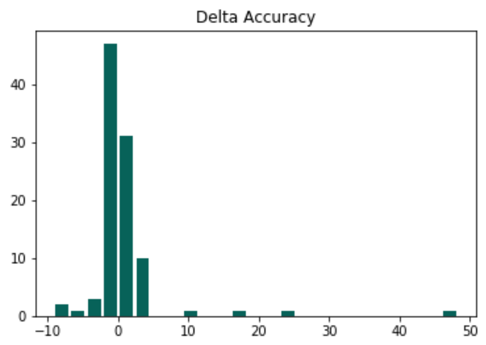

Figure 3.3: Exploratory analysis of the delta between the original accuracy vs extended accuracy.

Analysing the average for both strategies considering the 100 data sets, the systematic feature generation results in an overall improvement of 0.97 points for the filtered strategy and 0.68 points of improvement for the non-filtered strategy. The third quartile shows that we obtain gains of 0.83 and 0.57 percentage points in the on the filtered and non-filtered strategies, respectively.

In spite of this, there are many data sets in which the extension does not lead to improvements, which means that, given a data set, it is necessary to decide whether it is worthwhile to generate new features or not. We address this problem with meta-learning approaches in the following chapters. The filtered extended data set leads us to conclude that filtering high correlated features lead to a better model. Also, we can conclude that the systematic combination of features using simple operations can lead us to better predictive models.

Chapter 4

Meta-learning to select complete sets of features

In this chapter, we present our meta-learning approach to select complete sets of features. In the previous chapter (Chapter 3), we observed that many data sets after the extension did not lead to a better model, leading us to a new problem: *Can a meta-learning approach predict when a method for automatic generation of features leads to better predictive models?*

4.1 Problem definition

The problem is to anticipate whether the generation of automatic features will generate any gain in the performance of the model or not. Besides aiming to improve accuracy, we can reduce the computational resources required for learning the models.

4.2 Approach

We are going to address the problem using a meta-learning approach, but the typical meta-learning approaches, meta-features characterise a single data set. In our case, we have two data sets: the original and the extended data sets. Most importantly, we expect that it is the combined ratio in the meta-features between the original and extended meta-data that determines whether the algorithm will be able to obtain a better model on the extended one or not.

Given the sets of data sets $\{D_1, D_2, \ldots, D_m\}$, the sets of combination between original data and new features $\left\{X_1^+, X_2^+, \ldots, X_m^+\right\}$, the sets of extended data sets $\left\{D_1^+, D_2^+, \ldots, D_m^+\right\}$ and for each p a function mf_i that computes a given meta-feature (e.g., average correlation), we apply it to the original data set, $X_{i,j}^X = mf_j(D_i)\forall_{i,j}$, to the extended one, $X_{i,j}^{X+} = mf_j(D_i^+)\forall_{i,j}$ and then we combine using the function, $X_{i,j}^{(X,X^+)} = \mathscr{T}(X_{i,j}^X, X_{i,j}^{X^+})$. The function used to combine the characteristics of the original and derived data sets should quantify the differences between them, so a ratio or a subtraction are suitable functions, and in this thesis, we choose the ratio.

For all data sets, all relevant information was recorded, making it possible to compare the original and the extended data sets, both in terms of their characteristics as well as in terms of the performance of the algorithm. The meta-level class attribute holds the final results: 1 if it is worthwhile to do the automatic generation of features or 0 if the algorithm should be applied to the original data set. We want to apply the meta-learning approach to predict when a method for the systematic generation of features leads to better predictive models.

4.3 Experimental setup

For the generation of meta-features, we used the Meta-Feature Extractor (MFE) [15] R package. It provides five distinct groups of meta-features: general, statistical, information-theoretic, model-based, landmarking. However, we do not use all the groups or variables for this experiment, and we have some restrictions in the selected groups.

The domain of information-theoretic group is only for categorical data [15], and in our experiment, we used only numerical data. This lead us to eliminate the whole group. The Landmarking group measures the performance of a decision tree for each data set, in our case it leads to a significant computational cost, and, thus, we decided to eliminate these meta-features as well for this project. Nevertheless, we believe they may be useful and should be considered in future extensions of this work.

Additionally, as we have a reasonable variety of data sets, some meta-features generated invalid results. On that case, meta-features with more than 10% of not available or not applicable were also eliminated, namely: *general.numtocat, statistical.sdratio, statistical.gmean.sd, statistical.gmean.mean, statistical.wlambda, statistical.can cor.sd, statistical.skewness.sd, statistical.nrcorattr, statistical.kurtosis.sd, statistical.cor.sd, statistical.cor.mean, statistical.skewness.mean, statistical.kurtosis.mean, statistical.hmean.sd*. Most of these are based on standard deviation or mean and were caused by divisions by zero, as expected [15].

Also, in order to remove noise, we applied a drop of highly correlated features, performed with a threshold of 95% for meta-features of the original data set. For each data preparation operations, we performed two hypothesis, the first was performed for the base data set without filtering at the base level, and the second hypothesis we added the filter mentioned above, at the base level data set.

4.4 Results

Our meta-learning approach can predict from the metadata of the original data set and extended features, whether using the new features produces a better model, as we can see from the model

Table 4.1: Performance achieved by applying the meta-learning approach.

	No filter		**Filtered**	
	Average (%)	sd (+/-)	Average (%)	sd (+/-)
Base	60.94	(0.21)	61.93	(0.30)
Base - add ratio	62.14	(0.19)	56.38	(0.31)
Base - drop NA	60.83	(0.23)	62.91	(0.33)
Base - drop correlations	58.85	(0.21)	60.04	(0.34)
Base - add ratio & drop NA	60.92	(0.19)	62.18	(0.33)
Base - add ratio & drop correlations	55.05	(0.24)	61.09	(0.37)
Base - add ratio & drop NA & drop correlations	57.14	(0.25)	61.20	(0.37)

performance in table 4.1. Recalling that our baseline is 52%, and we are using 10-fold cross-validation, our approach obtained an accuracy of 62.91% (+/- 0.33) when the data set to drop the highly correlated features.

For the table 4.1 we can consider the base data as the original $X_{i,j}^{X}$ and the extended $X_{i,j}^{X+}$ metadata on the same data set. From this base meta-data, we performed different data preparation operations, remove all not available generated by the MFE package and remove highly correlated features, and most importantly we added the combination ratio as the equation $X_{i,j}^{(X,X^{+})}$. We applied these preparation operations to the two variants of the meta-datasets filter and without the filter.

However, this process still needs the generation of features to compute the meta-features. This leads to additional computational cost. Even if this is significantly smaller than the cost of learning the models, we need to investigate if successful recommendations can be obtained without the generation of the new features. In other words, if the meta-learning process uses only the metadata of the original data set to predict whether or not to add new features.

To predict when or not to add the new features, we apply several approaches to the original data set, to be able to remove any noise that impacts the performance of the model. Therefore, considering the original data set, we extracted all meta-data as we saw on the previous section $X_{i,j}^{X} = mf_j(D_i)\forall_{i,j}$, including only the general, statistical and model-based groups of meta-features.

Results in table 4.2 show that it is possible, by using only the meta-data of the original data set to predict when the method for automatic generation of features leads to better predictive models

Table 4.2: Performance achieved using only the metadata of the original data set.

	No filter		**Filtered**	
	Average (%)	sd (+/-)	Average (%)	sd (+/-)
Base	63.14	(0.26)	59.04	(0.27)
Base - drop NA	58.94	(0.23)	61.11	(0.33)
Base - drop correlations	66.96	(0.25)	62.95	(0.24)
Base - drop NA and correlations	65.05	(0.25)	61.31	(0.31)

Table 4.3: Loss impact for the proposed approaches.

	Average accuracy loss Non-filtered approach	**Average accuracy loss Filtered approach**
Our approach	1.11	0.91
Baseline	0.54	0.55
Upper bound	1.76	2.07

without applying it. We come to this conclusion by comparing the results of the meta-learning approach with the majority class. There are 51 cases for the unfiltered data set and 52 cases for the filtered data set in which the addition of features improve the model. Thus, considering that we have 100 data sets, the majority class baseline is 51% and 52% for the unfiltered and filtered, respectively. Our best result improved by 15 percentage points on the majority class, achieving 66.96%". Based on this successful result we conclude that the meta-learning approach can predict whether to use automatic feature generation, even without computing the new features.

Using the correlated Bayesian *t*-test it shows us the probability for each side of the rope, considering applying the rope min and max as +/- 0.05 [1]. The results show the comparison between our approach versus the baseline, revealing that our approach statistically can be better than our baseline in 88.09% of the cases, the baseline will be better in 11.70% of the cases, and a draw happens on 0.19%. This confirms that the meta-learning approach proposed can predict whether the features generated systematically will lead to more accurate models.

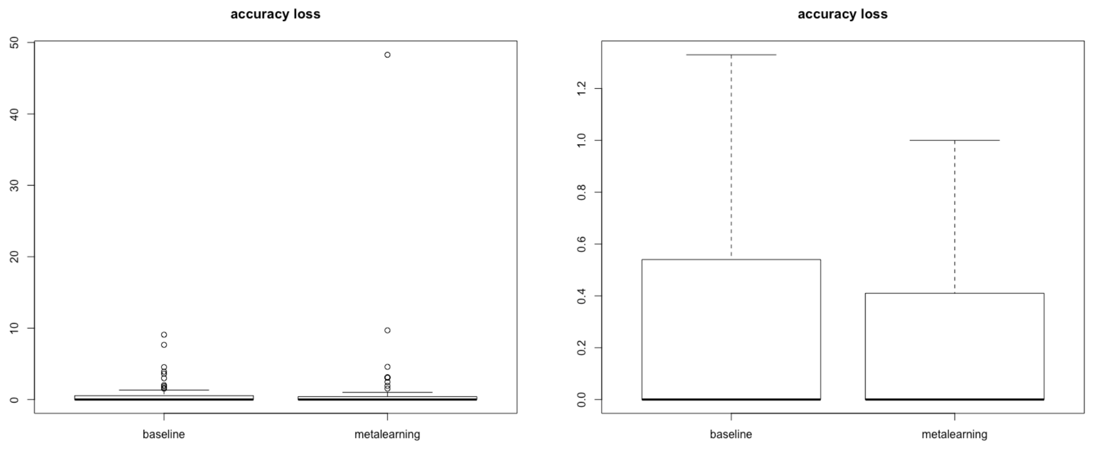

Figure 4.1: Accuracy loss between base-learning and meta-learning.

4.4.1 Base level

It is also essential to analyse the negative impact of our approach at the base level. In other words, if we miss a prediction, how many percentage points we lose in the accuracy of the base-level models. For this, we can analyse Table 4.3, which presents the average loss of our approach.

In the table 4.3, it is possible to see our approach, the baseline and the upper bound. The baseline is calculated from the most frequent class, that is, the average loss if we choose the most frequent class. This baseline method has a loss close to 0.54 percentage points. Our approach has an approximate average loss of 1.0 percentage point and the upper bound rate of 1.91 percentage points.

Although both approaches generate positive results, the meta-learning approach has higher loss than the baseline. This observation is not aligned with the meta-level evaluation. However, according to the boxplot in figure 4.1, it is possible to assume that this result is due to a single outlier. The Hill-valley data set had a base level gain of 48.28 percentage points, after our approach. When predicting if the new features could generate a better performance to the model, the meta-learning approach made an incorrect prediction. The generated model could not predict that generating new features for the Hill-valley data set would generate a better model than with the original data set.

Disregarding this significant loss generated by the Hill-valley data set, we can verify that our average loss is 0.42 percentage points. We believe that our approach needs a more meaningful variety of data sets to compose its metadata, mitigating some outliers as the Hill-valley.

Chapter 5

Meta-learning to select individual features

In the previous chapter (Chapter 4), we showed that it is possible to adapt a meta-learning approach to predict when a method for automatic generation of features leads to better predictive models. Here, we address a more complex problem: *Can a meta-learning approach predict when a feature leads to better predictive models?*"

5.1 Problem definition

The generation of features using the expand-reduce method generates a large number of features. Unfortunately, not all features increase the performance of the model. It is necessary to perform feature selection to avoid noise in the data set or even a loss of performance in the generated model. Feature selection is applied after feature generation. We saw that it is possible to predict when a method for automatic generation of features leads to better predictive models. To further help in this process, we will use the meta-learning approach to predict whether an individual feature will bring some performance improvement to the model.

Use meta-learning for an individual feature is a very complex problem, so we are going to address a simpler one: given a filtered data set, can the meta-learning approach predict whether an individual feature leads to a better predictive model.

5.2 Approach

Our approach goal is to predict when an individual feature leads to a better predictive model. However, this is a complex problem, because meta-learning is typically applied in a data set and not in an individual feature. In this case, we simplified the problem using some meta-features that can be applied for an individual feature.

Given the sets of data sets $\{D_1, D_2, \ldots, D_m\}$, the sets of combination between original data and new features filtered $\left\{X_1^{x,filt}, X_2^{x,filt}, \ldots, X_m^{x,filt}\right\}$, then for each extended set of features we have $D^{filt} = (X^{x,filt}, Y)$. Thus we have the sets of filtered extended data sets $\left\{D_1^{filt}, D_2^{filt}, \ldots, D_m^{filt}\right\}$ and for each x derived feature, a function mf_i that computes a given meta-feature $X_{i,j}^{x} = mf_j(x_i)\forall_{i,j,x}$. We extract the meta-features for each feature x when the feature is not in the data set D^{filt} , so we have $X_{i,j}^{X^{ext-x}} = mf_j(D_i^{filt-x})\forall_{i,j,x}$. After all we combine the using the function, $X_{i,j}^{(x,X^{x,filt})} = \mathscr{T}(X_{i,j}^{x}, X_{i,j}^{X^{x,filt}})$. The function used to combine the characteristics of the meta-data $X_{i,j}^{x}$ and $X_{i,j}^{X^{ext-x}}$ should quantify the differences between them, so a ratio or a subtraction are suitable functions. In this thesis, we combined by ratio the characteristics of the x and $ext - x$ to get our final meta-data.

The traditional meta-learning approaches, meta-features characterise a single data set. In our case, we have a single feature and we want to decide whether a single feature is useful or not. In this project we followed a simple approach, where we selected traditional meta-features that can be used for that purpose, meta-features which fits in the following characteristics: Argument:= *P or 1P + T, Task:= Classification, Domain:= Numeric. However, in the future, this may lead to different approaches. For understanding the patterns of each feature, we extracted the meta-features of x and also extracted the meta-features of the data set when it does not have the specific features $ext - x$.

5.3 Experimental setup

Our experiment was conducted in R an Python, and our approach has several challenges and restriction, for simplicity, we are not proposing new metadata variables, we are only using the metadata provided by the MFE library [15] and regarding the case of an individual feature, we only have a few variables that can be applied for an individual feature because meta-learning is prepared to work with a set of variables and not an individual feature. Considering that P is the predictive attribute and T is the target, we could extract the following meta-features for x: $attrToInst, catToNum, instToAttr, nrAttr, nrBin, nrCat, nrInst, nrNum, numToCat$. All of them is well described in the paper [15].

For the $ext - x$ set, we extract all the features contained in the groups general, statistical and model-based, as we have done before on the previous extractions for the full data set. After collecting our meta-data, we need to test how it performs. Since many different features can extend a data set, this means that multiple meta-level examples will be generated from a single data set. To have a reliable estimate of the performance of the approach, we split the meta-examples between training and test sets, using cross-validation with 10 folds divided by groups of data sets. In other words, one specific data set cannot have some part of the data on the train split and another part on the test split. We believe if this is allowed, the performance of the model would be optimistic, and the samples for train and test would not be independent. Given that the approach is computationally heavy, we were only able to collect results for 93 out of the 100 data sets.

Table 5.1: Comparison between two hypothesis for the same approach

	Accuracy		**AUC - ROC**	
	Average (%)	sd (+/-)	Average (%)	sd (+/-)
First set of meta-features	61.56	(0.35)	50.99	(0.37)
Second set of meta-features	65.52	(0.20)	59.80	(0.42)

5.4 Results

Concerning the class distribution for this meta-dataset, we have a reasonably balanced data. In total we have 23,930 features, which generate one meta-example each. Out of these, 14,640 features should be discarded and 9,290 should be kept. Thus, the baseline error is 39%.

This first set of meta-features uses the ratio combination of mfe_x and mfe_{ext-x} and it obtained 61.56% of accuracy with a standard deviation of +/- 0.35, and an AUC - ROC of 50.99% (+/- 0.37). See the table 5.1. Therefore, we applied a second set of meta-features for the same approach, instead of using only the combination of mfe_x and mfe_{ext-x} we also aggregate in the same data set the metadata of mfe_x and mfe_{ext-x}, resulting in a data set with more information about the feature. Considering this metadata set, we tested the second set of meta-features and reach in an accuracy of 65.52% and standard deviation of +/- 0.20, while our AUC - ROC has 59.80% (+/- 0.42).

To statistically prove results we used the correlated Bayesian t-test [1], considering using the rope min and max as +/- 0.05 including that for the left the baseline is the best approach.The results show the comparison between our approach versus the baseline, revealing that our approach statistically can be better than our baseline in 93.93% of the cases, the baseline will be better in 5.94% of the cases, and a draw happens on 0.12%.

Considering the restriction that we have, we can consider it as a satisfactory result that needs to be improved in further research. We are aware that for large data sets, our approach takes too much time, and maybe another approach can deliver a better result on time. We assume that the simplification we have made transforms the practical utility of this approach, ineffective. However, it allows us to answer our question about whether it is possible to use the meta-learning approach to decide to use a feature or discard it.

5.4.1 Base level

Although the results at the meta-level are promising, it is necessary to see how the base-level models learned on the selected features perform. Our approach does not make predictions concerning

Table 5.2: Comparison between accuracy of the approaches

	Average loss/gain (%)	**Max (%)**	**Min (%)**
Selected vs All	-0.72	10.4	-24
Selected vs Original	-0.03	36.7	-7.3

sets of features but a single feature and we have collected the performance of each feature for each data set, so we decided to use only the features that supposedly lead to an improvement in performance. This approach needs further research to address this issue.

Unfortunately, using only the features that our approach suggests, the average accuracy dropped by 0.72 percentage points compared to applying all generated features. Furthermore, the same method still dropping on an average of 0.043 percentage points compared to the original data set. See figure 5.1 that illustrates the accuracy difference of the two approaches.

According to table 5.2, the selected features had a maximum loss of 24 percentage points and a maximum gain of 10.4 percentage points compared to the data set with all the features created. However, comparing the selected results with the original data set results, we have a maximum loss of 7.3 percentage points, while we have a maximum gain of 36.7 percentage points.

Again, our biggest improvement was in the Hill-valley data set reaching an accuracy of 93.48%, losing -3.79 percentage points when compared to all generated features, which has an accuracy of 97.27%, and if compared to the original data set it is a gain of 36.7 percentage points. See detailed results in Appendix A.

These results indicate that even with the worse result compared to using all the features, and we can sometimes improve the results of the original data set and the extended data set. We do have a higher average loss with this approach, but it shows us that the approach can be improved and maybe we can overcome some results. The overall results indicate that even we collected promising results on the meta-level, it leads us to a worse result on the base-level. This is preliminary work and several challenges were addressed with simple solutions that could cause some loss of performance, as the selection of the final set of features and data characterisation. However, it shows us that the approach can be improved and it needs further investigations.

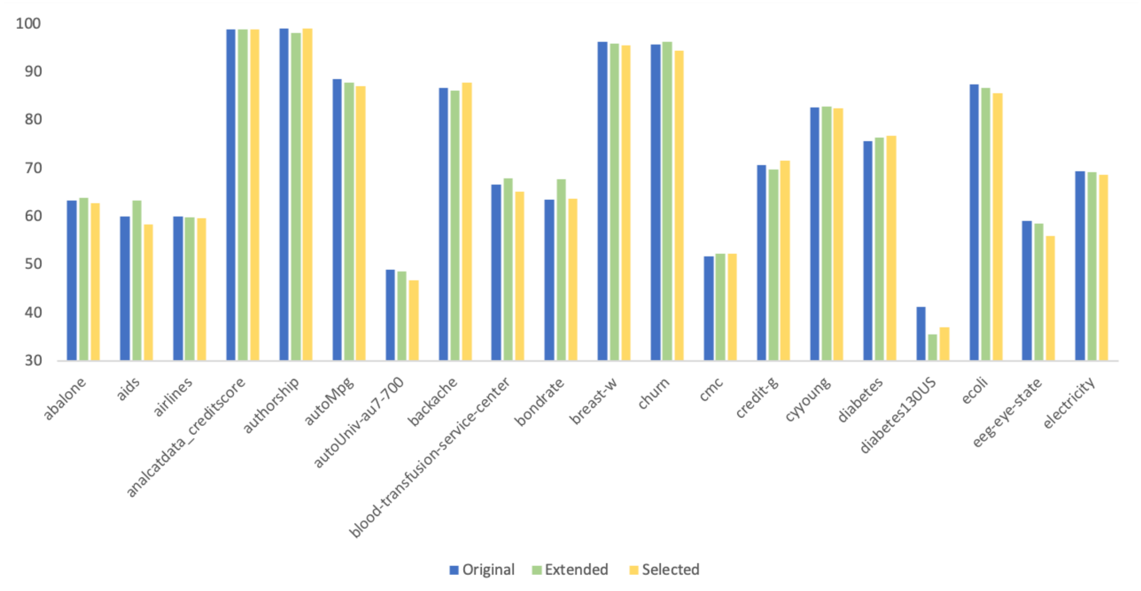

Figure 5.1: Comparison of the accuracy from selected features, complete set of features and original features

Chapter 6

Conclusions and Future Work

Preparing a data set correctly usually takes a significant amount of time and it requires substantial knowledge about the domain. An essential task on this process is feature engineering, which consists of generating new features from the existing ones. This is one of the most complex and time-consuming tasks today.

There is an increase of research on feature engineering over the last years, with a growing interest in automated machine learning. Our results show that automated approaches can extract many features from the raw data set independently of the problem and domain knowledge, often leading to more accurate models.

Previous research has focused on applying different operations on the raw data set. In this thesis, we propose a meta-learning approach to automate the process of feature engineering. Thus given a data set we systematically generate a set of new features, and as it can generate a vast number of features we use a filter to drop the highly correlated features using the filter correlation method.

Our approach uses meta-features that describe the extension and the original data sets, and a combination of both to quantify the differences between them. In this project, we used ratios for that purpose. This approach was used to answer: *"Can a meta-learning approach predict when a method for the systematic generation of features leads to better predictive models?"*. The empirical study carried out focuses only on numerical variables. However, the approach can be applied to variables of any nature.

We developed another meta-learning approach to the problem of predicting whether a new feature will lead to more accurate models. This problem has the additional challenge of characterising a single variable, while typical meta-features characterise data sets. We extracted the information for each new feature and the extension data set without the specific feature, also combining them to collect the difference between the data sets. This approach was used to answer our question *"Can a meta-learning approach predict when an individual feature leads to a better predictive model?"*

6.1 Results

This thesis proposed to identify the effects of automated feature generation using meta-learning as the approach. Based on empirical results, and we can assume that a systematic combination of features using simple operations can lead to better predictive models. Applying a simple filter of highly correlated features it is possible to improve the model performance.

Additionally, the empirical results show that the meta-learning approach can predict when a method for the systematic generation of features leads to better predictive models. The results also indicate that the meta-learning approach can predict when a method for automatic generation of features leads to better predictive models without applying it, using only the original data set without the metadata of the new features.

On the other hand, we did not get satisfactory results on the problem of predicting whether a feature will lead to better models. Even though the results at the meta-level were promising, the selected features did not lead to better base-level models.

However, this is a too complex problem and we decided to simplify, causing some limitations to the problem. Limitations such as the characterisation of the data and the computational cost to generate a meta-data of each feature. In addition, the results indicate that there is more research to be done for the selection of features using the meta-learning approach, applying different techniques for the same approach.

6.2 Future Work

For future work, we see the attempt to eliminate some limitations in the current work.

For the selection of individual features, we would like to address a more realistic approach: instead of removing only a single feature, remove a random fraction of features from the data set, repeating the process many times. Thus, avoiding the restriction, we had to generate metadata for a single feature, as investigate if it is possible to build new meta-features for an individual feature or a new approach for the selection of features.

New approaches should also seek to mitigate time consumption in metadata generation, making this approach more practical. Moreover, it would be good to implement other feature generation transformations such as unary and higher-order transformations. Additionally, we could combine the feature generation tools with the meta-learning approach cited here.

During the process of analysis of this work, it was possible to see the impact that a single data set generates on the final result, so an obvious step would be to add new data sets, mainly data sets with a larger volume.

Here, we only used the Random Forest algorithm. However, there are many algorithms that we can use for this problem and should be considered to make a comparison in future work.

Appendix A

Base level accuracy

The tables A.1, A.2, A.3 are related with the original and extended features without the filter of highly correlated features. The tables A.4, A.5, A.6 is about the original and extended features with the filter of highly correlated features. The tables A.7, A.8, A.6 is about the accuracy of each data set using only the individual features that supposedly would lead a better performance to the model.

Table A.1: Accuracy of each data set tested without filter on base-level.

Data sets	**Extended**	**Original**	**Extended vs Original**	**Class**
abalone	63.97	63.44	0.53	1
aids	63.33	60	3.33	1
airlines	59.94	60	-0.06	0
analcatdata_creditscore	98.89	98.89	0	0
authorship	98.1	99.17	-1.07	0
autoMpg	87.92	88.67	-0.75	0
autoUniv-au7-700	48.61	48.99	-0.38	0
backache	86.28	86.75	-0.47	0
blood-transfusion-service-center	68.06	66.6	1.46	1
bondrate	67.81	63.57	4.24	1
breast-w	96.01	96.29	-0.28	0
churn	96.28	95.74	0.54	1
cmc	52.34	51.79	0.55	1
credit-g	69.9	70.8	-0.9	0
cyyoung	82.97	82.75	0.22	1
diabetes	76.43	75.78	0.65	1
diabetes130US	35.59	41.25	-5.66	0
ecoli	86.7	87.57	-0.87	0
eeg-eye-state	58.64	59.21	-0.57	0
electricity	69.31	69.41	-0.1	0
glass	74.61	82.13	-7.52	0
haberman	67.38	64.78	2.6	1
heart-statlog	83.33	84.07	-0.74	0
hepatitis	75.03	75.03	0	0
houses	94.67	94.28	0.39	1
kddcup09_upselling	92.44	92.47	-0.03	0
letter	97.28	96.57	0.71	1
mammography	98.83	98.75	0.08	1
morphological	99.85	99.85	0	0
page-blocks	96.89	96.75	0.14	1
phoneme	91.17	91.36	-0.19	0
prnn_crabs	99.5	80	19.5	1
profb	63.4	59.67	3.73	1
rmftsa_sleepdata	36.76	36.25	0.51	1
satellite	99.33	99.29	0.04	1
satimage	91.66	91.85	-0.19	0
sonar	69.95	73.72	-3.77	0
tae	81.14	80.43	0.71	1
teachingassistant	63.99	65.2	-1.21	0
titanic	64.4	64.17	0.23	1
vehicle	78.25	75.99	2.26	1
vinnie	80.47	80.47	0	0
volcanoes	96.37	96.44	-0.07	0
wall-robot-navigation	99.58	99.76	-0.18	0
wholesale-customers	92.26	91.82	0.44	1
wine	96.14	96.73	-0.59	0
yeast	58.85	59.05	-0.2	0

Table A.2: Accuracy of each data set tested without filter on base-level [continuity].

Data sets	**Extended**	**Original**	**Extended vs Original**	**Class**
allbp	97.77	97.67	0.1	1
thyroid-ann	99.79	99.58	0.21	1
autos	69.15	67.83	1.32	1
optdigits	97.68	97.69	-0.01	0
engine	81.44	86.2	-4.76	0
calendardow	58.88	58.11	0.77	1
led-display-domain	71.51	71.77	-0.26	0
smartphone-based _recognition_of_ human_activities	96.67	97.22	-0.55	0
steel-plates-fault	97.16	91.52	5.64	1
volcanoes-d4	94.04	94.2	-0.16	0
BNG_breast-w	98.61	98.61	0	0
BNG_cmc	52.87	52.02	0.85	1
qsar-biodeg	85.4	84.64	0.76	1
kc2	80.19	80.79	-0.6	0
ozone-level-8hr	93.45	92.66	0.79	1
hill-valley	97.27	56.78	40.49	1
wdbc	96.85	96.15	0.7	1
climate-model -simulation-crashes	91.12	91.31	-0.19	0
spambase	93.59	94.09	-0.5	0
ilpd	68.24	68.26	-0.02	0
kc1	83.46	83.13	0.33	1
pc1	93.51	93.7	-0.19	0
pc3	90.08	89.95	0.13	1
pc4	91.29	91.02	0.27	1
banknote -authentication	99.85	99.34	0.51	1
mozilla4	84.89	84.55	0.34	1
pendigits	99.37	99.07	0.3	1
balance-scale	84.5	68.42	16.08	1
cardiotocography	100	100	0	0
first-order -theorem-proving	57.71	57.66	0.05	1
wilt	98.39	98.16	0.23	1
thyroid-allhyper	70.29	70.68	-0.39	0
thyroid-allbp	70.29	70.68	-0.39	0
thyroid-allhypo	70.29	70.68	-0.39	0
thyroid-allrep	70.29	70.68	-0.39	0
thyroid-dis	70.29	70.68	-0.39	0
satellite_image	89.9	89.68	0.22	1
seismic-bumps	91.26	92.19	-0.93	0
segment	98.1	97.88	0.22	1

Table A.3: Accuracy of each data set tested without filter on base-level [continuity].

Data sets	**Extended**	**Original**	**Extended vs Original**	**Class**
breast-tissue	51.01	66.48	-15.47	0
credit-approval	75.12	73.94	1.18	1
banana	89	89.68	-0.68	0
pc1_req	67.46	66.49	0.97	1
parkinsons	84.06	83.59	0.47	1
iris	96	96.67	-0.67	0
jungle_chess_2pcs_raw_endgame _complete	71.61	68.22	3.39	1
oil_spill	95.19	95.83	-0.64	0
thoracic-surgery	83.62	84.04	-0.42	0
jm1	79.18	79.05	0.13	1
JapaneseVowels	93.23	92.52	0.71	1
eucalyptus	22.13	20.79	1.34	1
allrep	98.54	97.8	0.74	1
user-knowledge	90.73	90.2	0.53	1

Table A.4: Accuracy of each data set tested with filter on base-level.

Data sets	**Extended Filtered**	**Original Filtered**	**Extended vs Original**	**Class**
abalone	64.35	54.66	9.69	1
aids	60.83	60	0.83	1
airlines	59.81	60	-0.19	0
analcatdata_creditscore	98.89	98.89	0	0
authorship	98.81	99.17	-0.36	0
autoMpg	89.71	86.64	3.07	1
autoUniv-au7-700	46.29	49.28	-2.99	0
backache	85.63	86.75	-1.12	0
blood-transfusion -service-center	67.13	67.53	-0.4	0
bondrate	64.38	63.57	0.81	1
breast-w	95.87	96.29	-0.42	0
churn	95.84	94.96	0.88	1
cmc	52.54	51.79	0.75	1
credit-g	69.9	70.8	-0.9	0
cyyoung	83.72	84.97	-1.25	0
diabetes	76.3	75.78	0.52	1
diabetes130US	44.88	41.25	3.63	1
ecoli	86.93	87.89	-0.96	0
eeg-eye-state	57.03	54.57	2.46	1
electricity	68.92	69.41	-0.49	0
glass	74.47	82.13	-7.66	0
haberman	69.36	64.78	4.58	1
heart-statlog	82.22	84.07	-1.85	0
hepatitis	75.03	75.03	0	0
houses	94.63	94.26	0.37	1
kddcup09_upselling	92.46	92.47	-0.01	0
letter	97.24	96.57	0.67	1
mammography	98.84	98.75	0.09	1
morphological	99.85	99.75	0.1	1
page-blocks	97.09	96.77	0.32	1
phoneme	91.52	91.36	0.16	1
prnn_crabs	99.5	75.5	24	1
profb	62.8	59.67	3.13	1
rmftsa_sleepdata	36.25	36.25	0	0
satellite	99.45	99.29	0.16	1
satimage	90.96	90.08	0.88	1
sonar	70.11	73.72	-3.61	0
tae	79.1	80.43	-1.33	0
teachingassistant	60.66	65.2	-4.54	0
titanic	64.55	64.17	0.38	1
vehicle	78.61	75.53	3.08	1
vinnie	80.98	80.47	0.51	1
volcanoes	96.34	96.44	-0.1	0

Table A.5: Accuracy of each data set tested with filter on base-level [continuity].

Data sets	Extended Filtered	Original Filtered	Extended vs Original	Class
wall-robot-navigation	99.65	99.82	-0.17	0
wholesale-customers	91.35	91.82	-0.47	0
wine	96.7	96.73	-0.03	0
yeast	58.65	59.19	-0.54	0
allbp	97.61	97.43	0.18	1
thyroid-ann	99.66	99.58	0.08	1
autos	66.37	70.27	-3.9	0
optdigits	97.77	97.69	0.08	1
engine	84.36	86.01	-1.65	0
calendardow	57.36	59.38	-2.02	0
led-display-domain	71.7	71.77	-0.07	0
smartphone-based _recognition_of_ human_activities	97.22	97.22	0	0
steel-plates-fault	97.16	93.62	3.54	1
volcanoes-d4	94.06	94.2	-0.14	0
BNG_breast-w	98.62	98.61	0.01	1
BNG_cmc	52.08	52.02	0.06	1
qsar-biodeg	85.49	84.64	0.85	1
kc2	79.98	79.41	0.57	1
ozone-level-8hr	93.84	93.84	0	0
hill-valley	97.61	49.33	48.28	1
wdbc	97.37	95.45	1.92	1
climate-model-simulation-crashes	91.31	91.5	-0.19	0
spambase	93.26	94.04	-0.78	0
ilpd	66.73	67.92	-1.19	0
kc1	84.12	83.12	1	1
pc1	93.61	93.34	0.27	1
pc3	90.08	90.15	-0.07	0
pc4	91.43	90.6	0.83	1
banknote-authentication	99.93	99.34	0.59	1
mozilla4	85.39	84.55	0.84	1
pendigits	99.34	99.07	0.27	1
balance-scale	84.65	68.42	16.23	1
cardiotocography	100	100	0	0
first-order-theorem-proving	57.68	57.55	0.13	1
wilt	98.26	94.32	3.94	1
thyroid-allhyper	70.5	71.04	-0.54	0
thyroid-allbp	70.5	71.04	-0.54	0
thyroid-allhypo	70.5	71.04	-0.54	0
thyroid-allrep	70.5	71.04	-0.54	0
thyroid-dis	70.5	71.04	-0.54	0
satellite_image	89.14	87.91	1.23	1
seismic-bumps	91.14	91.68	-0.54	0
segment	98.18	98.01	0.17	1

Table A.6: Accuracy of each data set tested with filter on base-level [continuity].

Data sets	**Extended Filtered**	**Original Filtered**	**Extended vs Original**	**Class**
breast-tissue	57.4	66.48	-9.08	0
credit-approval	75.26	74.09	1.17	1
banana	89	89.6	-0.6	0
pc1_req	67.48	66.49	0.99	1
parkinsons	84.56	86.09	-1.53	0
iris	96.67	94	2.67	1
jungle_chess_2pcs _raw_endgame_complete	71.41	68.22	3.19	1
oil_spill	96.05	95.94	0.11	1
thoracic-surgery	83.4	84.04	-0.64	0
jm1	79.27	79.35	-0.08	0
JapaneseVowels	93.35	92.52	0.83	1
eucalyptus	20.39	20.79	-0.4	0
allrep	98.3	97.93	0.37	1
user-knowledge	90.94	89.95	0.99	1

Table A.7: Accuracy of each data set tested with the selected features on base level.

	Original	**Extended**	**Selected**	**Selected vs Extended**	**Selected vs Original**
abalone	63.44	63.97	62.87	-1.1	-0.57
aids	60	63.33	58.33	-5	-1.67
airlines	60	59.94	59.78	-0.16	-0.22
analcatdata_creditscore	98.89	98.89	98.89	0	0
authorship	99.17	98.1	99.17	1.07	0
autoMpg	88.67	87.92	87.17	-0.75	-1.5
autoUniv-au7-700	48.99	48.61	46.87	-1.74	-2.12
backache	86.75	86.28	87.86	1.58	1.11
blood-transfusion-service-center	66.6	68.06	65.26	-2.8	-1.34
bondrate	63.57	67.81	63.81	-4	0.24
breast-w	96.29	96.01	95.58	-0.43	-0.71
churn	95.74	96.28	94.52	-1.76	-1.22
cmc	51.79	52.34	52.33	-0.01	0.54
credit-g	70.8	69.9	71.7	1.8	0.9
cyyoung	82.75	82.97	82.61	-0.36	-0.14
diabetes	75.78	76.43	76.82	0.39	1.04
diabetes130US	41.25	35.59	37.11	1.52	-4.14
ecoli	87.57	86.7	85.58	-1.12	-1.99
eeg-eye-state	59.21	58.64	56.01	-2.63	-3.2
electricity	69.41	69.31	68.73	-0.58	-0.68
glass	82.13	74.61	78.44	3.83	-3.69
haberman	64.78	67.38	61.23	-6.15	-3.55
heart-statlog	84.07	83.33	84.07	0.74	0
hepatitis	75.03	75.03	75.03	0	0
houses	94.28	94.67	94.68	0.01	0.4
KDDCup09_upselling	92.47	92.44	92.47	0.03	0
letter	96.57	97.28	97.23	-0.05	0.66
mammography	98.75	98.83	98.75	-0.08	0
morphological	99.85	99.85	99.75	-0.1	-0.1
page-blocks	96.75	96.89	96.89	0	0.14
phoneme	91.36	91.17	91.14	-0.03	-0.22
prnn_crabs	80	99.5	75.5	-24	-4.5
profb	59.67	63.4	60.42	-2.98	0.75
rmftsa_sleepdata	36.25	36.76	36.25	-0.51	0
satellite	99.29	99.33	99.47	0.14	0.18
satimage	91.85	91.66	90.76	-0.9	-1.09
sonar	73.72	69.95	71.83	1.88	-1.89
tae	80.43	81.14	80.43	-0.71	0
teachingAssistant	65.2	63.99	57.9	-6.09	-7.3
titanic	64.17	64.4	64.17	-0.23	0
vehicle	75.99	78.25	76.11	-2.14	0.12
vinnie	80.47	80.47	80.47	0	0
volcanoes	96.44	96.37	96.32	-0.05	-0.12

Table A.8: Accuracy of each data set tested with the selected features on base level [continuity].

	Original	**Extended**	**Selected**	**Selected vs Extended**	**Selected vs Original**
wall-robot-navigation	99.76	99.58	99.63	0.05	-0.13
wholesale-customers	91.82	92.26	91.58	-0.68	-0.24
wine	96.73	96.14	95.03	-1.11	-1.7
yeast	59.05	58.85	57.39	-1.46	-1.66
allbp	97.67	97.77	97.48	-0.29	-0.19
thyroid-ann	99.58	99.79	99.2	-0.59	-0.38
autos	67.83	69.15	65.01	-4.14	-2.82
optdigits	97.69	97.68	97.81	0.13	0.12
engine	86.2	81.44	86.2	4.76	0
calendardow	58.11	58.88	58.14	-0.74	0.03
led-display-domain	71.77	71.51	71.97	0.46	0.2
smartphone-based _recognition_of_ human_activities	97.22	96.67	97.22	0.55	0
steel-plates-fault	91.52	97.16	93.62	-3.54	2.1
volcanoes-d4	94.2	94.04	94.02	-0.02	-0.18
BNG_breast-w	98.61	98.61	98.6	-0.01	-0.01
BNG_cmc	52.02	52.87	52.02	-0.85	0
qsar-biodeg	84.64	85.4	85.49	0.09	0.85
kc2	80.79	80.19	79.6	-0.59	-1.19
ozone-level-8hr	92.66	93.45	93.84	0.39	1.18
hill-valley	56.78	97.27	93.48	-3.79	36.7
wdbc	96.15	96.85	96.49	-0.36	0.34
climate-model-simulation-crashes	91.31	91.12	91.49	0.37	0.18
spambase	94.09	93.59	93.56	-0.03	-0.53
ilpd	68.26	68.24	68.24	0	-0.02
kc1	83.13	83.46	83.55	0.09	0.42
pc1	93.7	93.51	93.52	0.01	-0.18
pc3	89.95	90.08	90.34	0.26	0.39
pc4	91.02	91.29	91.63	0.34	0.61
banknote-authentication	99.34	99.85	99.85	0	0.51
mozilla4	84.55	84.89	85.11	0.22	0.56
pendigits	99.07	99.37	99.04	-0.33	-0.03
balance-scale	68.42	84.5	79.36	-5.14	10.94
cardiotocography	100	100	100	0	0
first-order-theorem-proving	57.66	57.71	57.94	0.23	0.28
wilt	98.16	98.39	98.04	-0.35	-0.12
thyroid-allhyper	70.68	70.29	70.71	0.42	0.03
thyroid-allbp	70.68	70.29	70.71	0.42	0.03
thyroid-allhypo	70.68	70.29	70.71	0.42	0.03
thyroid-allrep	70.68	70.29	70.71	0.42	0.03
thyroid-dis	70.68	70.29	70.71	0.42	0.03
satellite_image	89.68	89.9	88.67	-1.23	-1.01
seismic-bumps	92.19	91.26	91.41	0.15	-0.78
segment	97.88	98.1	98.14	0.04	0.26

Table A.9: Accuracy of each data set tested with the selected features on base level [continuity].

	Original	**Extended**	**Selected**	**Selected vs Extended**	**Selected vs Original**
breast-tissue	66.48	51.01	61.41	10.4	-5.07
credit-approval	73.94	75.12	73.81	-1.31	-0.13
banana	89.68	89	89	0	-0.68
pc1_req	66.49	67.46	67.46	0	0.97
parkinsons	83.59	84.06	86.54	2.48	2.95
iris	96.67	96	94.67	-1.33	-2
jungle_chess_2pcs_raw_endgame_complete	68.22	71.61	62.57	-9.04	-5.65
oil_spill	95.83	95.19	95.4	0.21	-0.43
thoracic-surgery	84.04	83.62	84.68	1.06	0.64
jm1	79.05	79.18	79.27	0.09	0.22
JapaneseVowels	92.52	93.23	93.35	0.12	0.83
eucalyptus	20.79	22.13	19.56	-2.57	-1.23
allrep	97.8	98.54	98.12	-0.42	0.32
user-knowledge	90.2	90.73	86.97	-3.76	-3.23

References

[1] Alessio Benavoli, Giorgio Corani, Janez Demšar, and Marco Zaffalon. Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. *The Journal of Machine Learning Research*, 18(1):2653–2688, 2017.

[2] Sibanjan Das and Umit Mert Cakmak. *Hands-On Automated Machine Learning*. Packt, First edition, 2018.

[3] Raul Eulogio. Introduction to random forests. Available at https://www.datascience.com/resources/notebooks/random-forest-intro/, Accessed last time in December 2018, 2017.

[4] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[5] Bernd Bischl Matthias Feurer Giuseppe Casalicchio Heidi Seibold Andreas Mueller Joaquin Vanschoren, Jan van Rijn. Openml data sets. Available at https://www.openml.org/search?type=data, Accessed last time in March 2019.

[6] James Max Kanter and Kalyan Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.

[7] Asha Gowda Karegowda, AS Manjunath, and MA Jayaram. Comparative study of attribute selection using gain ratio and correlation based feature selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277, 2010.

[8] Gilad Katz, Eui Chul Richard Shin, and Dawn Song. Explorekit: Automatic feature generation and selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984. IEEE, 2016.

[9] Ambika Kaul, Saket Maheshwary, and Vikram Pudi. Autolearn—automated feature generation and selection. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 217–226. IEEE, 2017.

[10] Udayan Khurana, Deepak Turaga, Horst Samulowitz, and Srinivasan Parthasrathy. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 1304–1307. IEEE, 2016.

[11] Hoang Thanh Lam, Johann-Michael Thiebaut, Mathieu Sinn, Bei Chen, Tiep Mai, and Oznur Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

[12] Hugh Leather, Edwin Bonilla, and Michael O'Boyle. Automatic feature generation for machine learning based optimizing compilation. In *Proceedings of the 7th annual IEEE/ACM International Symposium on Code Generation and Optimization*, pages 81–91. IEEE Computer Society, 2009.

[13] Carlos Soares Ricardo Vilalta Pavel Brazdil, Christophe Giraud-Carrier. *Metalearning: Applications to Data Mining*. Springer Berlin Heidelberg, 2009.

[14] Randy Kerber Thomas Khabaza Thomas Reinartz Colin Shearer Pete Chapman, Julian Clinton and Rüdiger Wirth. *CRISP-DM 1.0 - Step-by-step data mining guide*. CRISP-DM Consortium, First edition, 2009.

[15] Adriano Rivolli, Luís PF Garcia, Carlos Soares, Joaquin Vanschoren, and André CPLF de Carvalho. Towards reproducible empirical research in meta-learning. *arXiv preprint arXiv:1808.10406*, 2018.

[16] Rostislav Stríz. Metalearning for data mining and kdd. pages 1–21, 2013.

[17] Ricardo Vilalta, Christophe G Giraud-Carrier, Pavel Brazdil, and Carlos Soares. Using meta-learning to support data mining. *IJCSA*, 1(1):31–45, 2004.

[18] Mlle Bouaguel Waad. *On Feature Selection Methods for Credit Scoring*. PhD thesis, Université de Tunis - Institut Supérieur de Gestion, 2015.