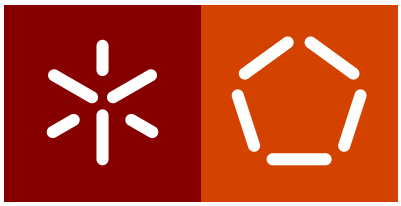
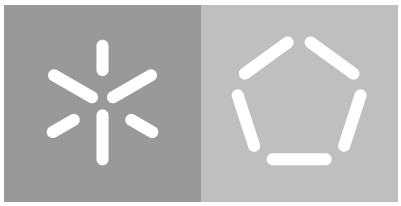


Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Pedro Santos Monteiro

A Meta-Learning Approach for Selecting Machine Learning Algorithms

May 2020



Universidade do Minho
Escola de Engenharia
Departamento de Informática

José Pedro Santos Monteiro

A Meta-Learning Approach for Selecting Machine Learning Algorithms

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
Professor Doutor João M. Fernandes
Professor Doutor Francisco J. Duarte

May 2020

AGRADECIMENTOS

O trabalho realizado para esta dissertação não estaria concluído sem o contributo significativo dos meus orientadores, Professor Doutor João M. Fernandes e Professor Doutor Francisco J. Duarte aos quais agradeço pelas sugestões, correções e pela, igualmente importante, confiança demonstrada.
À empresa Bosch Car Multimédia pela oportunidade, por todo o auxílio e a experiência transmitida. Em especial, um forte agradecimento a toda a equipa de logística que me acolheu.
Aos meus pais e à minha irmã, pela minha educação e acima de tudo pelo amor e apoio incondicional.
À Sara pela paciência, amparo, incentivo e inspiração.
O meu profundo e sentido agradecimento a todos os amigos e colegas que, mesmo não citados nesta lista de agradecimentos, contribuíram de forma direta ou indireta não só para a concretização desta dissertação, mas também para o meu crescimento pessoal e profissional.

A todos, o meu sincero obrigado.

ABSTRACT

One of the major challenges in Machine Learning is to investigate the capabilities and limitations of the existing algorithms to identify when one algorithm is more adequate than another to solve particular problems. Traditional approaches to predicting the performance of algorithms often involve costly trial-and-error procedures or expert knowledge, which is not always straightforward to acquire. Thus, the main goal of this dissertation is to support beginners or even experienced data scientists by automatically indicating which classification algorithm is most suitable for their datasets.

This dissertation proposes the use of Meta-Learning as a possible solution to the above-mentioned problem. In this respect, we introduced a novel framework for the automatic generation of meta-datasets. Taking advantage of the developed framework, several classification datasets from public sources were used. The result is the meta-dataset for the experiment of this research project.

Concerning the goal of forecasting the best model for a classification dataset, two different solutions are presented: the first toward binary classification and the second on multiclass classification. A variety of Machine Learning algorithms are tested and compared through cross-validation.

The experiment confirms the feasibility of applying Meta-Learning to select the algorithm that is expected to obtain the best performance for classification problems.

Keywords: Machine Learning; Meta-Learning; Metadata; Machine Learning algorithms selection; Classification; Data Mining.

RESUMO

Um dos principais desafios do *Machine Learning* passa por investigar os recursos e as limitações dos algoritmos existentes para identificar quando é que um algoritmo é mais adequado do que outro para resolver um determinado problema. Por norma, as abordagens tradicionais envolvem procedimentos de tentativa e erro, que requerem muito tempo ou conhecimento especializado, o que nem sempre é fácil de adquirir. Assim, a presente dissertação pretende auxiliar iniciantes, indivíduos que não são cientistas de dados e até cientistas de dados experientes, indicando automaticamente qual o algoritmo que é mais vantajoso para os seus conjuntos de dados de classificação.

O presente trabalho propõe a utilização de *Meta-Learning* como uma possível solução para o problema acima mencionado. Numa primeira etapa é apresentada uma *Framework* para extração automática de meta-características informativas. Tirando recurso da *Framework* desenvolvida, foram utilizados vários conjuntos de dados de classificação de fontes públicas, gerando assim o meta conjunto de dados para o experimento desta dissertação.

Relativamente à meta previsão do melhor modelo a utilizar, foram abordadas duas soluções: uma primeira focada em classificação binária e a segunda em classificação com múltiplas classes. Em ambas foram testados e comparados vários algoritmos de *Machine Learning* através de validação cruzada.

O experimento confirmou a viabilidade da aplicação de *Meta-Learning* para a seleção de algoritmos com melhor desempenho em problemas de classificação.

Palavras-chave: Machine Learning; Meta-Learning; Metadados; Seleção de Algoritmos; Problemas de classificação; Análise de Dados.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1

INTRODUCTION

1.1 FRAMEWORK AND MOTIVATIONS

Just fifty years ago, Machine Learning (ML) was seen as science fiction, now ML is revolutionizing our society by transforming data into useful predictions and information.

With the increase in the processor speed and memory size, ML has become quite popular, as a result of many algorithms that use mathematical or statistical analysis to learn, draw or infer data. For that reason, ML Algorithms have been categorized based on the purpose for which they are designed [57].

This number continues to increase as evidenced by the number of scientific publications that propose variations or combinations of ML algorithms. Numerous significant commercial applications have already appeared, including recommendation engines, speech and handwriting recognition systems, content identification, image classification/retrieval, automatic captioning, spam filters, and demand forecasting. [48]

However, the ML field does not have a clear classification scheme for its algorithms, mainly because of the high number of approaches and the variations proposed in the literature [51]. A conventional question at the beginning of a project, when facing a wide variety of ML algorithms, is "which algorithm should I use?" The answer to this question can be very difficult to obtain, because it depends on many factors, such as:

- The purpose for which the data is being used.
- The urgency of the task.
- The size, quality, and nature of data.
- The available computational time.

Even an experienced data scientist may have doubts and difficulties in telling which algorithm will perform the best before trying different algorithms. Hence, it is an important task to decide for any given set of data which method produces the best results. Selecting the best approach can be one of the most challenging parts of performing statistical learning in practice [38].

Meta-Learning for algorithm selection arises in this context as an effective solution, capable of automatically predicting an algorithm's performance, thus assisting users in the choice of the most adequate techniques for dealing with the problems at hand [19].

This dissertation focuses on using Meta-Learning to address the automatic selection of algorithms via features extracted from the set of problems to be solved.

1.2 OBJECTIVES

With the purpose of tackling the problem stated before, this dissertation intends to:

- Devise a recent in-depth review of the scientific literature of data mining methodologies, ML algorithms and meta-learning.
- Analyze and study the influence of meta-learning in the algorithm selection problem.
- Develop a generic meta-dataset generator framework for meta-learning-based algorithm ranking. The framework aims to significantly improve the predictive performances of different meta-learners.
- Apply and evaluate the application of ML algorithms to select the most appropriate algorithm given a new unseen meta-dataset instance.
- Develop insight of how different metrics and different evaluation criterion affect the performance of the different used algorithms.

Thus, the main goal of this dissertation is to support beginners or even experienced data scientists by automatically indicating which classification algorithm is most suitable for their datasets.

1.3 DISSERTATION OUTLINE

This research project is composed of two main Parts.

The First Part introduces well known concepts related to Data Mining (Chapter 2), ML algorithms (Chapter 3) and Meta-Learning (Chapter 4), taken from the literature. In Chapter 4, a brief explanation regarding the application of meta-learning for the algorithm recommendation is also provided. Throughout the First Part, some scientific literature references and examples are given.

The Second Part uses the theoretical background raised in the First Part to design and implement the Meta-Dataset Generator (Chapter 5). We showcase the application of the Meta-Dataset Generator for the binary classification algorithms selection, using different algorithms and different performance metrics.

The following Chapter - Machine Learning Algorithm Recommendation - goes through the six steps of the application of the Knowledge Discovery Process model to propose a solution to the main topic of this research, the automatic algorithm selection.

This dissertation ends with a section devoted to conclusions and some suggestions for future work.

Part I

STATE OF THE ART

2

DATA MINING

2.1 INTRODUCTORY REMARKS

As mentioned before, our society is overwhelmed with huge amounts of collected data. We are living in the data age, data collecting and storing is a regular practice for most of today's companies and business. The term Data Mining (DM) is often used as a synonym for Knowledge Discovery from Data or KDD, which is a process that can provide *new, interesting, non-trivial, hidden and potentially useful knowledge* about collected data. The seminal book "Data Mining Concepts and Techniques" [33] establishes that DM is technically only a small part of the whole *search-for-knowledge* process and the process itself contains the following parts:

- **Data cleaning** (to remove noise and inconsistent data)
- **Data integration** (where multiple data sources may be combined)
- **Data selection** (where data relevant to the analysis task are retrieved from the database)
- **Data transformation** (where data are transformed and consolidated into forms appropriate for mining by performing summary or aggregation operations)
- **Data mining** (an essential process where intelligent methods are applied to extract data patterns)
- **Pattern evaluation** (to identify the truly interesting patterns representing knowledge based on *interestingness measures*)
- **Knowledge presentation** (where visualization and knowledge representation techniques are used to present mined knowledge to users)

The first four steps are often referred to as *data preprocessing*.

2.2 KNOWLEDGE DISCOVERY PROCESS MODELS

There are no known models that focus only on the selection of ML algorithms, but there are some methodologies for analytics, DM, and data science projects. In [16] the authors describe the knowledge discovery process (also called knowledge discovery in databases) as a nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. The process generalizes to non-database sources of data, although it emphasizes databases as the primary source of data. To understand what leads to the choice of ML algorithms is essential to comprehend these methodologies/models. Although the models usually emphasize independence from specific applications and tools, they can be broadly divided into those that take into account industrial issues and those that do not. However, the academic models, which are usually not concerned with industrial issues, can be made applicable relatively easily in the industrial setting and vice versa. The discussion about the models was restricted to the models that have been popularized in the literature and have been used in real knowledge discovery projects.

2.2.1 *Academic Research Models*

The efforts to establish a KDP model were initiated in academia. In the mid-1990s, when the DM field was being shaped, researchers started defining multi-step procedures to guide users of DM tools in the complex knowledge discovery world. The main emphasis was to provide a sequence of activities that would help to execute a KDP in an arbitrary domain. The two-process models developed in 1996 and 1998 are the nine-step model [25]. The nine parts can be defined as:

1. *Developing and understanding the application domain*. This step includes learning the relevant prior knowledge and the goals of the end-user of the discovered knowledge.

2. *Creating a target data set*. Here the data miner selects a subset of variables (attributes) and data points (examples) that will be used to perform discovery tasks. This step usually includes querying the existing data to select the desired subset.

3. *Data cleaning and preprocessing*. This step consists of removing outliers, dealing with noise and missing values in the data, and accounting for time sequence information and known changes.

4. *Data reduction and projection.* This step consists of finding useful attributes by applying dimension reduction and transformation methods and finding invariant representation of the data.

5. *Choosing the data mining task.* In this step, the data scientist matches the goals defined in the first step with a particular DM method, such as classification, regression, clustering, etc... (see chapter 3).

6. *Choosing the data mining algorithm.* In this step, the data scientist selects methods to search for patterns in the data and decides which models and parameters of the methods used may be appropriate.

7. *Data mining.* This step generates patterns in a particular representational form, such as classification rules, decision trees, regression models, trends, etc (see chapter 3).

8. *Interpreting mined patterns.* In this step, the data analyst performs some visualization of the extracted patterns and models.

9. *Consolidating discovered knowledge.* The final step consists of incorporating the discovered knowledge into the performance system, documenting and reporting it to the interested parties. This step may also include checking and resolving potential conflicts with previously believed knowledge.

The number of loops between any two steps is usually executed, but they give no specific details. The model provides a detailed technical description for data analysis but lacks a description of business aspects. This model has become a cornerstone of later models [25]. Thus, we can infer that the process is iterative [16].

2.2.2 *Industrial Models*

Industrial models quickly followed academic efforts. Several different approaches were undertaken, ranging from models proposed by individuals with extensive industrial experience to models proposed by large industrial consortiums. The most representative industrial model is the industrial six-step CRISP-DM model, developed by a large consortium of European companies, aimed to create a standard, non-proprietary and free process model for the development of DM [14; 16].

Hierarchical breakdown

The CRISP-DM methodology is described in terms of a hierarchical process model, consisting of sets of tasks described at four levels of abstraction (from general to specific): phase, generic task, specialized task, and process instance.

The CRISP-DM reference model

CRISP-DM is divided into six phases, as shown in Figure 1. The sequence of the phases is not rigid. Moving back and forth between different phases is always required. The outcome

of each phase determines which phase, or particular task of a phase, has to be performed next. The arrows indicate the most important and frequent dependencies between phases [14].

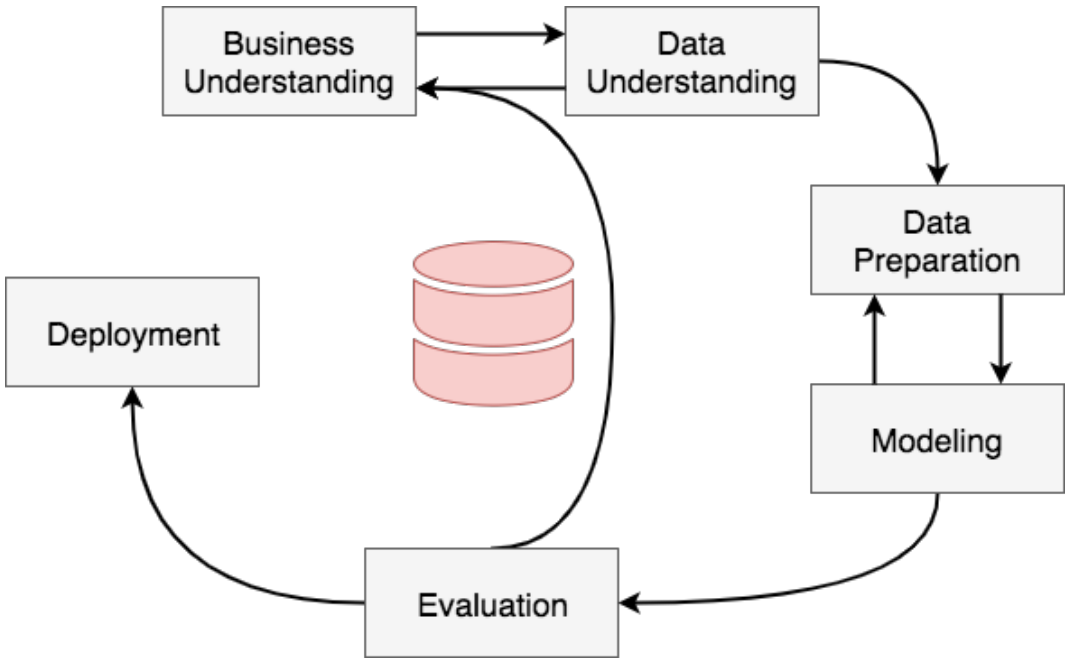


Figure 1.: Phases of the CRISP-DM reference model (adapted from [14])

Business Understanding

This initial phase focuses on understanding the project objectives and requirements from a business perspective, then converting this knowledge into a data mining problem definition and a preliminary plan designed to achieve the objectives.

Data Understanding

The data understanding phase starts with initial data collection and proceeds with activities that enable the data analyst to become familiar with the data, such as, identifying data quality problems, discovering first insights into the data, and/or detecting interesting subsets to form hypotheses regarding hidden information.

Data Preparation

This stage encompasses all the activities necessary to construct the *dataset* or *datasets*, which will be used in the modeling phase (including ML algorithms). Tasks include table, record, and attribute selection, as well as transformation and cleaning of data for modeling tools.

Modeling

In this phase, various modeling techniques are selected and applied, and their parameters are calibrated to optimal values. Typically, there are several techniques (including ML algorithms) for the same DM problem type. Some techniques have specific requirements in the form of data. Consequently, stepping back to the data preparation phase is often needed.

Evaluation

The Evaluation stage consists of evaluating model results based on business metrics created at the beginning of the project and then refining the model to prepare it for deployment. A key objective of this phase is to determine if there is some important business issue that has not been sufficiently considered. At the end of this phase, a decision on the use of the DM results should be reached.

Deployment

The creation of the model is generally not the end of the project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that the customer can use it. Depending on the requirements, the deployment phase can be as simple as generating a report or as complex as implementing a repeatable DM process.

2.2.3 *Hybrid Models*

The development of academic and industrial models has led to the development of hybrid models. The most known model is a six-step Knowledge Discovery Process (KDP) model (see Figure 2) it was developed based on the CRISP-DM model by adopting it to academic research [16].

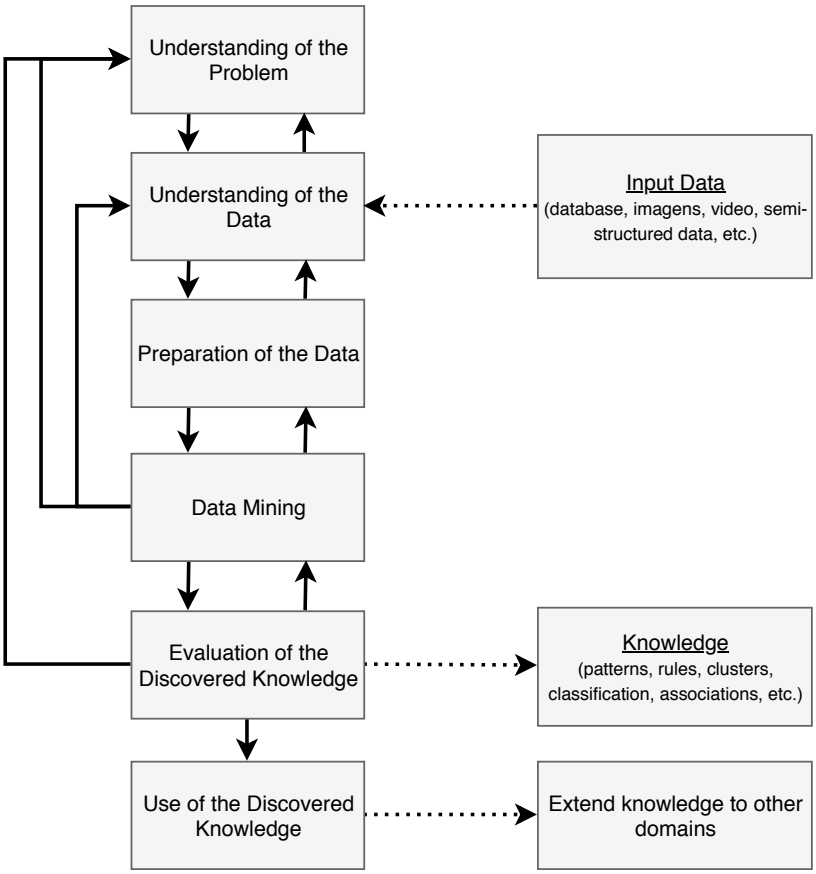


Figure 2.: The six-step KDP model [1]

This hybrid model consists of six steps, which are outlined as follows:

1. *Understanding of the problem domain.* This initial phase focuses on working closely with domain experts to define the problem and determine the project goals, identifying key people, and learning about current solutions to the problem. It also involves learning domain-specific terminology. A description of the problem, including its restrictions, is prepared. Finally, project goals are translated into DM goals, and the initial selection of DM tools to be used later.

2. *Understanding of the data.* This phase includes collecting sample data and deciding which data, including format and size, will be needed. Background knowledge can be used to guide these efforts. Data are checked for completeness, redundancy, missing values, plausibility of attribute values, etc. Finally, the step includes verification of the usefulness of the data concerning the DM goals.

1 This image was adapted from [16]

3. *Preparation of the data.* Mining methods in the subsequent step. It involves sampling, running correlation and significance tests, and data cleaning, which includes checking the completeness of data records, removing or correcting for noise and missing values, etc. The cleaned data may be further processed by feature selection and extraction algorithms (to reduce dimensionality), by derivation of new attributes, and by summarization of data (data granularization). The results are data that meet the specific input requirements for the DM tools selected in the first phase.

4. *Data mining.* In this phase are used various DM methods to derive knowledge from preprocessed data.

5. *Evaluation of the discovered knowledge.* The evaluation phase includes understanding the results, checking whether the discovered knowledge is novel and interesting, interpretation the results by domain experts, and checking the impact of the discovered knowledge. If the results do not go according to was expected, is necessary to return to the first step.

6. *Use of the discovered knowledge*. This final step consists of planning where and how to use the discovered knowledge. The application area in the current domain may be extended to other domains.

The main differences between this approach, is providing more general, research-oriented description of the steps, introducing several new explicit feedback mechanisms, and modification of the last step, since in the hybrid model, the knowledge discovered for a particular domain may be applied in other domains (see, e.g., [16] for more detailed information).

2.3 Concluding Remarks

KDP models were introduced and some literature was referenced (see, e.g., [25; 14; 16]). These robust and well-proven methodologies provide a structured approach for planning a Data Mining project.

All the methodologies presented in this chapter when it comes to the proper algorithm selection only divides the problem into classes, for example, if it is a regression or a classification problem (explained in chapter 3), but the algorithm selection itself is completely relegated to the experience of the data scientist or a trial and error task. At this point, it's known the lack of optimization on the algorithm selection phase (also called modeling), and that highlights the goal of this dissertation, an improvement in the algorithm selection area.

3

MACHINE LEARNING ALGORITHMS

3.1 INTRODUCTORY REMARKS

Machine Learning (ML) is a sub-set of Artificial Intelligence (AI), which has been studied since the late 1950s [53]. According to A. L. Samuel in his remarkable article published in 1959, "Some Studies in Machine Learning Using the Game of Checkers" describes ML as a field of computer science that gives computers the ability to learn without being explicitly programmed [63]. More formally, ML is defined as follows: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"[75].

In the last years, ML resurfaced as a popular technology. In the '90s and 2000s, the internet transformed the way we live and do business, and in the process generated many petabytes of data that contributes to this. Now ML algorithms are used in several areas besides computer science, including business [4], advertising [18] and medicine [32].

Today, there are a large number of ML algorithms proposed in the literature. We can classify them based on the approach used for the learning process. There are some variations of how to define the types of ML algorithms (e.g., [21]) but commonly they can be divided into supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning [52; 66].

Deep learning is a subset of ML, and ML is a subset of AI, which is an umbrella term for any computer program that does something smart.

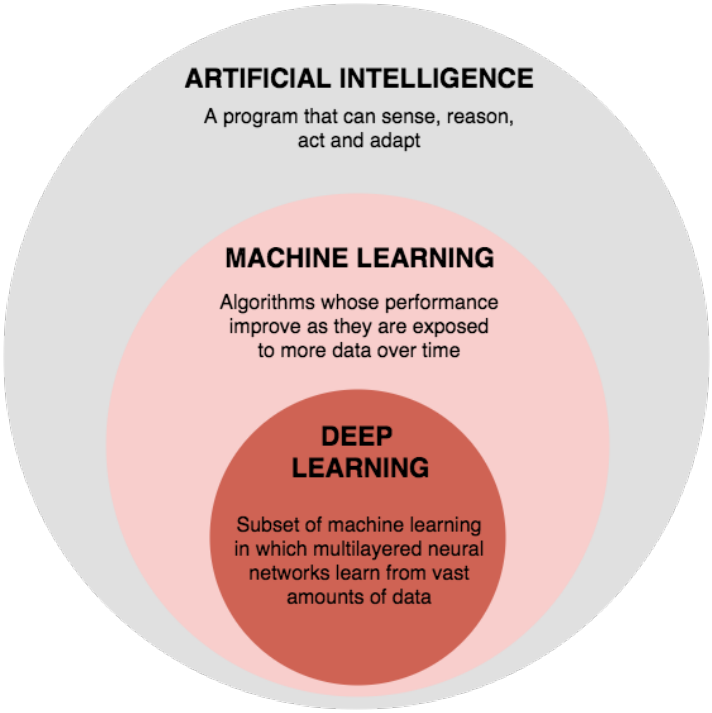


Figure 3.: Artificial Intelligence is an umbrella term, encompassing machine learning and deep learning (adaption from [37])

Hereupon, in this dissertation, the focus will be only on ML and more precisely in Classification, which is a subclass of supervised learning. This Chapter intends to introduce and describe some fundamental learning processes.

3.2 SUPERVISED LEARNING

Supervised learning [47] is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown. Every instance in any dataset used by ML algorithms is represented using the same set of features. The features may be continuous, categorical or binary. If instances are given with known labels (the corresponding correct outputs) then the learning is called supervised (See table 1).

Dataset					
case	Feature 1	Feature 2	...	Feature n	Class
1	x	xxx		xx	True
2	x	xxx		xx	False
3	x	xxx		xx	True
...					...

Table 1.: Instances with known labels (the corresponding correct outputs)

Supervised learning is used in classification(see table 1) and regression (see table 2).

ID	Years of Higher Education (X)	Income (Y)
1	4	80,000
2	5	91,500
3	0	42,000
...	...	...
N	2	100,000

Table 2.: Instances with a quantitative known labels

3.2.1 *Regression Versus Classification Problems*

Supervised learning problems can be further grouped into Regression and Classification problems. Both problems have as goal the construction of a succinct model that can predict the value of the dependent attribute from the attribute variables.

Variables can be characterized as either quantitative or qualitative (also known as categorical). Quantitative variables take on numerical values. Examples include a person's age, height, or income, the value of a house, and the price of a stock. In contrast, qualitative variables take on values in one of K different classes, or categories. Examples of qualitative variables include a person's gender (male or female), the brand of product purchased (brand A, B, or C), whether a person defaults on a debt (yes or no), or a cancer diagnosis (Acute Myelogenous Leukemia, Acute Lymphoblastic Leukemia, or No Leukemia). The literature tend to refer to problems with a quantitative response as regression problems (see Table 2), while those involving a qualitative response are often referred to as classification problems (see Table 1). [38]

Some algorithms can be used for both classification and regression with small modifications, such as decision trees and artificial neural networks. Some algorithms cannot, or cannot easily be used for both problem types, such as linear regression for regression predictive modeling and logistic regression for classification predictive modeling [84; 72]. Table

3 illustrates where some algorithms can applied.

	Classification	Regression
Decision trees	Y	Y[1]
Rule learning	Y	
Neural networks	Y	Y
Linear regression		Y
Logistic regression	Y	
Support Vector Machine	Y	Y

Table 3.: Modelling algorithm classes vs model types (Adaption from [30])

3.3 UNSUPERVISED LEARNING

Unsupervised learning [38] in a statistical view describes the situation in which for every observation $i = 1,...,n$, it can be observed a vector of measurements x_i but no associated response y_i. It is not possible to fit a linear regression model, since there is no response variable to predict. The situation is referred to as unsupervised because we lack a response variable that can supervise our analysis. In unsupervised learning [12], ML algorithms do not have a training set. They are presented with some data about the real world and have to learn from that data on their own. Unsupervised learning algorithms are mostly focused on finding hidden patterns in data. For example, suppose that an ML algorithm has access to user profile information in a social network. By using an unsupervised learning approach, the algorithm can separate users into personality categories, allowing the social network company to target advertising more directly at specific groups of users.

Unsupervised learning problems can be further grouped into clustering and association problems [9], presented in 3.7.5 and 3.7.6, respectively.

3.4 SEMI-SUPERVISED LEARNING

ML algorithms can also be classified as semi-supervised learning. Semi-supervised learning refers to the use of both labeled and unlabeled data for training. It contrasts supervised learning (data all labeled) or unsupervised learning (data all unlabeled), see Figure 4. The goal is to learn a predictor that predicts future test data better than the predictor learned from the labeled training data alone. This approach is motivated by the fact that labeled data is often costly to generate, whereas unlabeled data is generally not. It can be concluded

1 Not all algorithms support.

that semi-supervised learning is motivated by its practical value in learning, faster, better, and cheaper. Therefore, being capable to utilize the surplus of unlabeled data is desirable.

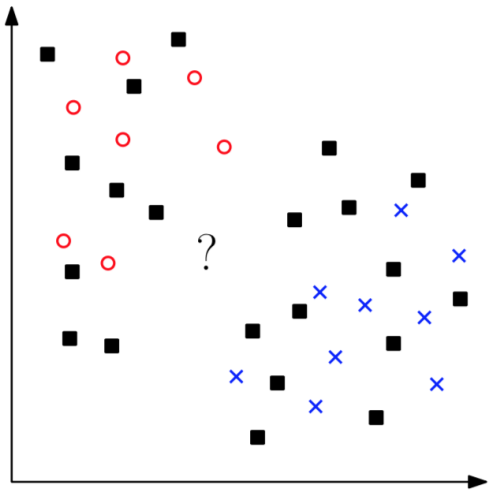


Figure 4.: Semi-supervised learning: The black boxes represent unlabeled data [50]

Semi-supervised learning could be classified into three groups, classification, clustering and regression.

Furthermore, common semi-supervised learning methods include generative models, semi-supervised support vector machines, graph Laplacian based methods, co-training, and multiview learning. These methods make different assumptions on the link between the unlabeled data distribution and the classification function. Such assumptions are equivalent to prior domain knowledge, and the success of semi-supervised learning depends to a large degree on the validity of the assumptions [13].

3.5 Reinforcement learning

Reinforcement learning [77] occurs when algorithms learn based on external feedback given either by a thinking entity or the environment.

Supervised or unsupervised learning methods learn to encode/predict/classify patterns, but they do not learn to act or make decisions. Reinforcement learning does this by:

1. Observing the real-time responses of the environment when random or non-optimal actions are taken
2. Learning, either implicitly or explicitly, the cost associated with the given state (and possibly the action).

For example, consider an ML algorithm that plays games against an opponent. Moves that lead to victories (positive feedback) in the game should be learned and repeated, whereas moves that lead to losses (negative feedback) are avoided [48].

3.6 DEEP LEARNING

Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Models are trained by using a large set of labeled data and neural network architectures that contain many layers. Most deep learning methods use neural network architectures, which is why deep learning models are often referred to as deep neural networks. Traditional neural networks only contain 2-3 hidden layers, while deep networks can have as many as 100 layers [68; 48].

3.6.1 *Shallow vs deep networks*

Deep networks can be distinguish from shallow by the number of hidden layers, shallow networks have one hidden layer that have nodes that perform a transformation of their inputs and sums them up, whereas deep networks have at least four or more and may have mixtures of types of layers (see figure 5) [48].

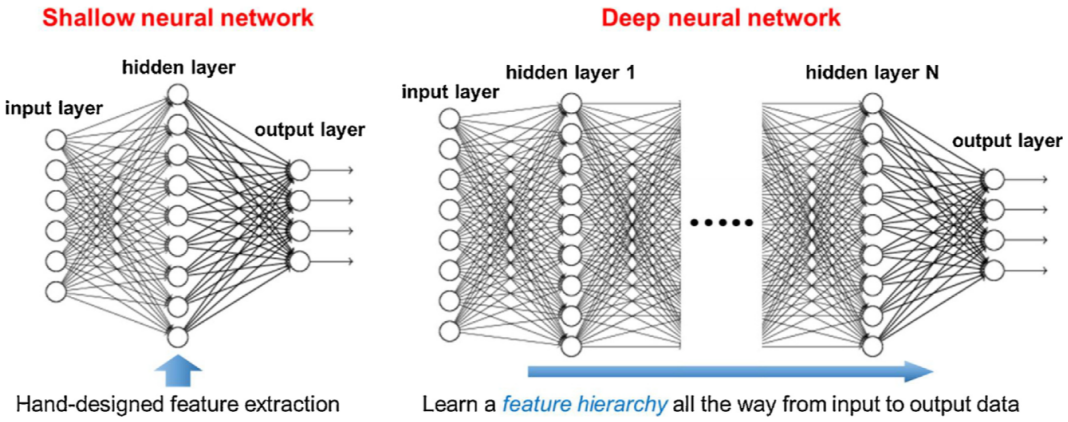


Figure 5.: Shallow vs deep networks [48]

3.7 MACHINE LEARNING ALGORITHMS

In this section we will be looking at some ML model types including popular algorithm implementations meant for classification. The types are chosen because they are frequently referenced in literature and cover a wide area of applications. We will outline the general properties of each model type and provide (illustrative) examples where necessary.

3.7.1 *Decision Trees*

Tree models, also known as decision trees, are a popular method of choice for modeling data. The classification problem is solved by asking a series of questions about the attributes of the observations. Each time an answer is received, a follow-up question is asked until a conclusion about the class of the observation's target attribute is reached. The series of questions and their possible answers are organised in the form of a decision tree. A decision tree is a hierarchical structure comprising nodes and directed edges. The tree has three types of nodes:

- Root node: node that has no incoming edges and two or more outgoing edges.
- Internal nodes: nodes that have exactly one incoming edge and two or more outgoing edges.
- Terminal nodes: nodes that have no outgoing edge

Each terminal node in a decision tree is assigned to a class label. Each non-terminal node (root node or internal nodes) contains attribute test conditions to divide observations that have different attributes. The classification of an observation from the test set is straightforward once the decision tree is constructed [47].

3.7.2 *Artificial Neural Networks*

Artificial neural network (ANN) is a mathematical model for predicting system performance (i.e., system output) inspired by the structure and function of human biological neural networks. The ANN is developed and derived to have a function similar to the human brain by memorizing and learning various tasks and behaving accordingly. It is trained to predict specific behavior and to remember that behavior in the future like the human brain does. Its architecture also is similar to human neuron layers in the brain as far as functionality and inter-neuron connection [1].
A ANN comprises simple processing units called as neurons, directed weighted connection between the neurons, and then mathematical function is applied to determine the activation of the neuron. Neuron takes input data and performs operations on it. The outcome

of these operations, called activation, is passed onto the other neurons. The desired output can be obtained by adjusting the weights. More the weight of the neuron, stronger will be the input that has to be multiplied by it. ANN are suitable for the problems in which the solution requires a knowledge which is difficult to specify but contains enough observations [85].

3.7.3 *Linear models*

A popular class of procedures for solving classification tasks are based on linear models. Linear models approach the learning task from a geometric perspective reasoning about data as points in a (Cartesian) coordinate system. Modeling techniques apply geometric concepts such as lines and planes (2-D surface) to structure points in coordinate spaces and in turn enable the classification of these points [71]. Logistic Regression [84], SVM [17] and Discriminant Analysis [78] are some popular implementations of the linear model type known for their high accuracy on classification problems.

A distinction can be made between Binary Logistic Regression and Multinomial Logistic Regression. The Binary Logistic Regression can predict a binary variable (0 or 1). The Multinomial Logistic Regression is an extension of Binary Logistic Regression. This model is able to predict more than two discrete categorical outcomes. They both use maximum likelihood estimation to find the parameters that best fit the data [20].

The formulation of Support Vector Machine (SVM) is proposed by Vapnik et al. in 1990s which is based on statistical learning theory [17]. Initially, SVM was developed to solve the two-class classification problem but later it was formulated and extended to solve multiclass classification problems [34]. SVM divides the data samples of two classes by determining a hyper-plane in input space that maximizes the separation between them.

3.7.4 *Probabilistic models*

Probabilistic classification models determine class membership as a function of the probabilities that specific feature values belong to a certain class [26], this is assessed by applying Bayes' theorem. In short Bayes' theorem describes how to determine the probability of an event given the conditions related to that event. This particular perspective on ML revolves around the reduction of uncertainty. Modeling initiates with maximum uncertainty about the prior probability of an instance belonging to a certain class. The act of learning is performed by adjusting the probability estimates after screening each instance and its class information, thus leading to reduced uncertainty about class membership [3].

The **Naive Bayes** modeling technique follows the principles as explained above. The algorithm relies on the assumption that features are independent of each other. For instance,

if we take features like temperature, humidity and wind speed to predict the rain we would assume that all those three features independently contribute to the probability of upcoming rain. Even if these features have some relation we would naively tell that they are not. This is one of the reasons why the algorithm is called "Naive".

3.7.5 *Clustering*

Clustering [38] refers to a very broad set of techniques for finding subgroups, or clusters. A clustering problem is where you want to discover the inherent groupings in the data.

For example, in a market segmentation study, we might observe multiple characteristics (variables) for potential customers, such as zip code, family income, and shopping habits. We might believe that the customers fall into different groups, such as big spenders versus low spenders. If the information about each customer's spending patterns is not available, we do not know whether each potential customer is a big spender or not. Therefore, we can try to cluster the customers based on the variables measured in order to identify distinct groups of potential customers. Identifying such groups can be of interest because it might be that the groups differ concerning some property of interest, such as spending habits.

There are several clustering approaches, in "A review of clustering techniques and developments" it is stated that the main reason is due to the fact that there is no such precise definition to the notion of "cluster" and that is why, different clustering approaches have been proposed, each of which uses a different inclusion principle [64]. Fraley and Raftery suggested dividing the clustering approaches into two different groups: hierarchical and partitioning techniques [27] (see Figure 3.7.5).

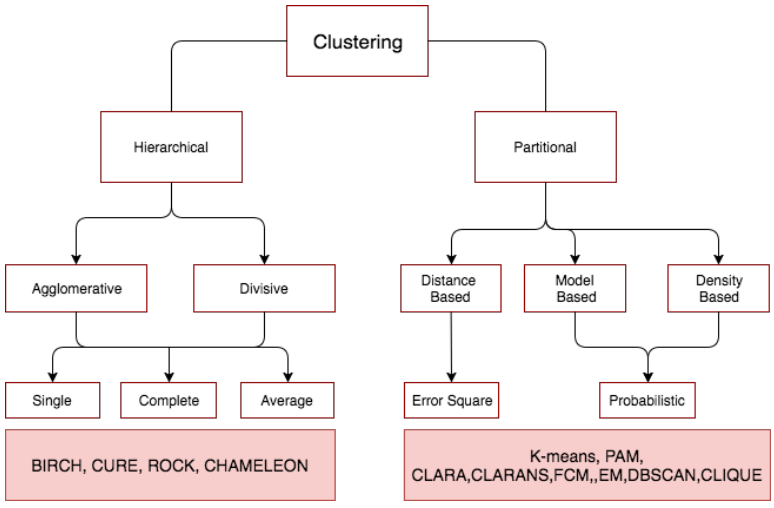


Figure 6.: Taxonomy of clustering approaches [27] [2]

3.7.6 *Rule-Based Machine Learning*

Rule-based Machine Learning is a term in computer science intended to encompass any ML method that identifies, learns, or evolves "rules" to store, manipulate or apply. The defining characteristic of a rule-based machine learner is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system [81].

Association Rules Learning

An association rule has the form $X \rightarrow Y$, where X and Y are item sets, which are subsets of I, the set of all items in the domain of investigation, consisting of a set of transactions.

In the standard ML terminology, transactions correspond to training examples (records in a database, see table 4), an item is a binary feature, and item sets are conjunctions of features. [43].

transaction ID	bread	beer	milk	butter
1	1	1	0	0
2	0	1	0	1
3	1	0	1	1
4	0	1	0	0

Table 4.: Example database with four transactions and four items

In order to illustrate the concept, let's use a small example from a supermarket domain. The set of items is $I = bread, beer, milk, butter$ and Table 4 shows a small database containing the items, where, in each entry the value 1 means the presence of the items in the corresponding transaction, and the value 0 represents not being present in that transaction.

An example rule for the supermarket could be:

$$\{milk, butter\} \Rightarrow \{bread\} \tag{1}$$

That means, if milk and butter are bought, costumers also buy milk. Association rule Learning finds all rules in the database that satisfy some minimum support and minimum confidence constraints.

2 This image was adapted from [64]

3.8 EVALUATION OF MACHINE LEARNING ALGORITHMS

A crucial part in ML is the problem of evaluating the performance of a ML model, which is an integral component of any data science project. Model evaluation aims to estimate the generalization accuracy of a model on future (unseen/out-of-sample) data. This section reviews the cross validation method and some ML evaluation metrics for classification problems, which is the main focus of this dissertation.

3.8.1 *Cross-validation*

The evaluation of classification tasks is normally done by splitting the data set into a training data set and a test data set. The ML algorithm is then trained on the first one, while the test data set is used to calculate performance indicators in order to evaluate the quality of the algorithm. A common problem for ML algorithms lies in the access to limited test and training data. Therefore, overfitting can be a serious problem when evaluating these programs. In order to address this problem, a common approach is, to use an *X-Fold Cross Validation*. Cross Validation [46] describes the process of splitting the whole data set into X parts and using each one of them sequentially as the test data set while combining the others to the training data. Cross-validation has been used widely in model selection, model/algorithm comparison and feature selection [11; 35].

3.8.2 *Metrics For Classification Model*

There is no perfect metric for every subject concerning evaluation of ML algorithms, since everyone has its flaws and advantages. The most important metrics for evaluating the performance of a ML program are the following:

- The **precision** value, also called positive prediction value, is defined as the relative amount of correctly as true classified instances among all as true classified instances [58].

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{2}$$

- The **recall** value, also called sensitivity is defined as the relative amount of as true classified instances among all true instances [58].

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{3}$$

- The **F-Measure**, also called F-score or F1 Score aims to combine the statements of recall and precision by using the harmonic mean between the two:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{4}$$

- The **confusion matrix**, also called contingency table, is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes. On the other hand, the main disadvantage of a confusion matrix is that it requires human interpretation [58].

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	True Positive	False Negative	P′
	n′	False Positive	True Negative	N′
	total	P	N	

- The **accuracy** in classification problems is the number of correct predictions made by the model over all kinds predictions made.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative} \tag{5}$$

Accuracy is a good measure when the target variable classes in the data are nearly balanced. But if the data is unbalanced, it should not be used. For example, let's consider that there are 98% samples of class A and 2% samples of class B in the training set. The model can easily get 98% training accuracy by simply predicting every training sample belonging to class A, and that can lead to a false sense of achieving high accuracy. The real problem arises, when the cost of misclassification of the minor class samples are very high. For example, if a data scientist deals with a rare but fatal disease, the cost of failing to diagnose the disease of a sick person is much higher than the cost of sending a healthy person to more tests.

- **AUC (Area Under The Curve) - ROC (Receiver Operating Characteristics)** curve, also written as AUROC (Area Under the Receiver Operating Characteristics), is one of the most important evaluation metrics for checking any classification model's performance. AUC - ROC curve is a performance measurement for classification problems at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability, it tells how much model is capable of distinguishing between classes [24]. Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. The better the classification algorithm is, the higher the area under the ROC curve. The score ranges from 0.5 to 1, and the score being 1 is the ideal case where True Positive Rate (TPR) is 1 and Flase Positive Rate (FPR) is 0, which means we correctly classify all positives and negatives.

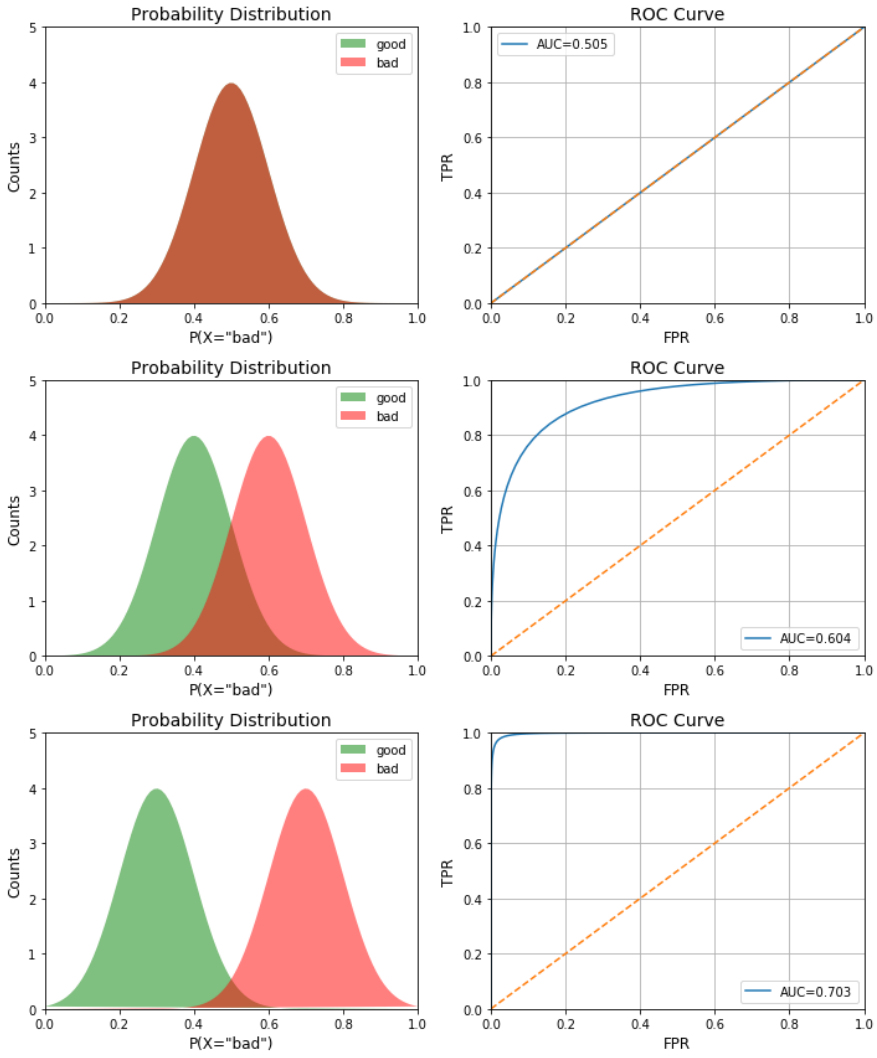


Figure 7.: Area Under the Receiver Operating Characteristics [54].

3.9 CONCLUDING REMARKS

In this chapter, the main concepts related with ML were introduced. Although the multiple variations of how to define the types of ML algorithms, in this dissertation we present the most common one, which divided into categories according to their purpose. The main presented categories are supervised learning, unsupervised Learning, semi-supervised Learning and reinforcement Learning. For each one of them popular algorithms and major properties were recalled and some literature was referenced.

We highlighted the large number of ML algorithms proposed in the literature and some metrics to evaluate classification algorithms were presented. Since many classifiers exist, all containing a number of parameters that potentially influence predictive performance, this is a challenging problem. Performing a cross-validation evaluation procedure on all possible combinations of classifiers and parameters (e.g.,using a grid search) is typically infeasible, as this would take too much time. In the next chapter, Meta-Learning as a solution to the algorithm selection problem is discussed.

4

META-LEARNING FOR ALGORITHM RECOMMENDATION

4.1 INTRODUCTORY REMARKS

In the previous chapter 2, it was discussed that user interactivity in the KDD process is, to some extent, necessary to achieve the required results. The main issue regarding data mining is a huge variety of different evolving techniques and methods. Therefore, it's practically impossible at the beginning of the analysis to know what is the ideal algorithm to achieve the desired results. Thus, the analysts must constantly reconfigure and alter the algorithms according to the obtained results. This is often very much a trial and error based procedure. To avoid these drawbacks, researchers have investigated the use of meta-learning to select the best recommendation algorithms in different scopes. Such studies allow understanding the relationships between data characteristics and the relative performance of different algorithms, which can be used to select the best algorithm(s) for a new problem.[19]

Meta-learning is concerned with discovering patterns in data and understanding the effect on the behavior of algorithms. It has been extensively explored for algorithm selection [61; 74]. Successful contributions can be seen in StatLog [45], METAL [7], and NOEMON [42] projects, which were large-scale European funded projects.

The algorithm selection task can be viewed as a learning problem itself, by casting it as a predictive task. For such, it uses a meta-dataset, where each meta-example corresponds to a dataset. For each meta-example, the predictive features are characteristics (meta-features) extracted from the corresponding dataset and the targets are the performance (meta-labels) of a set of algorithms when they are applied to the dataset.

According to [6], there are several basic applications of meta-learning:

- Selecting and recommending ML algorithms.
- Employing meta-learning in KDD.
- Employing meta-learning to combine base-level ML systems.
- Control of the learning process and bias management.

- Transfer of meta-knowledge across domains

Therefore, in this chapter, we will describe several representative contributions that are related to meta-learning in general. The list is eclectic and by no means exhaustive. Hence, in this chapter is discussed these contributions with an emphasis on the degree of relevance to the work presented in this dissertation.

4.2 META-LEARNING APPROACHES

Meta-Learning is indeed a rich field, usually explained as ”learning to learn”, different researchers hold different views of exactly what the term “meta-learning” means. In the novel *”A perspective view and survey of meta-learning”* has given a comprehensive review on the different perspectives on meta-learning. As a summary, will be listed some of the views and approaches to meta-learning [80].

4.2.1 *Meta-data based algorithm recommendation (ranking)*

In this procedure, a meta-dataset is created by using various features of a given dataset collection, like for example, the statistical information or the information-theoretic characteristics, such characteristics are termed as ”meta-features”. Another type of meta-feature is called ”landmarkers” [56], in which some properties (e.g., predictive performance) of a model learned by an algorithm are used as meta-features. To be efficient, the landmarker-based features need to be computed relatively quickly otherwise one could simply run the candidate algorithms on the dataset. Given a meta-dataset, another algorithm, usually called a metalearner, learns a model using the given meta-features. With a new dataset, firstly the meta-features are calculated, and the expected or relative predictive performances of different algorithms can be predicted.

4.2.2 *Ensemble Learning*

Ensemble methods are an ML technique that combines several base models in order to produce one optimal predictive model, and because of that can be viewed as a type of meta-learning. For instance, the stacking generalization method [82] works by combining several base-level learning algorithms and using a meta-level learner to learn a linear function of the models produced by the base-level algorithms. Given a new dataset, the predictions of the base-level algorithms are combined to provide the final prediction. Boosting [67] is another popular ensemble learning strategy in which the same base-level algorithm is used multiple times, wherein each boosting iteration, a re-weighted training set is used. The final

boosting prediction is also a combination of base-level models. Bootstrap aggregating also called bagging [8], is another example of ensemble learning wherein is used multiple weak models and aggregate the predictions from each of the models to get the final prediction. As suggested by the name, it consists of two parts, bootstrapping and aggregation. Extensive theoretical and empirical studies have been done in the previous years. Comprehensive reviews on ensemble techniques can be found on [69].

4.2.3 *Inductive Transfer*

In [80], the authors discussed the term inductive transfer as a variant of meta-learning. Inductive transfer refers to the ability of a learning mechanism to improve performance on the current or target task after having learned a different but related concept or skill on a previous source task. Transfer may additionally occur between two or more learning tasks that are being undertaken concurrently. This idea has been widely studied in multi-task learning [10].

4.3 THEORETICAL CONSIDERATIONS

Theoretical motivations as well as some arguments on meta-learning research is due to the No Free Lunch (NFL) theorem [83]. The basic idea is: *"When taken across all learning tasks, the generalization performance of any learner sums to o "*. In the context of ML, it is also known as a Law of Conservation for Generalization Performance (LGC) [65].

As a simple illustration of the NFL theorem, consider the simple space, χ, of binary functions defined over $\mathbb{B} = \{1, 0\}^3$ and assume that the instances of set $\mathcal{T}_r = \{000, 001, ..., 101\}$ are observed. The instances of set $\mathcal{T}_e = \mathbb{B}^3 - \mathcal{T}_r = \{110, 111\}$ constitute the off-training test set. The situation is shown in table 5.

	Inputs	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	...
Training Set	0 0 0	0	0	0	0	0	0	0	0	0	0	...
	0 0 1	0	0	0	0	0	0	0	0	0	0	
	0 1 0	0	0	0	0	0	0	0	0	0	0	
	0 1 1	0	0	0	0	0	0	0	0	0	0	
	1 0 0	0	0	0	0	0	0	0	0	1	1	
	1 0 1	0	0	0	0	1	1	1	1	0	0	
Test Set	1 1 0	0	0	1	1	0	0	1	1	0	1	...
	1 1 1	0	1	0	1	0	1	0	1	0	0	

Table 5.: Sample train/test setting for binary functions over 3 boolean variables (Adaption from [29])

The NFL theorem in this table shows that the behavior on $\mathcal{T}_e$ of any learner $(f1, f2, ..., f256)$ trained on $\mathcal{T}_r$ is that of a random guesser (can be seen, e.g., while considering functions f1 through f4 in table 5). The NFL theorem in essence simply restates Hume's conclusion about induction having no rational basis [36]: "There can be no demonstrative argument to prove, that those instances, of which we have had no experience, resemble those, of which we have experience..." The crucial contribution of the NFL theorem, is pointing out that whenever a learning algorithm performs well on some function it must perform poorly on some others.

Taking that into consideration NFL Theorem and Hume's conclusion, let's discuss the application of an Ultimate Learning Algorithm (ULA), therefore for a given learning algorithm, a model M is induced, which defines a class probability distribution p over the instances space. An ULA is a learning algorithm that induces a model M^*, such that:

$$\forall M' \neq M^* \quad E(\delta(p^*, p^\Omega)) \leq E(\delta(p', p^\Omega)) \tag{6}$$

where the expectation is computed for a given training/test set sample of the instance space, over the entire function space, δ is some appropriate distance measure and p^*, p^Ω and p' are class probability distributions. As pointed out in [29], here we are interested in an "ultimate" but not a "universal" algorithm because "universal" generally means either:

1. (mathematically) applicable independent of any assumptions, or
2. applicable throughout the entire universe.

When asking about the real world (our universe) it is the second definition that is important; what could happen in other conceivable universes is of no possible interest to us. As pointed out in multiple research papers and books [22; 29].

4.4 Algorithm Recommendation

The purpose of this dissertation focuses on the recommendation of a meta-learning approach for the problem of algorithm selection. Therefore, it's important to discuss the general ideas of meta-learning-based algorithm ranking and motivations. As discussed in the section 4.3, there is no single best algorithm to be used in all problems, theoretically due to the NFL theorem [83]. The brute-force approach is to try all the algorithms with different parameter settings (e.g., using a grid search) for a given dataset at hand. In practice this is usually not feasible due to the computation time and the enormous amount of alternative algorithms available as described in chapter 3.

A usual task of the data scientist normally involves the preparation of a dataset that can be processed by a model (ML algorithm). Usually, there are several algorithms available,

so the data scientist needs to select one of them for a business goal. Given the selected algorithm, the data scientist also needs to further fine tune the parameters in order to obtain a stable and accurate model. The choice of algorithm and parameter is guided by the performance estimation methodology that the analyst uses (see Chapter 2).

The most common practice is to use the ”trial and error” strategy. Taking algorithm selection as an example, the data scientist could get the performance estimations of the algorithms based on cross-validation, and then use performance measures to determine the best algorithm to use. Although feasible, this strategy may still require a reasonable amount of computing time, especially when there are many algorithms available. Therefore, the choice of which algorithm(s) to use depends on the dataset at hand, and systems that can provide such recommendations would be very useful [6]. Regarding to the meta-learning systems, an important discussion focus on ”which type of recommendation should be provided to the end-user?”, as described in [79] : ”...As the first recommended classifier is not always the correct choice, multiple recommendations should be made, making this a ranking problem rather than a classification problem....”.

In [6], the user’s goal is stated as: ”Save time by reducing the number of alternative algorithms tried out on a given problem with minimal loss in the quality of the results obtained when compared to the best possible ones.”. Meta-Learning for algorithm ranking uses a general ML approach to generate meta-knowledge mapping the characteristics of a dataset, captured by meta-features, to the relative performances of the available algorithms.

This approach is particularly important for business domains that require fast deployment of analytical techniques (e.g., stock market prediction and customer response prediction, etc...).

For a comprehensive review of meta-learning research and its applications, we refer the reader to [6; 80; 73; 76]

4.5 META-FEATURES

To be able to implement a meta-learning system, it is necessary to have the ability to characterize the source dataset. This can be resolved by using ”meta-features”. The idea is to gather descriptors about the data distribution that correlate well with the performance of learned models. Traditionally, meta-features are clustered in the following categories [6; 40]:

- features that describe the nature of attributes
- features that describe attributes
- features that describe relationships between attributes
- features that describe relationships between attributes and the target.

As known, the performance of meta-learning depends crucially on the quality of the meta-features available. In terms of the meta-features, existing meta-learning systems are mainly based on four types of meta-features:

- **Simple** - Number of instances, number of attributes, number of classes or targets [23]...
- **Statistical** - Mean kurtosis of attributes, mean skewness of attributes [23]...
- **Information-theoretic** - Class entropy, mean entropy of attributes, noise signal ratio [40]...
- **Landmarking** – Exploiting dataset properties from the performance of a set of simple and fast learners with significantly different learning mechanisms [56]. Various landmarking methods are evaluated in [28].

Thorough reviews and explanations of these meta-features are given in [5; 6; 40; 29].

4.6 CONCLUDING REMARKS

In this chapter, it was presented the research area of Meta-Learning as a solution for the algorithm selection problem. Examples of successful research projects were introduced with their main contributions to the literature in different domains. It was also explained the different categories of meta-features and their use as well as the added knowledge they provide to basic ML.

The literature indicates the use of Meta-Learning as one of the best-known approaches for the algorithm selection problem, however, most of the research papers usually indicate only the number of the selected features or the performance of a classifier designed using a feature subset of a specific size. Also, the available public meta-datasets are scarce and without a lot of important meta-features, algorithms, and results of several evaluation metrics.

Next, in the Second Part of this work, a novel framework for meta-dataset generation is presented, and a solution for the ML algorithm selection problem is proposed.

Part II

CONTRIBUTION

5

META-DATASET GENERATOR FOR MACHINE LEARNING ALGORITHMS RECOMMENDATION

In this chapter, we propose a framework for the automatic generation of meta-datasets in the context of Meta-Learning. The main motivation comes from the work described in the literature review, it was clear the lack of public meta-datasets, and most of the research papers usually only indicate the number of the selected features, or the performance of a classifier designed using a feature subset of a specific size, or the execution time. This framework aims to receive an arbitrary dataset, to extract the meta-features from that dataset and to store the evaluation results (e.g., AUC) of multiple available algorithms.

In this thesis, we focus on the performance of meta-learning for algorithm ranking on classification datasets only. Nevertheless, the solution presented is designed to be flexible, and can incorporate any type of meta-features and ML algorithms.

5.1 PROPOSED SOLUTION

Taking the dataset illustrated in Table 7 as an example, our approach, therefore, is twofold. First, it's necessary to extract the meta-features from the original dataset, then it's necessary to train and test the algorithms using the original dataset to have the algorithm performance results. Finally, we must concatenate the performance results to the meta-features already extracted. Given the above information, the proposal meta-dataset, is a $n \times m$ data matrix, where $m = mf + mt$. Here, m is the sum of the number of meta-features mf and the number of algorithm performance results mt, and n is the number of datasets.

dataset	mf1	mf2	mf3	algorithms	ae1	ae2	ae3
Iris	100	20	0.78	Random Forest	0.88	0.79	0.82
Iris	100	20	0.78	BernoulliNB	0.75	0.73	0.67
Iris	100	20	0.78	Ada Boost Classifier	0.90	0.80	0.85
Heart Disease	300	55	0.08	Random Forest	0.90	0.81	0.82
Heart Disease	300	55	0.08	BernoulliNB	0.87	0.83	0.83
Heart Disease	300	55	0.08	Ada Boost Classifier	0.92	0.84	0.83

Table 6.: Example of a meta-dataset with the sample of two datasets (Iris, Heart Disease), three meta-features (mf1,mf2,mf3), the algorithm used (e.g., BernulliNB) and the results of three different evaluation metrics (ae1, ae2, ae3).

The proposed meta dataset generator was instantiated in Python [62], more precisely in Python 3.6. Python brings enormous advantages for the developing of the proposed solution, highlighting the large collection of data science libraries. For the meta-features extraction the pymfe library [60] has been chosen. By making available a large set of meta-feature extraction functions, this package allow us to extract 108 meta-features from the datasets. Concerning the ML algorithms, it was used the library scikit-learn [55], due to the large range of ML algorithms and a consistent interface.

As mentioned in chapter 3, the ML field is vast, and to make the implementation practicable for the proposed solution, in this dissertation, we focus only on classification problems.

5.2 ARCHITECTURE

The architecture of the Meta-Dataset Generator is explained in figure 8 and consists of four main components. The ETL Dataset Module, Meta-feature Extractor Module, Machine Learning Module, and the Meta-Dataset Module. Each of these elements and their overall flow are explained next.

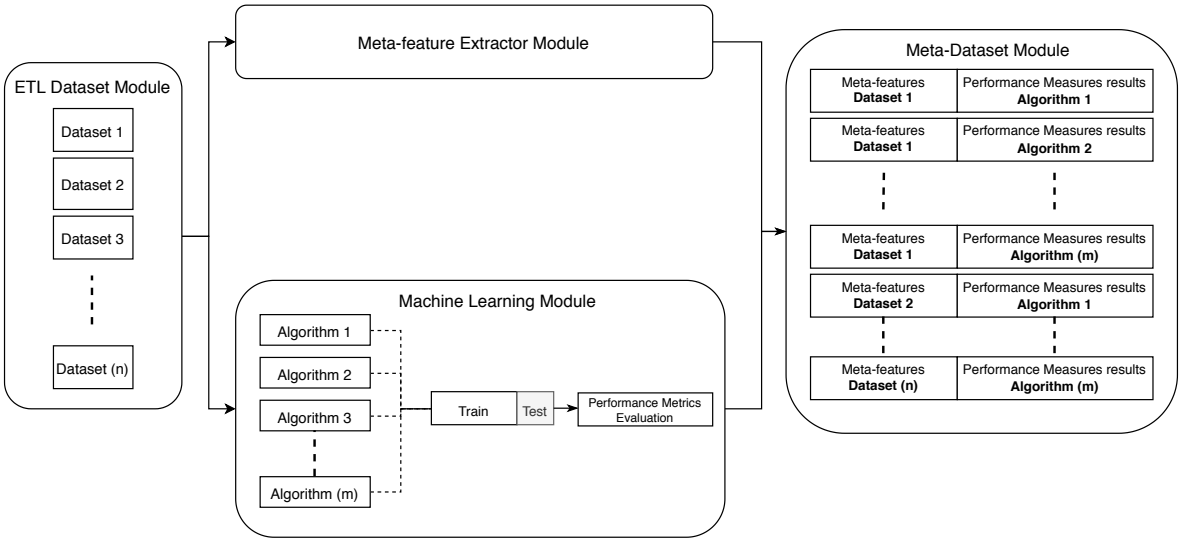


Figure 8.: Generic meta-dataset generator architecture

5.2.1 *ETL Dataset Module*

The ETL Dataset Module is responsible for handling the data from the original datasets. The main inspiration for this module comes from the traditional ETL (Extract, Transform and Load) process [44], and as the name implies, first the module extracts the data from the sources of the different datasets (e.g, MySQL, Excel, etc) then transform the data into a specific format and loads the treated datasets into a database.

The most relevant phase of this module is the transformation since it's necessary to provide a standardize dataset(s) to the Meta-feature Extractor Module and to the Machine Learning Module, the two main transformation steps are the following:

- The name of the target variable is changed to *y_target*, creating a standard separation between the target and the features.
- Since in this dissertation we propose only to work with classification problems, and to be able to draw statistically significant conclusions, in this module we also convert multiple-class classification dataset (by keeping the top two majority classes) and regression (by using the mean as a binary splitting point to transfer the numeric target to a binary target) datasets to binary classification datasets.

After the transformation, the treated datasets files are organized and stored in a specific folder.

This module was mainly developed in the KNIME Analytics Platform [2]. KNIME is an open-source data analytics, reporting, and integration tool. The use of this platform has

tremendously simplified the developing of this module. Nevertheless, the data extraction and the target name transformation continues to be a manual job. Figure 9 is an example of a macro KNIME Workflow developed in this dissertation.

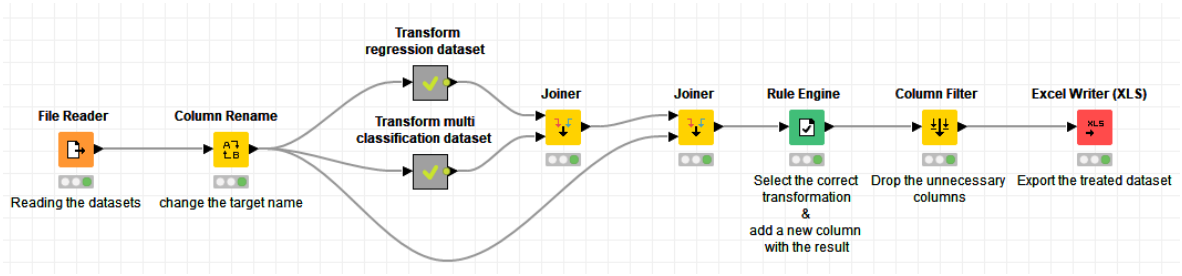


Figure 9.: Macro view from the KNIME Workflow

The last step is developed in Python, using the Pandas library to load the excel datasets as a Pandas DataFrame, to be then used by the adjacent modules.

5.2.2 *Meta Features Extractor Module*

As explained in the section 4.5, the performance of meta-learning depends crucially on the quality of the meta-features available. For that very reason, and to ensure an expressive characterization of the dataset, in this module we use the Python library pyfme [60] (version 0.1.0) due to the large set of meta-feature extraction functions, this library allow us to extract 108 meta-features from six different groups (General, Statistical, Information-theoretic, Model-based, Landmarking and Clustering). An example of the meta-feature extraction using pymfe is the following:

```
def meta_features_extractor(self, data_set_name):
    mfe = MFE(groups=["general", "statistical", "info-theory", "model-base
        ", "landmarking"])
    mfe.fit(self.X.values, self.y)
    ft = mfe.extract()
    meta_data = list(ft)
    headers = meta_data.pop(0)
    meta_data_pandas = pd.DataFrame(meta_data, columns=headers)
    meta_data_pandas['data_set_name'] = data_set_name
    return meta_data_pandas
```

Listing 5.1: Code sample of the Meta Feature Extractor Module

A complete overview of the meta-features used in this research project can be found in Appendix B Table 18 and in [60].

5.2.3 *Machine Learning Module*

The Machine Learning Module is responsible for applying ML algorithms over the dataset(s) provided by the ETL Dataset Module, and to store the results from the different metrics used to evaluate the algorithm performance.

As mentioned before, in this dissertation we focus only on the performance of meta-learning for algorithm ranking on classification datasets, and because of that, it was only implemented state-of-the-art supervised learning algorithms (explained in chapter 3). The implemented algorithms are the following:

Algorithm	Algorithm Group
Ada Boost Classifier	Ensemble Methods
Bagging Classifier	Ensemble Methods
Bernoulli Naive Bayes	Naive Bayes
Decision Tree Classifier	Decision Trees
Extra Tree Classifier	Decision Trees
Gaussian Naive Bayes	Naive Bayes
Gradient Boosting Classifier	Ensemble Methods
K Neighbors Classifier	Nearest Neighbors
Linear Discriminant Analysis	Discriminant Analysis
Logistic Regression (Binary and Multinomial)	Generalized Linear Models
Neural Networks (Multi-layer Perceptron)	Neural network models (supervised)
NuSVC	Support Vector Machines
Quadratic Discriminant Analysis	Discriminant Analysis
Random Forest Classifier	Ensemble Methods
Stochastic Gradient Descent	Generalized Linear Models
Support Vector Machines	Support Vector Machines
Voting Classifier	Ensemble Methods

Table 7.: Machine Learning algorithms implemented

The ML algorithms are developed using Scikit-Learn [55] Python package (version 0.18.1), and since many ML algorithms don't deal with categorical data, we use the One Hot Encoder (*get_dummy* function from scikit-learn) to convert categorical data to numerical data. An example of how the One Hot Encoder works can be found in the table below:

Color
red
orange
blue

red	orange	blue
1	0	0
0	1	0
0	0	1

Table 8.: Example of One Hot Enconding

The presented ML models were developed and evaluated using the 10-fold cross validation technique, and for some algorithms, the hyperparameter tuning was implemented using Grid Search (can be consulted in Appendix B Table 20). We also store the time required for the model fitting as a meta-feature (e.g., for future algorithms comparison analysis, algorithm selection decision criteria, etc...). The code below describes an example of a real implementation.

```
def gradient_boosting_classifier(self, hypertuning):
    gbc = GradientBoostingClassifier()
    if hypertuning:
        parameters = {'learning_rate': [0.01, 0.05, 0.1, 0.5, 1],
                      'min_samples_split': [2, 5, 10, 20],
                      'max_depth': [2, 3, 5, 10]}

        clf = GridSearchCV(gbc, parameters, cv=10)
        clf.fit(self.X,self.y)
        y_score = clf.predict(self.X)
        return metrics.average_precision_score(self.y, y_score)
    else:
        gdc_fit = gbc.fit(self.X, self.y)
        cv_10 = KFold(n_splits=10, random_state=457, shuffle=True)
        y_score = cross_val_predict(estimator=gdc_fit, X=self.X, y=self.y,
            cv=cv_10, method='predict_proba')
    evaluation = Evaluation(self.y, y_score)
    return evaluation.binary_classification_evaluation()
```

Listing 5.2: Gradient Boosting Classifier implementation example

The performance measure reported by 10-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a major advantage in problems such as inverse inference where the number of samples is very small.

Since we are only working with binary classification datasets the following performance metrics are applied and stored:

- AUC-ROC Score

- Precision
- Recall
- F1 Score
- Accuracy

5.2.4 *Meta-Dataset Module*

This module aggregates the information received from the Meta-feature Extractor Module with all the results from the ML Module. The final result is a matrix with 108 meta-features from the original data-set by all the algorithms (with all evaluation results and computation time). After the concatenation, the final meta-dataset is stored in an SQLite database.

5.3 EXPERIMENT SETUP AND RESULTS

In this section, the results obtained based on the proposed solution introduced in section 5.2 are presented. To be able to draw statistically significant conclusions, we choose a variety of datasets from various public data sources, including the UCI [1], Kaggle [2] and mldata [3] repositories, with a total of 34 datasets. All the ML algorithms presented in table 7 have been used over the 34 selected datasets.

The experiment was carried out using a MacBook Pro machine running on a 2.7 GHz dual-core Intel Core i5 processor.

Figure 10 shows some of the aggregated properties of the 34 datasets.

1 https://archive.ics.uci.edu/ml/
2 https://www.kaggle.com/datasets
3 https://www.mldata.io/datasets/

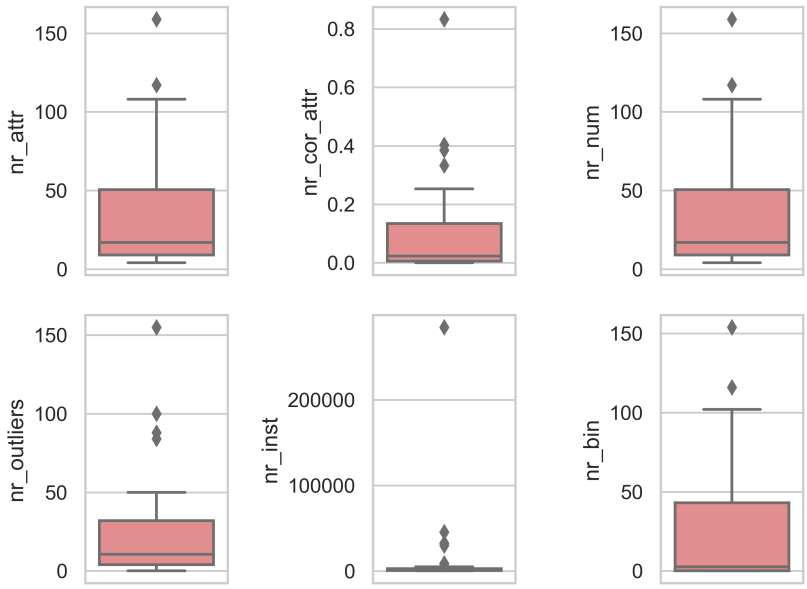


Figure 10.: Some aggregated meta-features of the 34 datasets showing the diversity of the meta-dataset[1]

5.3.1 *Datasets*

The datasets used in this research project came from various public dataset sources and different areas (e.g., business, advertising, medicine, etc), to bring a varied data sample. All the 34 datasets have been extracted as CVS files and cataloged as regression, multi-classification or binary classification, as illustrated in Table 9. The purpose of this catalog is related to the transformation by the ETL Database Module explained in 5.2.1, to ensure the transformation of all datasets into binary classification problems.

1 Number of Attributes (nr_attr), Number of attributes pairs with high correlation (nr_cor_attr), Number of numeric attributes (nr_num), Number of attributes with outliers values (nr_outliers), Number of instances (nr_inst), Number of binary attributes (nr_bin)

Dataset	Source	Type
Adult income	https://www.kaggle.com/uciml/adult-census-income	Binary Classification
Balance scale	https://www.mldata.io/dataset-details/balance_scale	Multiclass Classification
Balloons	https://archive.ics.uci.edu/ml/datasets/balloons	Binary Classification
Bank	https://www.kaggle.com/lovelesh/bank-dataset	Binary Classification
Bank marketing	https://www.mldata.io/dataset-details/bank_marketing	Binary Classification
Breast cancer	https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin	Binary Classification
Cars evaluation	https://archive.ics.uci.edu/ml/datasets/car+evaluation	Multiclass Classification
Cereals	https://www.kaggle.com/crawford/80-cereals	Regression
Chess king rook	https://www.mldata.io/dataset-details/chess_king_rook	Multiclass Classification
Congressional voting	https://www.mldata.io/dataset-details/congressional_voting	Binary Classification
Credit card fraud	https://www.kaggle.com/mlg-ulb/creditcardfraud	Binary Classification
Cryotherapy	https://www.mldata.io/dataset-details/cryotherapy	Binary Classification
Gender voice	https://www.mldata.io/dataset-details/gender_voice	Binary Classification
German credit data	https://www.mldata.io/dataset-details/german_credit_data	Binary Classification
Glass Identification	https://archive.ics.uci.edu/ml/datasets/Glass+Identification	Multiclass Classication
Heart disease	https://www.kaggle.com/ronitf/heart-disease-uci	Binary Classification
Horse Colic	http://archive.ics.uci.edu/ml/datasets/Horse+Colic	Binary Classification
Indian liver patient	https://www.kaggle.com/uciml/indian-liver-patient-records	Binary Classification
Iris species	https://www.kaggle.com/uciml/iris	Multiclass Classication
Mammogram	https://www.mldata.io/dataset-details/mammogram	Binary Classification
Monk	https://www.mldata.io/dataset-details/monk	Binary Classification
Mushroom	https://archive.ics.uci.edu/ml/datasets/Mushroom	Multiclass Classication
NBA logreg	https://data.world/exercises/logistic-regression-exercise-1	Binary Classification
Pima native american diabetes	https://www.mldata.io/dataset-details/pima_native_american_diabetes	Binary Classification
School grades	https://www.mldata.io/dataset-details/school_grades/	Regression
Soccer international	https://www.mldata.io/dataset-details/soccer_international_history	Multiclass Classification
Student Alcohol Consumption - Math Grades	https://archive.ics.uci.edu/ml/datasets/STUDENT+ALCOHOL+CONSUMPTION	Regression
Student Alcohol Consumption - Portuguese Grades	https://archive.ics.uci.edu/ml/datasets/STUDENT+ALCOHOL+CONSUMPTION	Regression
Tic tac toe	https://www.mldata.io/dataset-details/tic_tac_toe	Binary Classification
Titanic	https://www.kaggle.com/c/titanic	Binary Classification
Vehicle silhouette	https://www.mldata.io/dataset-details/vehicle_silhouette	Multiclass Classification
Wine quality (red)	https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality	Regression
Wine quality (white)	https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality	Regression
Zoo Animal	https://archive.ics.uci.edu/ml/datasets/Zoo	Multiclass Classification

Table 9.: Datasets used in this work

Afterward, we generate the 108 meta-features from the 34 datasets which took approximately 1 hour to compute.

5.3.2 *Algorithms analyses*

The class distributions of the 34 datasets vary a lot, with some datasets being very skewed, which can cause high variance on zero-one loss estimation. Consequently, for a fair algorithms comparison, we choose the area under the receiver operating characteristic curve (AUC) metric as the main performance measure, as it is less affected by class skew.

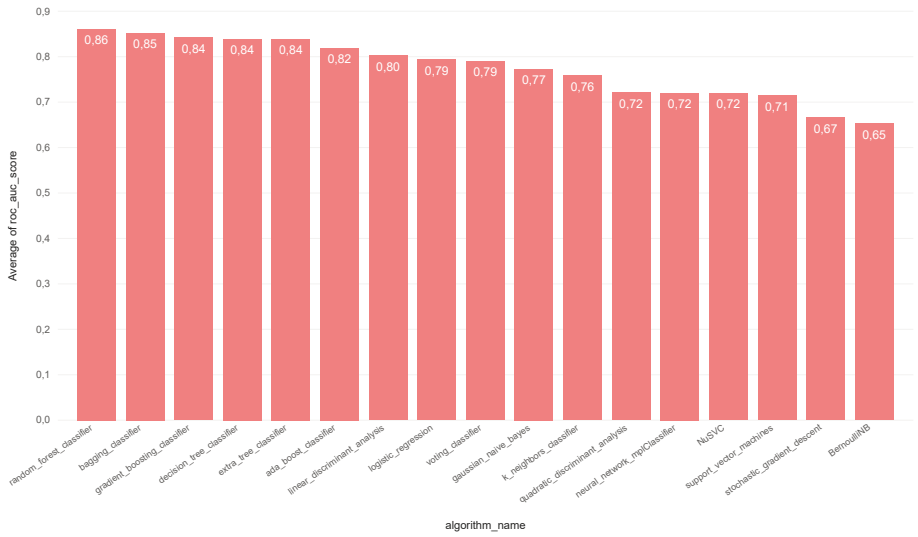


Figure 11.: Average AUC results per algorithm over the 34 datasets (without hyperparameter optimization)

For simplicity and in order to speed up experiments, it was used the default parameter settings from the Scikit Learn library as a first step, and we use 10-fold cross validation for the ranking generation which took $\approx$ 1 day to complete. Figure 11 shows the average AUC score result (from 0 to 1) over the 34 datasets, in which Random Forest outperformed with an average of 0.86.

Although the default parameter settings are a valid approach for fast results, in practice is a suboptimal solution, because to make useful predictions, most algorithms have to be optimized for each specific dataset. Technically, predicting the full combination of parameter settings is not feasible. Nevertheless, as explained in 5.2.3 for 12 algorithms we define some parameter settings (see Appendix B Table 20), and it was implemented using Grid search with 10 fold cross-validation based AUC scores for ranking generation. Building up this meta-dataset with the hyperparameter optimization was the most expensive part of our meta-learning experiment, it took roughly 37 days to complete.

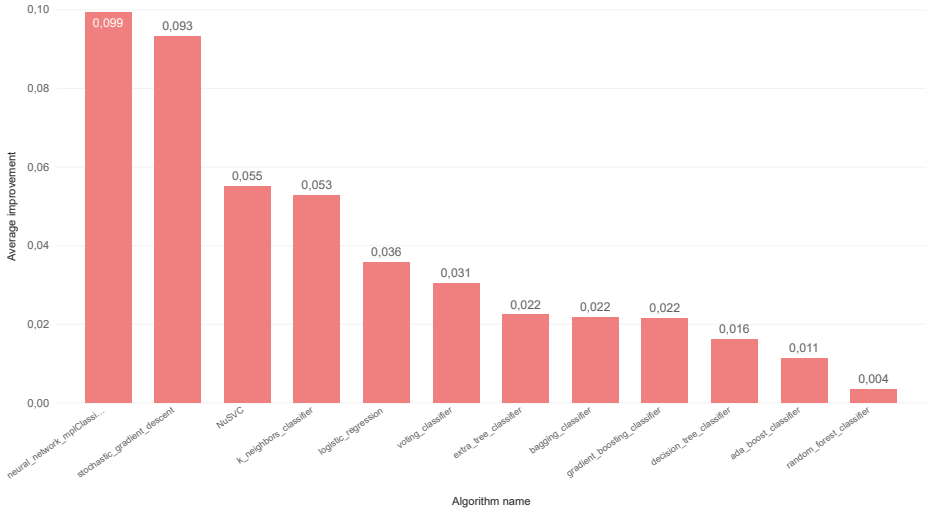


Figure 12.: Average improvement of the algorithm performance (AUC) using Gridsearch

Figure 12 shows the average percentage (from 0 to 1) of improvement of the best AUC score among the 12 classification machine algorithms for each dataset over the best AUC score of the same 12 algorithms using their default parameters. The result demonstrates the benefit of using the performances of optimised algorithms for generating algorithm rankings.

An optimized and non-optimized algorithm is also a piece of valid information to be stored into meta-dataset. Therefore, these optimized algorithms, have been saved into the meta-dataset as, for example, (optimized) Random_Forest.

A supplementary analysis can be done on the training and testing time per algorithm, which can be an important requirement for the algorithm selection problem [7]. It can be seen in figure 13 the total sum and the average of learning time by the 17 algorithms over the 34 datasets without hyperparameter optimization.

In our experiment, the support vector machines took exponential more time to fit and to predict than the other algorithms, in comparison with the lowest time consumer algorithm the support vector machines took more 9794 times longer. It was also interesting that most of the algorithms achieve satisfactory results in a relatively short amount of time.

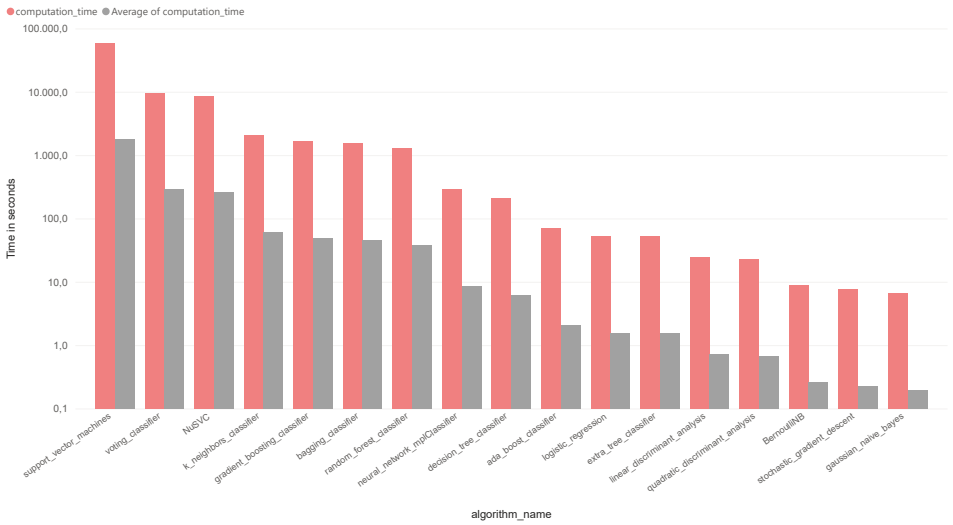


Figure 13.: Fit and predict computation time in seconds by algorithm (without hyperparameter optimization)

5.4 DISCUSSION

In this chapter, the overall proposed architecture for the meta-feature extractor, and the four modules comprising the architecture are described in detail. For the experiment, 108 meta-features from 34 datasets arising out of various public data sources (e.g., UCI) were extracted. Unlike previous work, experiments in this chapter were based on an unusually large number of meta-features and algorithms, to allow the applicability of the new techniques to a wide range of meta-learning problems, and consequently, the algorithm selection problem.

In terms of the implementation of the algorithms, our results show that the use of a 10-fold cross-validation estimator allows us to have a lower variance than a single hold-out set estimator, which can be very important since the amount of data available is limited. Our experimental results also indicate the use of hyperparameter tuning can increase significantly the algorithms performance.

On the other hand, we observe that the generation of the meta-dataset with hyperparameter optimization and the complete meta-features extraction took a huge amount of time and computer performance, in practice this is usually not feasible if there are too many alternative algorithms available and huge datasets. However, for this experiment, we wanted to extract the full potential of the meta-dataset generator to create a robust meta-dataset for further analyses.

6

MACHINE LEARNING ALGORITHM RECOMMENDATION

This chapter presents an approach to solve the dilemma given in this dissertation, the algorithm selection problem. To sum up, a system for algorithm recommendation can be defined as a tool that supports the user in the algorithm selection step of the data mining process. Given a dataset, it indicates which algorithm should be used to achieve the best possible results.

In this regard, it will be discussed and analyzed the application of data mining techniques to recommend the best algorithm for a specific classification dataset. A proposal that is studied in this dissertation is based on the meta-dataset obtained in the proposed framework introduced in Chapter 5. Throughout this Chapter, we apply the six-step KDP methodology (see 2.2.3) as structured approach for providing a blueprint for conducting this data mining project.

Hence, the case is solved according to the six phases of the KDP model given in figure 2.

6.1 UNDERSTANDING THE PROBLEM DOMAIN

6.1.1 *Background information*

One of the major challenges in many domains of Computational Intelligence, Machine Learning, Data Analysis, and other fields is to investigate the capabilities and limitations of the existing algorithms in order to identify when one algorithm is more adequate than another to solve a particular problem [41]. Traditional approaches to selecting algorithms involve, in general, costly trial-and-error procedures, or require expert knowledge, which is not always easy to acquire.

Therefore, the algorithm selection is the case under study in this dissertation, and since the problem is already presented extensively in the first chapter, as well as the motivation behind it, thus it shall not be debated any longer.

The six-step KDP model also requires the presentation of the advantages and disadvantages of current solutions. This task has already been done in the chapter dedicated to the Literature review.

6.1.2 *Research goals*

The main goal to be achieved with this project is to give a valid suggestion for which classification algorithm should be one of the most beneficial to use in a data science project. For that, the input must be the original dataset and the output the algorithm suggestion (or suggestions). The suggestion must take into consideration the algorithm performance and/or the learning time as the evaluation criteria. There are, however, other minor objectives that have to be achieved, namely to:

- Identify patterns in the meta-data that are relevant to the algorithm performance.
- Develop one or several ML models and choose the most relevant ones for the prediction of the algorithm selection.
- Develop other Data Mining models, towards the success in fulfilling the leading objective.

The KDP model states that for every objective a success criteria shall be defined. For the main goal of this dissertation to be a success, the algorithm prediction must achieve a reasonable performance result(i.e. AUC $>$ 80%). Also, valuable knowledge about the meta-data must be learned. This know-how might help to understand what meta-information is more relevant for the output prediction.

6.2 UNDERSTANDING OF THE DATA

Once the result of the project is well identified and understood, it's time to build an understanding of the data collected so that appropriate attributes and relationships between them can be selected for further use.

Since the data for this project has been collected using the meta-dataset generator from the previous chapter, he data analysis has already been done in the section 5.3.

Even though, as a summary, the meta-dataset is composed of 118 columns (see table 18), that includes:

- 108 meta-features from the original dataset
- The algorithm name (used to generate the results)
- 5 performance results (e.g., f1 score)

- The algorithm learning time
- A boolean for hyperparameter optimization identification
- The meta-dataset computation time
- The original dataset name

This meta-dataset matrix is composed of 34 different datasets and 17 ML algorithms (see chapter 5 for more detailed information).

6.3 PREPARATION OF THE DATA

As explained in subsection 2.2.3 the data preparation is probably the most important phase of the KDP model. It is the point where the data engineer finally starts to manipulate the data, preparing and packaging it for mining. Under normal circumstances, most of the project's time would be spent in preparing the data, but in the previous chapter, the data preparation has already started, and the thorough understanding of the problem domain and the data understanding stages performed earlier, have already minimized this overhead.

In what concerns to the output variable (also known as *y_target*) we have defined two approaches:

1. y_target1 = 1 or 0 (binary classification)
2. y_target2 = TOP 3 ranking algorithms and rest all are as *Others*

After deciding what is the output for our ML models to predict, it's necessary to transform the meta-dataset target into the desired values. As defined in the understanding stages, the output must take into consideration the algorithm performance and/or the learning time. For the algorithm performance we choose the attribute area under the receiver operating characteristic curve (AUC) as our output, because it is less affected by class skew.

Hence, by dataset we apply a descending ranking in the AUC value and an ascending ranking in the learning time as described in table 10.

Dataset name	Algorithm name	Computation time (s)	AUC score (%)	Rank AUC score	Rank time
Heart disease	BernoulliNB	0,04	81,86	1	6
Heart disease	Random Forest Classifier	3,52	81,83	2	16
Heart disease	Linear Discriminant Analysis	0,04	81,38	3	4
Heart disease	Logistic Regression	0,05	81,26	4	8
Heart disease	Gaussian Naive Bayes	0,04	80,59	5	1
Heart disease	Voting Classifier	3,73	80,22	6	17
Heart disease	Extra Tree Classifier	0,23	79,28	7	12
Heart disease	Quadratic Discriminant Analysis	0,04	79,01	8	2
Heart disease	Gradient Boosting Classifier	0,75	78,89	9	15
Heart disease	Bagging Classifier	0,24	78,59	10	13
Heart disease	Decision Tree Classifier	0,04	75,20	11	5
Heart disease	AdaBoost Classifier	0,06	72,93	12	9
Heart disease	K Neighbors Classifier	0,04	65,16	13	7
Heart disease	Neural Network MLP Classifier	0,47	64,86	14	14
Heart disease	Stochastic Gradient Descent	0,04	56,83	15	3
Heart disease	NuSVC	0,11	50,24	16	10
Heart disease	Support Vector Machines	0,22	49,39	17	11

Table 10.: Example of time and AUC ranking by 17 algorithms on the heart disease dataset

The next step consists in the definition of the relative importance of AUC score and time in order to have a unique output as defined above. In this regard, we create a new column which is the ascending ranking result of $\beta = 0.7 \times rank\_roc\_auc\_score + 0.3 \times rank\_time$. After the final ranking we have created three outputs according to the expectations above.

For the y_target1, we did the following approach:

$$y\_target_1(\beta) = \begin{cases} 1 & if\ \beta = 1 \\ 0 & if\ \beta \neq 1 \end{cases} \tag{7}$$

Hence, for the y_target2, we apply more and less the same approach as in as y_target1:

$$y\_target_2(\beta) = \begin{cases} \beta & if\ \beta <= 3 \\ Others & if\ \beta >= 3 \end{cases} \tag{8}$$

6.3.1 *Synthetic Minority Over-sampling Technique (SMOTE)*

Most ML algorithms perform better when the number of observations of each class of the target attribute is practically equal. The so-called class imbalance problem arises when the number of observations of one class far exceeds the other.

In Tables 11 and 12, we can observe that our datasets have imbalanced classes.

y_target2	# of Instances	Distribution (%)
1	91	15,74
0	487	84,26
total	578	100

Table 11.: Meta-dataset class distribution for y_target1

y_target2	# of Instances	Distribution (%)
1	91	15,74
2	36	6,23
3	32	5,54
Others	419	72,49
Total	578	100

Table 12.: Meta-dataset class distribution for y_target2

To deal with these imbalanced datasets, we apply a technique called Synthetic Minority Over-sampling Technique (SMOTE) [15]. SMOTE is a combination of under-sampling the majority class and over-sampling the minority class by creating "synthetic samples". To perform SMOTE, the library SMOTE from imbalanced-learn[1]is applied to the training datasets.

6.3.2 *Dimensionality Reduction*

Application of feature selection methods on the data containing many irrelevant features has become a necessity in many applications because many pattern recognition techniques cannot cope with high dimensional [31]. The selection of attributes is critically important because it can mean the difference between great performance with short training time and average performance with long training time.

In this dissertation, we have more than 100 columns in our meta-dataset (see Appendix B Table 19), and some variables might contain redundant and insignificant information for predicting outcome of the trainees. So, feature selection method has been applied with the help of the scikit-learn library. More specifically, the recursive feature elimination with cross-validation (RFECV)[2]as been used to select important attributes that contribute the most in the performance of the prediction.

1 The imbalanced-learn library can be found in [49]
2 The source code can be found in [70]

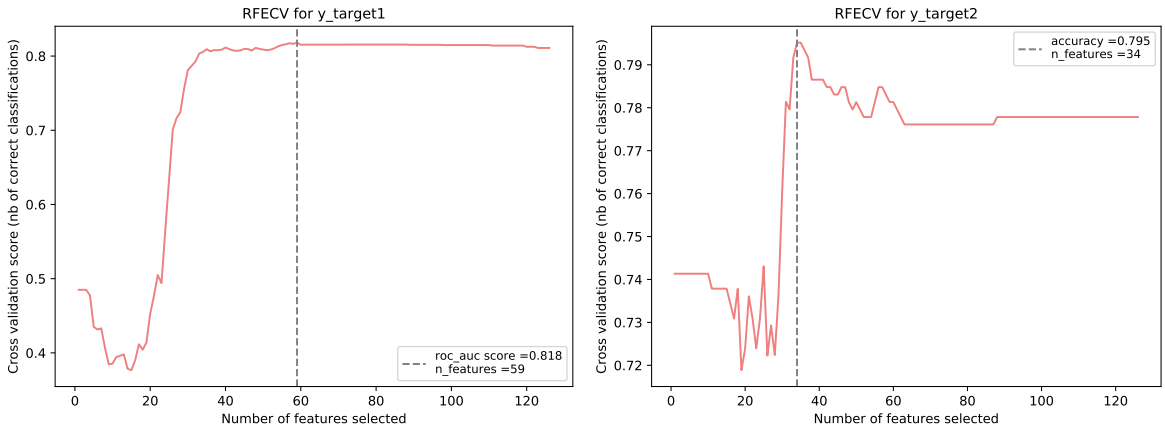


Figure 14.: Output of the RFECV for the meta-dataset with y_target1 and y_target2[1]

An overview of the selected attributes, including the ranking output from the RFECV, is given in Appendix B Table 21.

6.3.3 *Overview of the considered datasets*

Throughout this section, the changes done over the dataset have been described. Nonetheless, the way they were implemented was not referred to so far, therefore the purpose of this final subsection is to document those modifications and the created datasets. As stated several times in this document, all this modifications have been implemented in the training set, and the table 13 collects the considered datasets in an overview.

Dataset Name	Feature Selection	SMOTE	Target
meta-dataset_1	No	No	y_target1
meta-dataset_2	No	Yes	y_target1
meta-dataset_3	Yes	Yes	y_target1
meta-dataset_4	Yes	No	y_target1
meta-dataset_5	No	No	y_target2
meta-dataset_6	No	Yes	y_target2
meta-dataset_7	Yes	Yes	y_target2
meta-dataset_8	Yes	No	y_target2

Table 13.: Overview of 8 considered datasets

The following section briefly describes the algorithms which will be applied to the transformed meta-datasets.

1 Y-axis does not start at zero

6.4 DATA MINING

In the Data Preparation phase, described in section 6.3, several choices had to be made to prepare the data for the Modelling phase. These choices concerned for instance feature selection or over-sampling the dataset.

To investigate which choices are optimal, a total of 8 different training datasets are created. In the Modelling phase, the considered models are implemented.

6.4.1 *Selecting Modelling Techniques*

The selection of modelling techniques is preceded by a reflection about the data types available for mining, the Data Mining goals and the specific modelling requirements.

This phase demands investment of the time to discover and find models which suits best to the problem and to the data. Availability of the labeled data gives the ability to make use supervised learning techniques.

Chapter 4 describes the theory behind supervised learning and some families of algorithms are explained. Nevertheless, in chapter 5, 17 supervised learning algorithms have been implemented with and without hyperparameter optimization, these models have been selected based on their usefulness in solving ML problems and on their popularity.

Consequently, and since our use cases are classification problems, we took advantage of the work developed in the previous chapter and embrace all the 17 algorithms for our 8 meta-datasets. These algorithms are summarized in Table 7.

6.4.2 *Building the Models*

For building the models, as explained in 5.2.3, we use the 10-fold cross-validation technique, and for some algorithms, the hyperparameter tuning was implemented using Grid Search (see Appendix B Table 20 for more details). Although applying 10-fold cross-validation on our meta-datasets, for the ones with SMOTE, we have previously divided the dataset with the proportion of 70% for the training set (with SMOTE) and 30% for the test (original dataset).

In practice, the 8 training sets are fitted on the 17 different models. Hence, for every training set the corresponding test set is run through the 17 built models. At the end of the Modelling phase $17 \times 8 = 136$ prediction vectors of the target attributes are obtained.

6.5 EVALUATION OF THE DISCOVERED KNOWLEDGE

In the Data Mining phase, described in section 6.4, 136 prediction are made. In this section, these predictions are compared with the actual target attributes y_target1 and y_target2 in order to find out which data preparation steps and which classification model lead to the most accurate model. The most accurate model is defined as the model that predicts most correctly which algorithm or algorithms are the finest for a specific dataset. Recall that there are two different target attributes, y_target1 and y_target2, we divide the evaluation in two, the evaluation of the binary classifiers (for the y_target1) and the evaluation of the multi-class classifiers (for the y_target2).

6.5.1 *Evaluation of the Binary Classifiers*

To evaluate the performance of the models for the meta-datasets with the y_target1, the measures accuracy, precision, recall, F-measure and AUC-ROC are used. The Evaluation measures are theoretically explained in section 3.8.

The accuracy measure, although very popular in model evaluation, can be misleading when considering an imbalanced dataset. Consequently, we choose the AUC-ROC metric as the main performance measure, since is assumed to be a better measure when dealing with imbalanced binary datasets.

Figure 15 shows the aggregated performances of the 17 ML algorithms by meta-dataset. Clearly, the application of SMOTE and RFECV improved the overall performances. Based on the discussion in the preparation phase, this result is not surprising. The total results of the performance metrics of classification models are given in Appendix C Table 23.

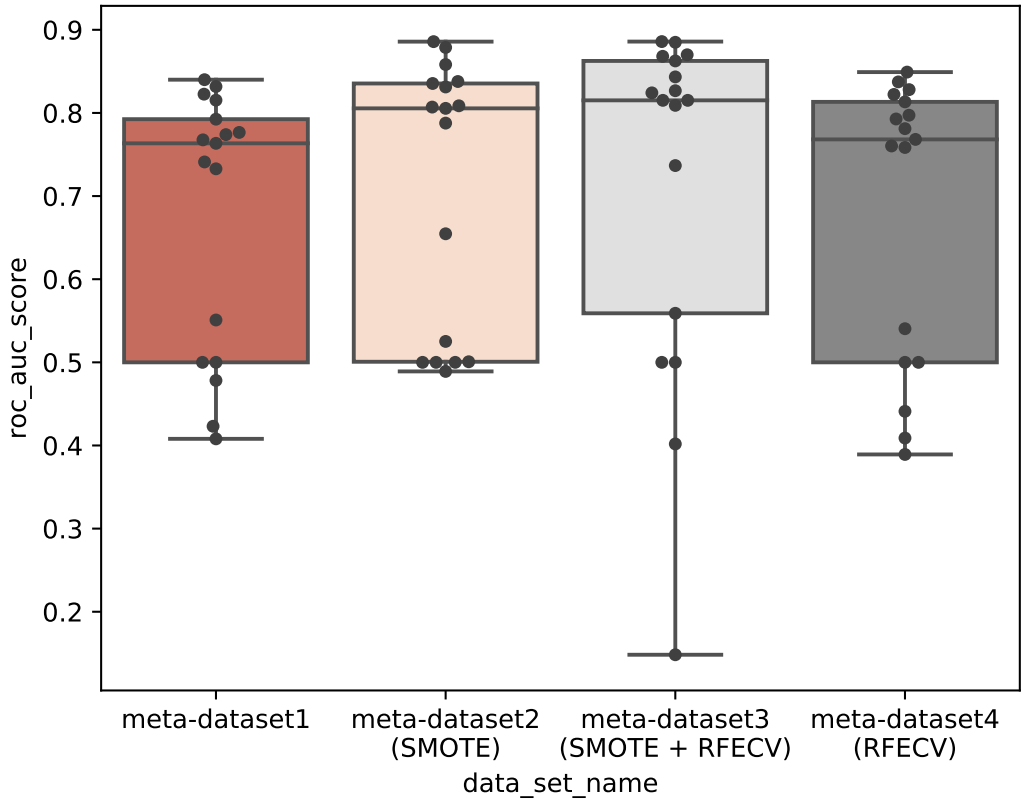


Figure 15.: Box plot with the meta-datasets results for the binary target[1]

Although the overall improvement with SMOTE and RFECV, for some algorithms the RFECV had an negative impact. Table 14 shows the top 5 worst performance results.

SMOTE	RFECV	Dataset name	Algorithm name	AUC score	Precision score	Recall score	F1 score
Yes	Yes	meta-dataset3	Support Vector Machine	0,1483	0,6959	0,7919	0,7408
No	Yes	meta-dataset4	Stochastic Gradient Descent	0,3892	0,7090	0,8420	0,7698
Yes	Yes	meta-dataset3	Gaussian Naive Bayes	0,4018	0,8656	0,2081	0,1346
No	No	meta-dataset1	Stochastic Gradient Descent	0,4080	0,7090	0,8420	0,7698
Yes	Yes	meta-dataset4	Support Vector Machine	0,4089	0,7090	0,8420	0,7698

Table 14.: Top 5 worst algorithms performances for y_target1

Gaussian Naive Bayes and Support Vector Machines were the most negatively affected by the feature extraction. A possible reason for that, is one of the most important strengths of these algorithms are the ability of working better with high dimensions [39].

On the other hand, table 15 shows the top 5 best results. The meta-datasets with SMOTE outperformed the other meta-datasets in every category. Moreover, feature selection also had a beneficial contribution to the best results.

1 Y-axis does not start at zero

SMOTE	RFECV	Dataset name	Algorithm name	AUC score	Precision score	Recall score	F1 score
Yes	No	meta-dataset2	BernoulliNB	0,8858	0,9354	0,9364	0,9358
Yes	Yes	meta-dataset3	Gradient Boosting Classifier	0,8858	0,929	0,9306	0,9296
Yes	Yes	meta-dataset3	Linear Discriminant Analysis	0,885	0,9409	0,9422	0,9393
Yes	No	meta-dataset2	Bagging Classifier	0,8788	0,9147	0,9191	0,9159
Yes	Yes	meta-dataset3	Logistic Regression	0,87	0,7025	0,8382	0,7644

Table 15.: Top 5 best algorithms performances for y_target1

Therefore, it can be agreed that based on our data, Bernoulli Naive Bayes and Gradient Boosting Classifier with 88.58 % (AUC performance) are the most accurate models for our binary classification target. Furthermore, other models like Linear Discriminant Analysis and Bagging Classifier also present very satisfactory results in all performance measures.

6.5.2 *Evaluation of the Multi-Class Classifiers*

In terms of the evaluation for the y_target2, the measures accuracy, precision, recall, F-measure are used. As explained in the previous subsection, accuracy is not a good measure if our class labels are not uniformly distributed. Which is the case, as exposed in table 12. F1 Score is the Harmonic Mean between precision and recall. The range for the F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances), and is assumed to be a better measure when dealing with imbalanced datasets. For that very reason, the F1 Score is the leading measure for the meta-datasets with the y_target2. F1 score and the other evaluation metrics are are theoretically explained in section 3.8. The complete results are given in table 16 Appendix C.

As we can see in Figure 16, the best results outcome from the meta-datasets with SMOTE. On the other hand, RFECV has not brought any benefit to the performances of the algorithms, and for the same cases, SMOTE worsened the results. Also, while evaluating the performance of the classifiers it is seen that the linear models (see table 16) has shown the lower overall performances than the ensemble methods (e.g., Decision Trees). One of the possible reasons behind this could be overfitting of the model. Overfitting is caused when the algorithm is heavily swayed by the training set. This can be mitigated by adding more data for training the model.

Table 17 shows the top 5 best results, and it's undeniable the benefits of the SMOTE on the training sets since all of the top 5 best results are the meta-datasets with SMOTE. In terms of performance results, ensemble methods have retrieved satisfactory results, with Bagging Classifier achieving 83.09 % on the F1 Score measure.

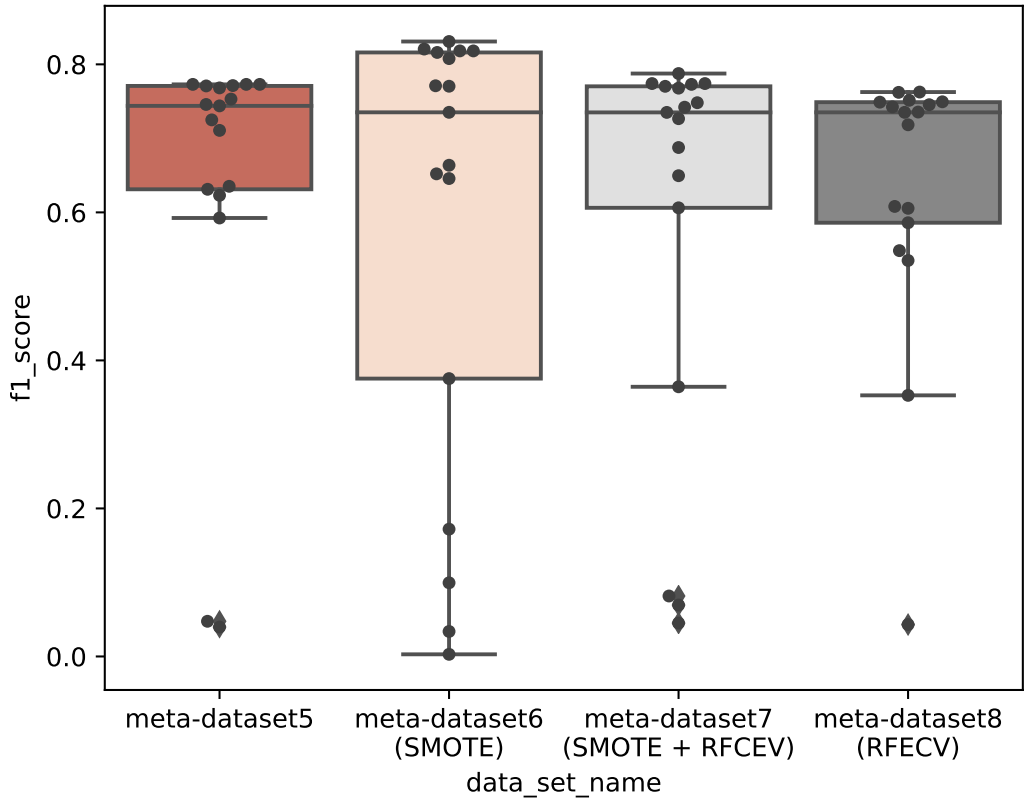


Figure 16.: Boxplot of the meta-datasets results for the multi-class target

SMOTE	RFECV	Dataset name	Algorithm name	Precision score	Recall score	F1 score
Yes	No	meta-dataset6	Stochastic Gradient Descent	0,0015	0,0347	0,0029
Yes	No	meta-dataset6	Logistic Regression	0,0192	0,1387	0,0338
No	No	meta-dataset5	Quadratic Discriminant Analysis	0,3928	0,1024	0,0397
No	Yes	meta-dataset8	Logistic Regression	0,0250	0,1580	0,0431
Yes	Yes	meta-dataset7	Logistic Regression	0,0262	0,1618	0,0451

Table 16.: Top 5 worst algorithms performances for y_target2

SMOTE	RFECV	Dataset name	Algorithm name	Precision score	Recall score	F1 score
Yes	No	meta-dataset6	Bagging Classifier	0,8344	0,8555	0,8309
Yes	No	meta-dataset6	Extra Tree Classifier	0,7894	0,8555	0,8207
Yes	No	meta-dataset6	Random Forest Classifier	0,7849	0,8555	0,8182
Yes	No	meta-dataset6	Voting Classifier	0,7849	0,8555	0,8182
Yes	No	meta-dataset6	K Neighbors Classifier	0,7857	0,8555	0,8161

Table 17.: Top 5 best algorithms performances for y_target2

6.6 DISCUSSION

In this chapter, we proposed a solution for the algorithm selection problem with a meta-learning approach. For conducting the research, the six-step KDP model has been used.

One of the main goals of this chapter was to identify if the work developed in the meta-dataset generator (Chapter 5) was not fruitless. Which was proved by very satisfactory results presented in section 6.5. Our experimental results also indicate the use of hyper-parameter tuning (with Grid Search), feature selection (with RFECV) and over-sampling techniques (with SMOTE) can increase significantly the algorithm performances. However, for some algorithms, SMOTE and RFECV have shown unfavorable results. One reason could be that the number of instances in the meta-dataset for algorithm ranking is too small (only 578 examples), which led to a diverse ranking model library becoming more likely to overfit.

Regarding the last phase of the used methodology, due to the time constraints of the dissertation, it was not possible to follow the developed models. Notwithstanding, the devised models can, in the future, be deployed as part of a recommendation system.

The techniques described in this chapter could probably also be applied to regression, but this needs to be verified in a future study.

7

CONCLUSIONS AND FUTURE DIRECTIONS

With the increase in the processor speed and memory size Machine Learning (ML) has become quite popular and a rich set of data analysis techniques, including algorithms and methods, have been developed in the ML and Data Mining (DM) community. However, no single algorithm is guaranteed to outperform every other one in every case.

This dissertation deals with the challenge of using Meta-Learning to automate the process of algorithm selection for classification problems. With this purpose, the work was divided in two main areas, the meta-dataset creation (Chapter 5) and ML algorithm recommendation (Chapter 6). In this process, regardless of the research limitations, some major contributions were derived.

7.1 RESEARCH LIMITATIONS

During the development of the dissertation, some research limitations were found. The lack of available public meta-datasets was one of the main drawbacks of this research project. Despite the work developed in Chapter 5, it was necessary to acquire several datasets from public data sources, which proved to be very time expensive. As a result of this manual work, the meta-dataset created was not really extensive.

Since many ML algorithms require large amounts of data before they begin to give useful results, this was a clear limitation to accomplish even better results (e.g, Neural Networks). Hence, if the values of meta-features of a new dataset are outside the ranges of the training meta-dataset, the developed recommendation system may fail because the target distribution is changed.

Besides the difficulty of acquiring datasets, the necessary computational time to generate the meta-dataset blocked the creation of a larger experiment, as an example, for our 34 datasets in a MacBook Pro machine running on a 2.7 GHz dual-core Intel Core i5 processor took approximately 38 days to complete.

7.2 MAJOR CONTRIBUTIONS

This research led to general contributions to the field of DM and ML. Next, we summarize the main contributions of this dissertation.

In Chapter 5 we proposed a framework for generating meta-datasets. The developed framework was implemented with Python, taking advantage of the multiple libraries that language offers, such as Sklearn (for the implementation of the algorithms) and Pymfe (for meta-feature extraction). It was then possible to implement 17 different ML algorithms and extract 108 meta-features from several public data sources. To store the algorithms performance, 5 different metrics have been implemented, resulting in a meta-dataset with 118 meta-features. This framework is one of the major contributions in this research project since it allows us to create a meta-dataset with a large number of meta-features, which was not available in the ML and DM community.

The developed framework was tested with 34 datasets from public libraries (such as UCI, Kaggle, etc.) and empirical analysis over the performance of the algorithms is presented. At this point, cross-validation results show a performance improvement by using parameter optimization.

In Chapter 6, we examine the feasibility of solving the dissertation main goal as a classification problem. We proposed two meta-features generation methods for meta-learning based algorithm ranking. One as binary classification, and the other as multiclass classification.

A variety of preprocessing and model construction strategies were evaluated throughout this work. One of the major challenge in the empirical study was the imbalanced meta-dataset, which resulted in poor performing models regardless of the used feature selection approach. The problem was solved by applying Synthetic Minority Over-sampling Technique (SMOTE). In both cases promising results were obtained when applied to the meta-dataset created in Chapter 5. Using 10-fold cross validation as the evaluation criteria, Bernoulli Naive Bayes was able to achieve an average AUC score of 88.5% (and 93.64% of F1 Score) on predicting the most suitable algorithm for the binary target. Moreover, for the multiclass target our results also showed auspicious results, with Bagging Classifier we have been able to achieve an average F1 Score of 83.09%.

Our experimental results are very satisfactory, showing the ability of computer systems to store virtually infinite amounts of prior learning experiences (in the form of meta-data) and to use that experience in completely new ways. In our use case, for the ML algorithms

recommendation. Potentially, the algorithms and methods proposed in this dissertation can be used as part of a DM recommendation system.

7.3 FUTURE WORK

In the context of the Meta-Dataset Generator (Chapter 5), different improvements can be considered in future work, such as increasing the number of meta-features, investigating the viability of using artificial datasets in order to generate a larger database of meta-examples and performing experiments with other ML algorithms. Moreover, different metrics than the ones that were employed in this dissertation must be investigated. These metrics may involve meta-features (requirements) from the project itself, such as the available computational time, RAM consumption, interpretability, etc...

Furthermore, the creation of the meta-data set is tremendously time and computationally expensive, the hyperparameter tuning with Grid Search proved to be not feasible in a reasonable amount of time. In this regard, another direction for future work is to investigate the implementation of Evolutionary Algorithms for parameter optimization instead of Grid Search, since some literature highlights the results from EA-based techniques are not significantly worse than those found with Grid Search, in a much shorter time[59].

Although the meta-dataset created in this dissertation was focused only on binary datasets, it would be interesting to apply the same approach to multiclass and regression datasets. For that, new evaluation metrics must be investigated accordingly.

Regarding the ML algorithm recommendation (Chapter 6), alternative data representation strategies and more sophisticated data filtering, over-sampling and under-sampling methods should also be considered to attempt to increase the predictive power of future models. Additionally, for further research, it might be interesting to add other automatic feature selection approaches. This dissertation uses the recursive feature elimination algorithm, but perhaps another wrapper method or even a filter method performs in a better way.

BIBLIOGRAPHY

[1] M. H. Bataineh. *Artificial neural network for studying human performance*. The University of Iowa, 2012.

[2] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, C. Sieb, K. Thiel, and B. Wiswedel. KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer, 2007.

[3] W. M. Bolstad and J. M. Curran. *Introduction to Bayesian statistics*. John Wiley & Sons, 2016.

[4] I. Bose and R. K. Mahapatra. Business data mining — a machine learning perspective. *Information Management*, 39(3):211 – 225, 2001.

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

[6] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning - Applications to Data Mining*. 01 2009.

[7] P. B. Brazdil, C. Soares, and J. P. da Costa. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, 50(3):251–277, Mar 2003.

[8] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[9] J. Brownlee. Supervised and unsupervised machine learning algorithms. https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/, 2016. Accessed: 2018-12-05.

[10] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[11] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 161–168, New York, NY, USA, 2006. ACM.

[12] M. Celebi and K. Aydin. *Unsupervised Learning Algorithms*. Springer International Publishing, 2016.

[13] O. Chapelle, B. Scholkopf, and A. Z. Eds. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Transactions on Neural Networks*, 20(3), March 2009.

[14] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. CRISP-DM 1.0 Step-by-step data mining guide. Technical report, The CRISP-DM consortium, Aug. 2000.

[15] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. Smote: Synthetic minority oversampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 01 2002.

[16] K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. *Data mining: a knowledge discovery approach*. Springer Science & Business Media, 2007.

[17] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[18] Q. Cui, F.-S. Bai, B. Gao, and T.-Y. Liu. Global optimization for advertisement selection in sponsored search. *Journal of Computer Science and Technology*, 30(2):295–310, Mar 2015.

[19] T. Cunha, C. Soares, and A. C. de Carvalho. Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering. *Information Sciences*, 423:128 – 144, 2018.

[20] S. A. Czepiel. Maximum likelihood estimation of logistic regression models: theory and implementation. *Available at czep. net/stat/mlelr. pdf*, pages 1825252548–1564645290, 2002.

[21] A. Dey. Machine learning algorithms: a review. *International Journal of Computer Science and Information Technologies*, 7(3):1174–1179, 2016.

[22] P. Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. Basic Books, Inc., New York, NY, USA, 2018.

[23] R. Engels and C. Theusinger. Using a data metric for preprocessing advice for data mining applications. In *ECAI*, volume 98, pages 23–28, 1998.

[24] T. Fawcett. Introduction to roc analysis. *Pattern Recognition Letters*, 27:861–874, 06 2006.

[25] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, et al. Knowledge discovery and data mining: Towards a unifying framework. In *KDD*, volume 96, pages 82–88, 1996.

[26] P. Flach. *Machine learning: the art and science of algorithms that make sense of data*. Cambridge University Press, 2012.

[27] C. Fraley and A. E. Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The Computer Journal*, 41:578–588, 1998.

[28] J. Fürnkranz and J. Petrak. An evaluation of landmarking variants. In *Working Notes of the ECML/PKDD 2000 Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning*, pages 57–68, 2001.

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

[30] C. Giraud-Carrier and O. Povel. Characterising data mining software. *Intell. Data Anal.*, 7(3):181–192, Aug. 2003.

[31] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182, Mar. 2003.

[32] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, Jan 2002.

[33] J. Han, J. Pei, and M. Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[34] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002. cited By 4362.

[35] J. Hua, W. D. Tembe, and E. R. Dougherty. Performance of feature-selection methods in the classification of high-dimension data. *Pattern Recogn.*, 42(3):409–424, Mar. 2009.

[36] D. Hume. *A treatise of human nature*. Courier Corporation, 2003.

[37] Intel. How to get started as a developer in ai. https://software.intel.com/en-us/articles/how-to-get-started-as-a-developer-in-ai, 2016. Accessed: 2018-11-30.

[38] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[39] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[40] A. Kalousis. *Algorithm selection via meta-learning*. PhD thesis, University of Geneva, 2002.

[41] A. Kalousis, J. Gama, and M. Hilario. On data and algorithms: Understanding inductive performance. *Machine learning*, 54(3):275–312, 2004.

[42] A. Kalousis and T. Theoharis. Noemon: Design, implementation and performance results of an intelligent assistant for classifier selection. *Intelligent Data Analysis*, 3(5):319–337, 1999.

[43] B. Kavšek and N. Lavrač. Apriori-sd: Adapting association rule learning to subgroup discovery. *Applied Artificial Intelligence*, 20(7):543–583, 2006.

[44] R. Kimball and J. Caserta. *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data*. John Wiley & Sons, Inc., USA, 2004.

[45] R. King, C. Feng, and A. Sutherl. Statlog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9, 11 2000.

[46] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[47] S. Kotsiantis. Supervised machine learning: A review of classification techniques. 31, 10 2007.

[48] J. H. Lee, J. Shin, and M. J. Realff. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers Chemical Engineering*, 2017.

[49] G. Lemaitre. over-sampling using smote. https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html.

[50] Lu, Tyler (Tian). Fundamental limitations of semi-supervised learning. Master's thesis, University of Waterloo, 2009.

[51] H. Lv and H. Tang. Machine learning methods and their application research. In *2011 2nd International Symposium on Intelligence Information Processing and Trusted Computing*, pages 108–110, Oct 2011.

[52] S. Marsland. *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2014.

[53] H. H. Martens. Two notes on machine "learning". *Information and Control*, 2(4):364 – 379, 1959.

[54] S. S. Nazrul. Receiver operating characteristic curves demystified (in python). https://towardsdatascience.com/

receiver-operating-characteristic-curves-demystified-in-python-bd531a4364d0, 2018.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[56] B. Pfahringer, H. Bensusan, and C. G. Giraud-Carrier. Meta-learning by landmarking various learning algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 743–750, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[57] I. Portugal, P. Alencar, and D. Cowan. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97(Supplement C):205 – 227, 2018.

[58] D. Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. *Mach. Learn. Technol.*, 2, 01 2008.

[59] M. Reif, F. Shafait, and A. Dengel. Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning*, 87:357–380, 06 2012.

[60] A. Rivolli, L. P. Garcia, C. Soares, J. Vanschoren, and A. C. de Carvalho. Towards reproducible empirical research in meta-learning. *arXiv preprint arXiv:1808.10406*, 2018.

[61] A. L. D. Rossi, A. C. P. de Leon Ferreira de Carvalho, C. Soares, and B. F. de Souza. Metastream: A meta-learning based method for periodic algorithm selection in time-changing data. *Neurocomputing*, 127:52 – 64, 2014. Advances in Intelligent Systems.

[62] G. Rossum. Python reference manual. Technical report, Amsterdam, The Netherlands, The Netherlands, 1995.

[63] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[64] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin. A review of clustering techniques and developments. *Neurocomputing*, 267(Supplement C):664 – 681, 2017.

[65] C. Schaffer. A conservation law for generalization performance. In *Machine Learning Proceedings 1994*, pages 259–265. Elsevier, 1994.

[66] R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear estimation and classification*, pages 149–171. Springer, 2003.

[67] R. E. Schapire and Y. Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[68] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117, 2015.

[69] F. Schwenker. Ensemble methods: Foundations and algorithms [book review]. *Computational Intelligence Magazine, IEEE*, 8:77–79, 02 2013.

[70] Scikit-Learn. scikit-learn recursive feature elimination (rfe). https://github.com/scikit-learn/scikit-learn/blob/7e85a6d1f/sklearn/feature_selection/_rfe.py#L37.

[71] S. R. Searle and M. H. Gruber. *Linear models*. John Wiley & Sons, 2016.

[72] G. A. Seber and A. J. Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[73] K. A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.*, 41(1):6:1–6:25, Jan. 2009.

[74] C. Soares, P. B. Brazdil, and P. Kuba. A meta-learning method to select the kernel width in support vector regression. *Machine Learning*, 54(3):195–209, Mar 2004.

[75] M. J. Stefik. Machine learning: An artificial intelligence approach: R.s. michalski, j.g. carbonell and t.m. mitchell, (tioga, palo alto, ca). *Artificial Intelligence*, 25(2):236 – 238, 1985.

[76] Q. Sun. *Meta-learning and the full model selection problem*. PhD thesis, University of Waikato, 2014.

[77] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[78] A. Tharwat. Linear vs. quadratic discriminant analysis classifier: a tutorial. *International Journal of Applied Pattern Recognition*, 3(2):145–180, 2016.

[79] J. van Rijn, S. Abdulrahman, P. Brazdil, and J. Vanschoren. Fast algorithm selection using learning curves. volume 9385, pages 298–309, 10 2015.

[80] R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, Jun 2002.

[81] S. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. 3, 11 1995.

[82] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241 – 259, 1992.

[83] D. H. Wolpert, W. G. Macready, et al. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[84] R. E. Wright. Logistic regression. 1995.

[85] G. Zhang, B. E. Patuwo, and M. Y. Hu. Forecasting with artificial neural networks:: The state of the art. *International journal of forecasting*, 14(1):35–62, 1998.

A

GLOSSARY OF TERMS, ABBREVIATIONS AND ACRONYMS

AI Artificial Intelligence

ANN Artificial Neural Networks

AUC Area Under The Curve

AUROC Area Under the Receiver Operating Characteristics

CV Cross Validation

DM Data Mining

EA Evolutionary Algorithms

ETL Extract Transform and Load

KDP Knowledge Discovery Process

KDD Knowledge Discovery from Data

ML Machine Learning

RFECV Recursive Feature Elimination with Cross-Validation

ROC Receiver Operating Characteristic

ULA Ultimate Learning Algorithm

SMOTE Synthetic Minority Over-sampling Technique

SVM Support Vector Machines

B

SUPPORT MATERIAL

1	attr_conc.mean	41	leaves_corrob.sd	81	one_nn.mean
2	attr_conc.sd	42	leaves_homo.mean	82	one_nn.sd
3	attr_ent.mean	43	leaves_homo.sd	83	random_node.mean
4	attr_ent.sd	44	leaves_per_class.mean	84	random_node.sd
5	attr_to_inst	45	leaves_per_class.sd	85	range.mean
6	best_node.mean	46	linear_discr.mean	86	range.sd
7	best_node.sd	47	linear_discr.sd	87	sd.mean
8	can_cor.mean	48	mad.mean	88	sd.sd
9	can_cor.sd	49	mad.sd	89	sd_ratio
10	cat_to_num	50	max.mean	90	skewness.mean
11	class_conc.mean	51	max.sd	91	skewness.sd
12	class_conc.sd	52	mean.mean	92	sparsity.mean
13	class_ent	53	mean.sd	93	sparsity.sd
14	cor.mean	54	median.mean	94	t_mean.mean
15	cor.sd	55	median.sd	95	t_mean.sd
16	cov.mean	56	min.mean	96	tree_depth.mean
17	cov.sd	57	min.sd	97	tree_depth.sd
18	eigenvalues.mean	58	mut_inf.mean	98	tree_imbalance.mean
19	eigenvalues.sd	59	mut_inf.sd	99	tree_imbalance.sd
20	elite_nn.mean	60	naive_bayes.mean	100	tree_shape.mean
21	elite_nn.sd	61	naive_bayes.sd	101	tree_shape.sd
22	eq_num_attr	62	nodes	102	var.mean
23	freq_class.mean	63	nodes_per_attr	103	var.sd
24	freq_class.sd	64	nodes_per_inst	104	var_importance.mean
25	g_mean.mean	65	nodes_per_level.mean	105	var_importance.sd
26	g_mean.sd	66	nodes_per_level.sd	106	w_lambda
27	gravity	67	nodes_repeated.mean	107	worst_node.mean
28	h_mean.mean	68	nodes_repeated.sd	108	worst_node.sd
29	h_mean.sd	69	nr_attr	109	meta_dataset_computation_time
30	inst_to_attr	70	nr_bin	110	data_set_name
31	iq_range.mean	71	nr_cat	111	hypertunning
32	iq_range.sd	72	nr_class	112	algorithm_name
33	joint_ent.mean	73	nr_cor_attr	113	computation_time
34	joint_ent.sd	74	nr_disc	114	roc_auc_score
35	kurtosis.mean	75	nr_inst	115	precision_score
36	kurtosis.sd	76	nr_norm	116	f1_score
37	leaves	77	nr_num	117	recall_score
38	leaves_branch.mean	78	nr_outliers	118	accuracy_score
39	leaves_branch.sd	79	ns_ratio		
40	leaves_corrob.mean	80	num_to_cat		

Table 18.: List of attributes from the meta-dataset generator

1	attr_conc.mean	41	leaves_corrob.sd	81	one_nn.mean
2	attr_conc.sd	42	leaves_homo.mean	82	one_nn.sd
3	attr_ent.mean	43	leaves_homo.sd	83	random_node.mean
4	attr_ent.sd	44	leaves_per_class.mean	84	random_node.sd
5	attr_to_inst	45	leaves_per_class.sd	85	range.mean
6	best_node.mean	46	linear_discr.mean	86	range.sd
7	best_node.sd	47	linear_discr.sd	87	sd.mean
8	can_cor.mean	48	mad.mean	88	sd.sd
9	can_cor.sd	49	mad.sd	89	sd_ratio
10	cat_to_num	50	max.mean	90	skewness.mean
11	class_conc.mean	51	max.sd	91	skewness.sd
12	class_conc.sd	52	mean.mean	92	sparsity.mean
13	class_ent	53	mean.sd	93	sparsity.sd
14	cor.mean	54	median.mean	94	t_mean.mean
15	cor.sd	55	median.sd	95	t_mean.sd
16	cov.mean	56	min.mean	96	tree_depth.mean
17	cov.sd	57	min.sd	97	tree_depth.sd
18	eigenvalues.mean	58	mut_inf.mean	98	tree_imbalance.mean
19	eigenvalues.sd	59	mut_inf.sd	99	tree_imbalance.sd
20	elite_nn.mean	60	naive_bayes.mean	100	tree_shape.mean
21	elite_nn.sd	61	naive_bayes.sd	101	tree_shape.sd
22	eq_num_attr	62	nodes	102	var.mean
23	freq_class.mean	63	nodes_per_attr	103	var.sd
24	freq_class.sd	64	nodes_per_inst	104	var_importance.mean
25	g_mean.mean	65	nodes_per_level.mean	105	var_importance.sd
26	g_mean.sd	66	nodes_per_level.sd	106	w_lambda
27	gravity	67	nodes_repeated.mean	107	worst_node.mean
28	h_mean.mean	68	nodes_repeated.sd	108	worst_node.sd
29	h_mean.sd	69	nr_attr	109	meta_dataset_computation_time
30	inst_to_attr	70	nr_bin	110	algorithm_name
31	iq_range.mean	71	nr_cat	111	y_target
32	iq_range.sd	72	nr_class		
33	joint_ent.mean	73	nr_cor_attr		
34	joint_ent.sd	74	nr_disc		
35	kurtosis.mean	75	nr_inst		
36	kurtosis.sd	76	nr_norm		
37	leaves	77	nr_num		
38	leaves_branch.mean	78	nr_outliers		
39	leaves_branch.sd	79	ns_ratio		
40	leaves_corrob.mean	80	num_to_cat		

Table 19.: List of attributes after the data preparation phase

Algorithm Name	Hyperparameter Optimization
NuSVC	param_grid = {'gamma':[1, 0.1, 0.001, 0.0001], 'kernel':['linear','rbf']}
LogisticRegression	param_grid = { "C": np.logspace(-3, 3, 7), "penalty": ['l1','l2'], "max_iter":[10000]}
ExtraTreesClassifier	param_grid = {,'n_estimators': [200, 500, 1000], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth': [4, 5, 6, 7, 8], 'criterion': ['gini', 'entropy']}
VotingClassifier	param_grid= {'rf__n_estimators': [20,100, 200,500,1000]}
BaggingClassifier	param_grid = {,'bootstrap': [True, False], 'bootstrap_features': [True, False], 'n_estimators': [5, 10], 'max_samples': [0.6, 0.8], 'base_estimator__bootstrap': [True, False], 'base_estimator__n_estimators': [100, 200], 'base_estimator__max_features': [0.8, 1.0]}
AdaBoostClassifier with DecisionTreeClassifier	param_grid = {'n_estimators': [50, 500, 1000, 2000], 'learning_rate': [.001, 0.01, .1]}
MLPClassifier	param_grid = {'solver': ['lbfgs'], 'max_iter': [1000, 1200, 1600, 1800, 2000], 'alpha': 10.0 ** -np.arange(1, 10), 'hidden_layer_sizes': np.arange(10, 15), 'random_state': [0, 1, 2, 4, 6, 8]}
GradientBoostingClassifier	param_grid = {'learning_rate': [0.01, 0.05, 0.1, 0.5, 1], 'min_samples_split': [2, 5, 10, 20], 'max_depth': [2, 3, 5, 10]}
KNeighborsClassifier	param_grid = {'n_neighbors': np.arange(1, 25)}
SGDClassifier	param_grid = {"loss": ["hinge", "log", "squared_hinge", "modified_huber"], "alpha": [0.0001, 0.001, 0.01, 0.1], "penalty": ["l2", "l1", "none"]}
SupportVectorMachines	param_grid =,{'kernel': ['rbf'], 'gamma': [1e-2, 1e-3, 1e-4, 1e-5], 'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}, {'kernel': ['sigmoid'], 'gamma': [1e-2, 1e-3, 1e-4, 1e-5], 'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}, {'kernel': ['linear'], 'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}
DecisionTreeClassifier	param_grid = {'criterion':['gini','entropy'], 'max_depth':[4,5,6,7,8,9,10,11,12,15,20,30,40,50,70,90,120,150]}
RandomForestClassifier	param_grid = {,'n_estimators': [200, 500, 1000], 'max_features': ['auto', 'sqrt', 'log2'], 'max_depth': [4, 5, 6, 7, 8], 'criterion': ['gini', 'entropy']}
BernoulliNB	None
QuadraticDiscriminantAnalysis	None
LinearDiscriminantAnalysis	None
GaussianNB	None

Table 20.: Overview of the implemented hyperparameter optimization

Table 21.: Output of the RFECV by meta-dataset and attribute

Attribute Name	RFECV_ranking y_target1	RFECV_support y_target1	RFECV_ranking y_target2	RFECV_support y_target2
algorithm_name_BernoulliNB	1	True	1	True
algorithm_name_NuSVC	1	True	1	True
algorithm_name_ada_boost_classifier	1	True	1	True
algorithm_name_bagging_classifier	1	True	1	True
algorithm_name_decision_tree_classifier	1	True	1	True
algorithm_name_extra_tree_classifier	1	True	1	True
algorithm_name_gaussian_naive_bayes	1	True	1	True
algorithm_name_gradient_boosting_classifier	1	True	1	True
algorithm_name_k_neighbors_classifier	1	True	1	True
algorithm_name_linear_discriminant_analysis	1	True	1	True
algorithm_name_logistic_regression	1	True	1	True
algorithm_name_neural_network_mplClassifier	1	True	1	True
algorithm_name_quadratic_discriminant_analysis	1	True	1	True
algorithm_name_random_forest_classifier	1	True	1	True
algorithm_name_stochastic_gradient_descent	1	True	1	True
algorithm_name_support_vector_machines	1	True	1	True
algorithm_name_voting_classifier	1	True	1	True
best_node.sd	1	True	10	False
can_cor.mean	1	True	1	True
can_cor.sd	1	True	1	True
cat_to_num	1	True	1	True
cor.mean	1	True	23	False
cor.sd	1	True	22	False
elite_nn.sd	1	True	9	False
freq_class.sd	1	True	24	False
g_mean.mean	1	True	1	True
g_mean.sd	1	True	1	True
h_mean.mean	1	True	19	False
h_mean.sd	1	True	13	False
iq_range.sd	1	True	25	False
kurtosis.mean	1	True	1	True
leaves_branch.sd	1	True	8	False
leaves_corrob.sd	1	True	26	False
leaves_homo.sd	1	True	1	True
leaves_per_class.sd	1	True	1	True
linear_discr.sd	1	True	3	False
mad.mean	1	True	14	False
mad.sd	1	True	7	False
max.sd	1	True	6	False
min.mean	1	True	5	False
min.sd	1	True	11	False
naive_bayes.sd	1	True	4	False
nodes_per_level.sd	1	True	1	True
nodes_repeated.sd	1	True	1	True
nr_bin	1	True	1	True
nr_cat	1	True	1	True
nr_disc	1	True	12	False
nr_norm	1	True	1	True
nr_outliers	1	True	16	False
num_to_cat	1	True	1	True
one_nn.sd	1	True	15	False
range.sd	1	True	21	False
sd_ratio	1	True	1	True
skewness.mean	1	True	17	False
sparsity.sd	1	True	20	False
tree_imbalance.sd	1	True	1	True
tree_shape.sd	1	True	2	False
w_lambda	1	True	1	True
worst_node.sd	1	True	27	False
nr_cor_attr	2	False	18	False
median.sd	3	False	28	False
class_conc.sd	4	False	31	False
kurtosis.sd	5	False	30	False
t_mean.sd	6	False	34	False
mean.mean	7	False	36	False
mean.sd	8	False	35	False
var.sd	9	False	38	False
skewness.sd	10	False	32	False
mut_inf.sd	11	False	33	False
joint_ent.sd	12	False	29	False
sd.sd	13	False	37	False
attr_ent.sd	14	False	39	False

Continued on next page

Table 21 - continued from previous page

Attribute Name	RFECV_ranking y_target1	RFECV_support y_target1	RFECV_ranking y_target2	RFECV_support y_target2
class_conc.mean	15	False	42	False
sd.mean	16	False	65	False
ns_ratio	17	False	56	False
var_importance.sd	18	False	77	False
worst_node.mean	19	False	47	False
meta_dataset_computation_time	20	False	44	False
best_node.mean	21	False	48	False
sparsity.mean	22	False	80	False
leaves_branch.mean	23	False	87	False
inst_to_attr	24	False	70	False
max.mean	25	False	57	False
var_importance.mean	26	False	43	False
joint_ent.mean	27	False	76	False
class_ent	28	False	50	False
attr_to_inst	29	False	49	False
mut_inf.mean	30	False	64	False
iq_range.mean	31	False	72	False
linear_discr.mean	32	False	82	False
leaves	33	False	88	False
cov.mean	34	False	58	False
nodes_repeated.mean	35	False	86	False
cov.sd	36	False	41	False
nr_num	37	False	54	False
eigenvalues.mean	38	False	63	False
nr_attr	39	False	71	False
eigenvalues.sd	40	False	40	False
nr_inst	41	False	85	False
median.mean	42	False	53	False
elite_nn.mean	43	False	67	False
eq_num_attr	44	False	73	False
nr_class	45	False	84	False
freq_class.mean	46	False	74	False
random_node.mean	47	False	52	False
random_node.sd	48	False	60	False
leaves_corrob.mean	49	False	45	False
naive_bayes.mean	50	False	62	False
range.mean	51	False	46	False
tree_shape.mean	52	False	75	False
leaves_homo.mean	53	False	59	False
var.mean	54	False	61	False
t_mean.mean	55	False	66	False
tree_imbalance.mean	56	False	69	False
gravity	57	False	55	False
leaves_per_class.mean	58	False	78	False
one_nn.mean	59	False	68	False
nodes	60	False	79	False
tree_depth.mean	61	False	51	False
nodes_per_attr	62	False	83	False
tree_depth.sd	63	False	81	False
nodes_per_inst	64	False	89	False
nodes_per_level.mean	65	False	90	False
attr_ent.mean	66	False	91	False
attr_conc.sd	67	False	92	False
attr_conc.mean	68	False	93	False

C

ADDITIONAL RESULTS

Meta-dataset Transformation	Dataset name	Algorithm name	ROC-AUC score	Precision score	Recall score	F1 score
None	meta-dataset1	linear_discriminant_analysis	0,8400	0,9141	0,9184	0,9139
		bagging_classifier	0,8318	0,8936	0,8993	0,8953
		gradient_boosting_classifier	0,8226	0,8873	0,8924	0,8892
		BernoulliNB	0,8154	0,9075	0,9097	0,9085
		extra_tree_classifier	0,7925	0,9161	0,9201	0,9155
		k_neighbors_classifier	0,7766	0,9122	0,9167	0,9124
		neural_network_mplClassifier	0,7739	0,8441	0,8420	0,8430
		random_forest_classifier	0,7676	0,9181	0,9219	0,9171
		voting_classifier	0,7635	0,9181	0,9219	0,9171
		decision_tree_classifier	0,7410	0,8634	0,8646	0,8640
		ada_boost_classifier	0,7328	0,8548	0,8507	0,8526
		support_vector_machines	0,5509	0,7523	0,5990	0,6516
		NuSVC	0,5000	0,0250	0,1580	0,0431
		logistic_regression	0,5000	0,7090	0,8420	0,7698
		gaussian_naive_bayes	0,4782	0,8226	0,2153	0,1576
		quadratic_discriminant_analysis	0,4231	0,5396	0,1788	0,1368
		stochastic_gradient_descent	0,4080	0,7090	0,8420	0,7698
SMOTE	meta-dataset2	Bernoulli_NB	0,8858	0,9354	0,9364	0,9358
		bagging_classifier	0,8788	0,9147	0,9191	0,9159
		ada_boost_classifier	0,8582	0,9354	0,9364	0,9358
		linear_discriminant_analysis	0,8378	0,9334	0,9364	0,9332
		decision_tree_classifier	0,8354	0,9008	0,8671	0,8782
		gradient_boosting_classifier	0,8312	0,9165	0,9191	0,9176
		voting_classifier	0,8086	0,9334	0,9364	0,9332
		k_neighbors_classifier	0,8070	0,9334	0,9364	0,9332
		extra_tree_classifier	0,8055	0,9354	0,9364	0,9358
		random_forest_classifier	0,7878	0,9354	0,9364	0,9358
		neural_network_mplClassifier	0,6546	0,8518	0,6936	0,7398
		gaussian_naive_bayes	0,5252	0,8367	0,2370	0,2204
		support_vector_machines	0,5007	0,7618	0,8324	0,7909
		Nu_SVC	0,5000	0,0192	0,1387	0,0338
		logistic_regression	0,5000	0,7418	0,8613	0,7971
		stochastic_gradient_descent	0,5000	0,7418	0,8613	0,7971
		quadratic_discriminant_analysis	0,4891	0,7509	0,3006	0,3417
SMOTE AND RFECV	meta-dataset3	gradient_boosting_classifier	0,8858	0,9290	0,9306	0,9296
		linear_discriminant_analysis	0,8850	0,9409	0,9422	0,9393
		logistic_regression	0,8700	0,7025	0,8382	0,7644
		bagging_classifier	0,8681	0,9157	0,9191	0,9165
		Bernoulli_NB	0,8626	0,9290	0,9306	0,9296
		extra_tree_classifier	0,8433	0,9290	0,9306	0,9296
		ada_boost_classifier	0,8267	0,8927	0,8786	0,8839
		k_neighbors_classifier	0,8241	0,9290	0,9306	0,9296
		voting_classifier	0,8151	0,9290	0,9306	0,9296
		decision_tree_classifier	0,8151	0,9051	0,9075	0,9061
		random_forest_classifier	0,8091	0,9290	0,9306	0,9296
		neural_network_mplClassifier	0,7367	0,8584	0,7919	0,8127
		quadratic_discriminant_analysis	0,5590	0,7704	0,4277	0,4816
		Nu_SVC	0,5000	0,7025	0,8382	0,7644
		stochastic_gradient_descent	0,5000	0,7025	0,8382	0,7644
		gaussian_naive_bayes	0,4018	0,8656	0,2081	0,1346
		support_vector_machines	0,1483	0,6959	0,7919	0,7408
RFECV	meta-dataset4	bagging_classifier	0,8491	0,9029	0,9080	0,9040
		linear_discriminant_analysis	0,8373	0,9141	0,9184	0,9139
		BernoulliNB	0,8280	0,9075	0,9097	0,9085
		extra_tree_classifier	0,8222	0,9103	0,9149	0,9108
		gradient_boosting_classifier	0,8133	0,9029	0,9080	0,9040
		neural_network_mplClassifier	0,7973	0,8789	0,8819	0,8803
		random_forest_classifier	0,7927	0,9161	0,9201	0,9155
		k_neighbors_classifier	0,7811	0,8981	0,9045	0,8981
		decision_tree_classifier	0,7682	0,8684	0,8576	0,8622
		ada_boost_classifier	0,7603	0,8780	0,8819	0,8797
		voting_classifier	0,7585	0,9181	0,9219	0,9171
		quadratic_discriminant_analysis	0,5405	0,7596	0,5938	0,6479
		NuSVC	0,5000	0,0250	0,1580	0,0431
		logistic_regression	0,5000	0,7090	0,8420	0,7698
		gaussian_naive_bayes	0,4411	0,7309	0,7743	0,7505
		support_vector_machines	0,4089	0,7090	0,8420	0,7698
		stochastic_gradient_descent	0,3892	0,7090	0,8420	0,7698

Table 22.: Complete performance evaluation for the meta-dataset with binary target

Meta-dataset Transformation	Dataset name	Algorithm name	Precision score	Recall score	F1 score
None	meta-dataset5	random_forest_classifier	0,7395	0,8212	0,7729
		support_vector_machines	0,7395	0,8212	0,7729
		voting_classifier	0,7395	0,8212	0,7729
		extra_tree_classifier	0,7562	0,8056	0,7713
		bagging_classifier	0,7562	0,7882	0,7708
		k_neighbors_classifier	0,7331	0,8160	0,7682
		BernoulliNB	0,7252	0,7847	0,7531
		linear_discriminant_analysis	0,7516	0,7448	0,7458
		gradient_boosting_classifier	0,7276	0,7639	0,7439
		decision_tree_classifier	0,7250	0,7257	0,7250
		ada_boost_classifier	0,7087	0,7135	0,7108
		neural_network_mplClassifier	0,6485	0,6233	0,6352
		logistic_regression	0,5496	0,7413	0,6312
		NuSVC	0,5736	0,6858	0,6230
		stochastic_gradient_descent	0,5790	0,6111	0,5924
		gaussian_naive_bayes	0,0292	0,1319	0,0475
		quadratic_discriminant_analysis	0,3928	0,1024	0,0397
SMOTE	meta-dataset6	bagging_classifier	0,8344	0,8555	0,8309
		extra_tree_classifier	0,7894	0,8555	0,8207
		random_forest_classifier	0,7849	0,8555	0,8182
		voting_classifier	0,7849	0,8555	0,8182
		k_neighbors_classifier	0,7857	0,8555	0,8161
		gradient_boosting_classifier	0,7952	0,8208	0,8078
		decision_tree_classifier	0,7871	0,7572	0,7711
		linear_discriminant_analysis	0,7905	0,7514	0,7704
		ada_boost_classifier	0,7527	0,7283	0,7352
		Nu_SVC	0,6017	0,7399	0,6637
		quadratic_discriminant_analysis	0,6064	0,7052	0,6521
		Bernoulli_NB	0,8191	0,5607	0,6457
		neural_network_mplClassifier	0,7335	0,2775	0,3754
		gaussian_naive_bayes	0,7453	0,2081	0,1720
		support_vector_machines	0,6881	0,1676	0,0995
		logistic_regression	0,0192	0,1387	0,0338
		stochastic_gradient_descent	0,0015	0,0347	0,0029
SMOTE AND RFECV	meta-dataset7	bagging_classifier	0,7774	0,8208	0,7876
		random_forest_classifier	0,7304	0,8266	0,7740
		voting_classifier	0,7304	0,8266	0,7740
		extra_tree_classifier	0,7371	0,8150	0,7729
		ada_boost_classifier	0,7291	0,8208	0,7703
		gradient_boosting_classifier	0,7675	0,7688	0,7679
		decision_tree_classifier	0,7392	0,7630	0,7482
		k_neighbors_classifier	0,7172	0,7688	0,7421
		linear_discriminant_analysis	0,7764	0,7399	0,7351
		Bernoulli_NB	0,7541	0,7052	0,7266
		stochastic_gradient_descent	0,7349	0,7514	0,6876
		neural_network_mplClassifier	0,7371	0,5838	0,6496
		Nu_SVC	0,5221	0,7225	0,6062
		quadratic_discriminant_analysis	0,5758	0,2890	0,3644
		support_vector_machines	0,5580	0,0809	0,0817
		gaussian_naive_bayes	0,5442	0,0751	0,0695
		logistic_regression	0,0262	0,1618	0,0451
RFECV	meta-dataset8	random_forest_classifier	0,7243	0,8160	0,7626
		extra_tree_classifier	0,7375	0,8038	0,7622
		bagging_classifier	0,7347	0,7708	0,7515
		neural_network_mplClassifier	0,7269	0,7899	0,7494
		k_neighbors_classifier	0,7063	0,8038	0,7489
		linear_discriminant_analysis	0,7204	0,7743	0,7455
		gradient_boosting_classifier	0,7234	0,7691	0,7426
		BernoulliNB	0,6959	0,7917	0,7356
		decision_tree_classifier	0,7361	0,7344	0,7351
		ada_boost_classifier	0,7226	0,7153	0,7184
		support_vector_machines	0,5241	0,7240	0,6080
		NuSVC	0,5231	0,7188	0,6055
		voting_classifier	0,6278	0,5903	0,5860
		stochastic_gradient_descent	0,6179	0,5017	0,5483
		gaussian_naive_bayes	0,5419	0,5313	0,5350
		quadratic_discriminant_analysis	0,6276	0,2830	0,3528
		logistic_regression	0,0250	0,1580	0,0431

Table 23.: Complete performance evaluation for the meta-dataset with binary target

NB: place here information about funding, FCT project, etc in which the work is framed. Leave empty otherwise.