UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DATA PREPARATION PIPELINE RECOMMENDATION VIA META-LEARNING

FERNANDO REZENDE ZAGATTI

ORIENTADOR: PROF. DR. DIEGO FURTADO SILVA

São Carlos – SP

Maio/2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DATA PREPARATION PIPELINE RECOMMENDATION VIA META-LEARNING

FERNANDO REZENDE ZAGATTI

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial
Orientador: Prof. Dr. Diego Furtado Silva

São Carlos – SP

Maio/2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Fernando Rezende Zagatti, realizada em 26/05/2021.

Comissão Julgadora:

Prof. Dr. Diego Furtado Silva (UFSCar)

Profa. Dra. Marcela Xavier Ribeiro (UFSCar)

Prof. Dr. Ronaldo Cristiano Prati (UFABC)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Dedico este trabalho aos meus pais, irmãos e amigos, além dos meus orientadores do projeto, pessoas estas que não mediram esforços para que eu chegasse até esta etapa da minha vida.

AGRADECIMENTOS

Agradeço primeiramente à minha família, em que todos os momentos foram meu suporte em qualquer dificuldade encontrada durante meu percurso, me proporcionando abrigo, conselhos e sabedoria para que eu pudesse dar o meu melhor em qualquer atividade realizada. Agradeço também pela criação que me foi dada, a qual permitiu que eu chegasse a ser quem sou hoje.

Ao orientador deste presente projeto, Prof. Dr. Diego Furtado Silva, que através das suas vastas experiências e competências, me auxiliou em todas as dúvidas e questionamentos que apresentei. Sem sua assistência e dedicação a fim de transmitir seus conhecimentos para realização das pesquisas, não seria possível o correto desenvolvimento deste trabalho.

Aos membros da banca examinadora, Profª. Dra. Marcela Xavier Ribeiro e Prof. Dr. Ronaldo Cristiano Prati, e aos membros suplentes, Profª. Dra. Heloísa de Arruda Camargo e Prof. Dr. Ricardo Marcondes Marcacini, que tão gentilmente aceitaram participar e colaborar com este projeto de pesquisa. Suas contribuições serão utilizadas prontamente a fim de aprimorar as ideias e atingir os resultados esperados.

À empresa B2W Digital, parceira desta pesquisa, que me proporcionou a possibilidade de trabalhar dentro de um projeto real da empresa em paralelo com a realização do meu projeto de mestrado. Seu suporte que tornou possível a saída de minha cidade natal para realizar meus estudos e alcançar meus sonhos e objetivos.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pela bolsa de estudos e auxílio financeiro durante os primeiros meses do meu projeto de mestrado, o que me possibilitou a dedicação integral ao programa de pós-graduação e a operacionalização do estudo enquanto o estágio não estava efetivado pela B2W Digital.

Aos professores escolhidos para auxiliar no projeto Apache Marvin-AI em conjunto com meu orientador, Profª. Dra. Helena de Medeiros Caseli e Prof. Dr. Daniel Lucrédio, que além de prestar assistências e apoios durante o projeto da empresa, também realizaram diversas contribuições para o projeto de pesquisa do mestrado. Os três professores citados foram muito coerentes e não ajudaram apenas nas relações acadêmicas, mas também foram extremamente compreensíveis nas relações humanas. Os três são ótimas pessoas e quero levar para a vida toda como mestres e amigos.

Aos companheiros que entraram no mestrado e no projeto da B2W Digital na mesma época que eu, Bruno Silva Sette, Lucas Cardoso Silva e Lucas Nildaimon dos Santos Silva, que além de acrescentarem muito conhecimento durante nossas discussões e trocas de experiências, são amigos muito próximos que tive o prazer de conhecer. Sem eles, nada disso seria possível.

Ao corpo docente do Instituto de Computação da Universidade Federal de São Carlos pelo conteúdo que me apresentaram e da forma que me foi apresentado, tenho certeza que cada dia foi uma experiência de aprendizado única, tanto para mim quanto para os professores e demais alunos.

Deixo aqui um agradecimento especial a três professores que tive durante minha graduação, Prof. Dr. Silas Evandro Nachif Fernandes, Prof. Me. Patrick Pedreira Silva e Prof. Me. Renan Caldeira Menechelli, sem suas cartas de recomendação, nossas conversas e seus incentivos, tenho certeza que não seria possível chegar até aqui.

Por fim, a todos os amigos e amigas que conheci no Instituto de Computação, agradeço a todos pelo convívio, amizade e apoio. Também, a todos os profissionais e acadêmicos que contribuem compartilhando seus conhecimentos e experiências, direta ou indiretamente, permitindo a realização desta pesquisa, o meu sincero agradecimento. Tenho certeza que todas as novas ideias e descobertas serão levadas para toda a vida.

É muito melhor arriscar coisas grandiosas, alcançar triunfos e glórias, mesmo expondo-se a derrota, do que formar fila com os pobres de espírito que nem gozam muito nem sofrem muito, porque vivem nessa penumbra cinzenta que não conhece vitória nem derrota.

Theodore Roosevelt

RESUMO

A preparação de dados é uma etapa essencial no *pipeline* de aprendizado de máquina, com o objetivo de converter dados volumosos e inconstantes em dados refinados compatíveis com os algoritmos a serem aplicados. No entanto, a preparação de dados demanda muito tempo e requer conhecimento especializado. Cada conjunto de dados tem suas características particulares, que devem ser levadas em conta, e pode ser interpretado de maneiras diferentes. Nesse cenário, automatizar a preparação de dados e, por consequência, diminuir o esforço feito pelos cientistas de dados nesse estágio é um desafio científico de grande relevância prática. Apesar de sua relevância, as plataformas de automatização do aprendizado de máquina (AutoML) atuais desconsideram ou criam *pipelines* pré-definidos para a preparação de dados, que não se adaptam às características do conjunto de dados a ser tratado. Tentando preencher essa lacuna, apresentamos um sistema de recomendação baseado em meta-aprendizado para a preparação de dados. Nosso sistema recomenda cinco *pipelines*, classificados por relevância. Dessa maneira, é útil para usuários com níveis de experiência variados. Usando a principal recomendação para simular uma escolha totalmente automática, demonstramos que nossa proposta permite um melhor desempenho de um sistema AutoML, incapaz de encontrar um modelo de classificação devido aos dados ruidosos. Além disso, as taxas de precisão do nosso método são semelhantes às alcançadas por um algoritmo baseado no aprendizado por reforço com o mesmo objetivo, mas é até duas ordens de magnitude mais rápido. Além disso, demonstramos nosso método em uma aplicação do mundo real e avaliamos seus benefícios e limitações neste cenário.

Palavras-chave: Automatização, Preparação de dados, Meta-aprendizado, Pré-processamento, Aprendizado de máquina

ABSTRACT

Data preparation is a essential stage in the machine learning pipeline, aiming to convert noisy and disordered data into refined data compatible with the algorithms. However, data preparation is time-consuming and requires specialized knowledge. In this scenario, automating data preparation and decreasing the effort made by data scientists at this stage is a scientific challenge of great practical relevance. Each dataset has its particular characteristics and can be interpreted in different ways. Despite its relevance, current automated machine learning (AutoML) platforms disregard or make simple hardcoded pipelines for data preparation. Trying to fill this gap, we present a meta-learning-based recommendation system for data preparation. Our system recommends five pipelines, ranked by their relevance, so it is useful for users with varied experience levels. Using the top recommendation to simulate an entirely automatic choice of data preparation pipeline, we demonstrate that our proposal allows a better performance of an AutoML system, unable to find a classification model due to the noisy data. Besides, our method's accuracy rates are similar to those achieved by a reinforcement-learning-based algorithm with the same goal, but it is up to two orders of magnitude faster. Morevover, we demonstrate our method in a real-world application and evaluate its benefits and limitations in this scenario.

Keywords: Automated, Data preparation, Meta-learning, Preprocessing, Machine learning

List of Figures

List of Tables

Glossary

AD – *Approximate Duplicate*

AI – *Artificial Intelligence*

AutoML – *Automated Machine Learning*

CART – *Classification and Regression Trees*

CC – *Constraint discovery and checking*

DS – *Decimal Scale normalization*

ED – *Exact Duplicate*

EM – *Expectation-Maximization*

ETL – *Extraction, Transformation, Loading*

FN – *False Negative*

FP – *False Positive*

HCA – *Hierarchical Clustering*

ID – *Identifier*

IQR – *Interquartile Range*

LASSO – *Least Absolute Shrinkage and Selection Operator*

LC – *Removing Collinear features*

LDA – *Linear Discriminant Analysis*

LOF – *Local Outlier Factor*

MARS – *Multivariate Adaptive Regression Splines*

MF – *Most Frequent Value*

MICE – *Multiple Imputation by Chained Equations*

ML – *Machine Learning*

MM – *MinMax*

MR – *Missing Values Ratio*

MSE – *Mean Squared Error*

NB – *Naive Bayes*

OHE – *One-Hot-Encoding*

OLS – *Ordinary Least Squares Regression*

PC – *Pattern checking*

SMBO – *Sequential Model-Based Optimization*

SVM – *Support Vector Machine*

TB – *Tree-Based classifier for feature selection*

TN – *True Negative*

TP – *True Positive*

UFSCar – *Federal University of São Carlos*

VPN – *Virtual Private Network*

WR – *Wrapper subset evaluator*

ZSB – *Z-Score-Based method*

ZS – *Z-Score*

iForest – *Isolation Forest*

kNN – *k-Nearest Neighbors*

CONTENTS

Chapter 1

INTRODUCTION

1.1 Contextualization

Making a computer perform simple tasks that do not require the explicit definition of complex calculations is a significant challenge in Computer Science. One example of factors that make it difficult is common sense, knowledge acquired from experiences and observations of the world (ROSA, 2011). This idea is the basis of artificial intelligence (AI), which consists of computational mechanisms that aim to make the computer perceive, reason, and act, establishing how machines can perform different activities (NORVIG; RUSSELL, 2014). AI can be exemplified by the detection of Facebook faces[1] and the development of autonomous cars[2].

From another perspective, machine learning (ML) is a subarea of AI that seeks to develop or apply algorithms that can learn to solve particular problems by identifying patterns and extracting knowledge automatically, without the need of programming and defining specific routines (WITTEN et al., 2016). For example, search engines like Google, Yahoo, or Bing use machine learning to offer more accurate searches and ensure that no unwanted results appear.

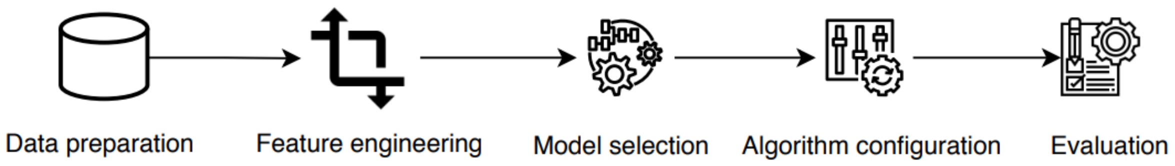
ML models require good input data to perform their functions, making acquiring and storing large datasets a common activity in many companies. Due to the enormous capacity to generalize problems, several companies apply ML to aid analysis and decision-making based on data. However, there are several steps during the development process of the ML pipeline[3], as illustrated in Figure 1.1.

[1]https://www.facebook.com/help/122175507864081

[2]https://www.nytimes.com/2020/10/26/technology/driverless-cars.html

[3]A pipeline is the segmentation of the orientations carried out by algorithms so that the processor searches for its instructions and places them in a queue in memory (DRORI et al., 2018).

Figure 1.1: Prototype of an ML pipeline



Source: Elaborated by the author

Each of these steps plays a fundamental role in the correct development of intelligent models:

- ***Data preparation:*** Performs different preprocessing steps in the raw data, such as cleaning missing values and normalizing continuous values.
- ***Feature engineering:*** Seeks to aggregate or separate attributes through extensive analysis to obtain better generalization for learning.
- ***Model selection:*** Selection of one or more ML algorithms to execute the learning.
- ***Algorithm configuration:*** Configuration of the hyperparameter[4] values of the selected algorithm.
- ***Evaluation:*** Analysis of the induced models and choice of the best case.

Currently, within the steps listed above, data preparation (also called data preprocessing) is the one that requires the longest time, in practice, due to all the possibilities of techniques and which can be used separately or together for each data source. Thus, this step performs a series of operations to make the data cleaner and ready for the attribute engineering process. Among its main methods, we can mention (GARCÍA; LUENGO; HERRERA, 2015):

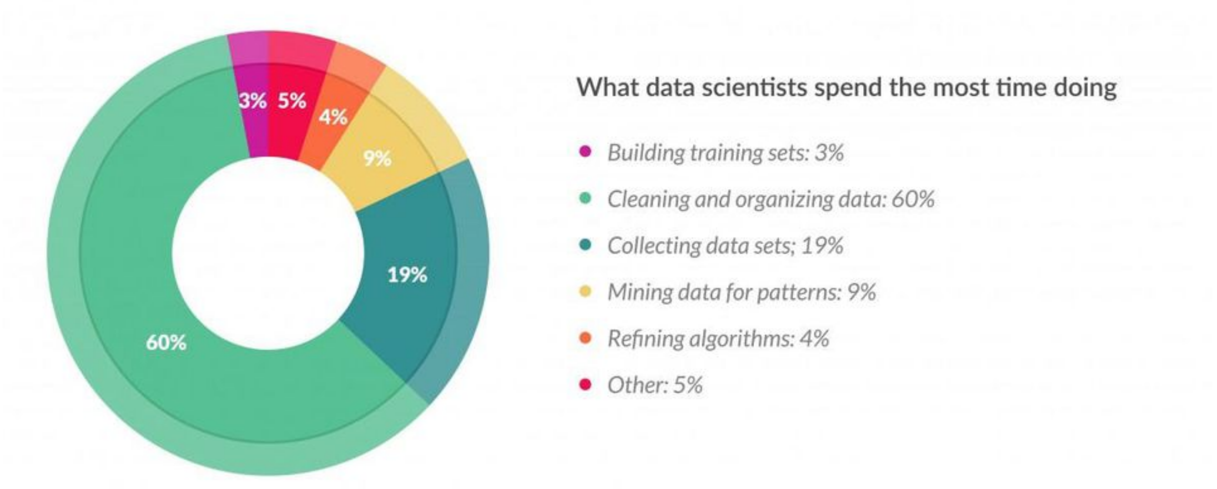
- ***Data integration:*** The integration between data from different sources is necessary to perform a more in-depth analysis of the data, managing to observe redundancies, dependencies between information, and values that conflict.
- ***Data cleaning:*** Seeks to treat inconsistencies and structural problems, such as missing values and class imbalance.
- ***Data transformation:*** Data transformations convert data to new types and allow normalization and standardization for the same numerical system, giving the ML model a greater capacity for generalization.

[4]Hyperparameters are parameters whose values are sets before starting the ML processes.

Problems associated with data preparation are common and significant in the ML context since unprepared data can directly interfere with how the learning algorithms will generalize knowledge about the studied phenomenon. Furthermore, according to Pyle (1999), due to the complex nature of the data received by ML algorithms, many developers spend much time preparing the information so that their concepts are understood by the machine and, thus, achieve satisfactory results in training.

As noted in Figure 1.2, an article in Forbes magazine[5] showed that about 80% of the time spent on data science projects is related to data preparation, in which 19% of the time regards collecting the data and 60% concerns cleaning and organizing them. If the data preparation steps could be made simpler, the long time consumed by them could be used in the other steps of the ML pipeline, such as selecting and configuring the algorithms.

Figure 1.2: Time consumed by each step in the process of creating ML models



Source: Forbes

The data quality is a significant concern for ML since poor-quality data can lead to unreliable knowledge. Therefore, good ML models depend not only on algorithms but also on adequate datasets. Besides, as previously noted, data preparation is the most time-consuming stage of the whole pipeline taking up to 80% of the total development time (CHU et al., 2016; ZHANG; ZHANG; YANG, 2003). Consequently, developing new approaches for automating data cleaning and preparation has been of increasing interest to the industry and academia.

ML models need to clean input data to perform their role and conduct predictive model training. Within the context of ML, automated machine learning (AutoML) has risen to automatically define the techniques that will be used and the hyperparameters of each one, opti-

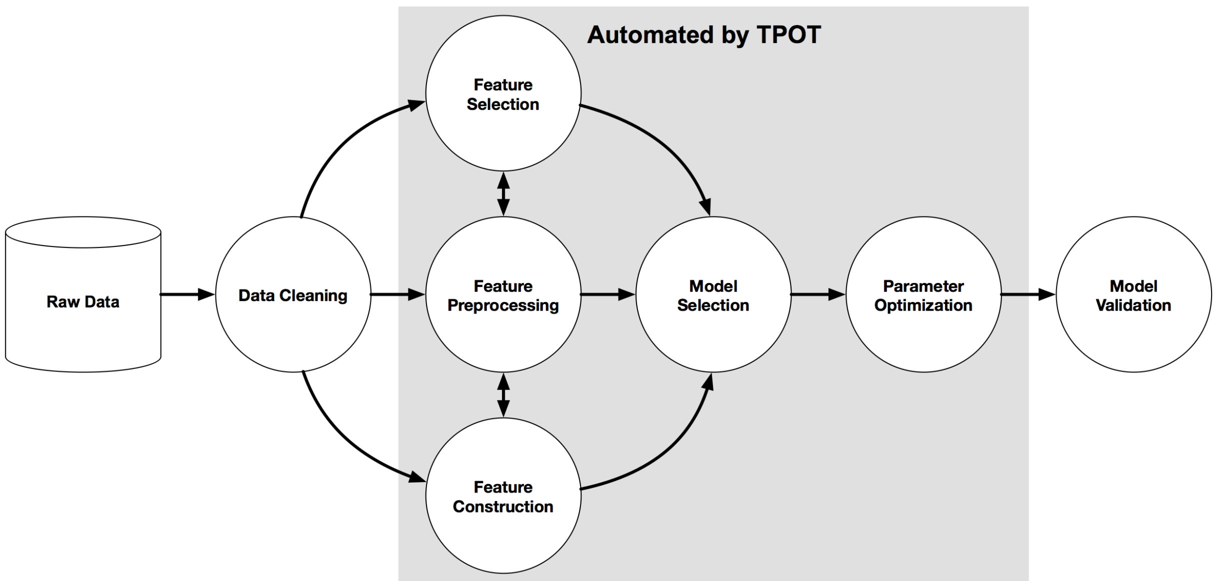
[5]https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/#2ea5ca046f63

mizing the performance of the generated models. Several tools were created with this purpose, including Auto-sklearn[6], TPOT[7], H2O.ai[8]. By automating the ML pipeline and reducing the human effort required to apply it, AutoML makes ML possible even for non-expert users. In data preparation, AutoML consists of taking raw data and producing a clean dataset for ML algorithms.

However, AutoML still faces some obstacles, mainly because there is still no platform able to automate the complete ML pipeline efficiently. Current AutoML platforms perform little or no data preparation, acting only on the feature engineering and further steps, but do not deal with data cleaning such as missing values imputation and normalization (LE; FU; MOORE, 2020).

An example of such lack of automated data preparation techniques is the TPOT's pipeline, illustrated by Figure 1.3 (OLSON et al., 2016a). It assumes that the input data is ready for application, covering only the steps of feature engineering and algorithm configuration and optimization. There are no considerable analyzes regarding the data structure and how to deal with the needed cleaning (OLSON et al., 2016b, 2016a). Other AutoML tools follow the same tendency.

Figure 1.3: AutoML pipeline by TPOT



Source: Olson et al. (2016a)

Another AutoML platform, Auto-sklearn was developed using Bayesian optimization and applying meta-learning to achieve a faster result in its search space. Auto-sklearn also requires the input data ready for application. However, unlike TPOT, it performs a default preprocessing

[6]https://automl.github.io/auto-sklearn/master/
[7]https://epistasislab.github.io/tpot/
[8]http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html

by performing imputation, one-hot-encoding, and standardization. However, this single pipeline may not be the most suitable for all cases (FEURER et al., 2015).

In this scenario, knowledge about past experiences seems to be a good starting point for the process. By analyzing the characteristics of a dataset, it may be possible to reuse procedures that have already been successful in other similar datasets. For this purpose, meta-learning emerged as an approach for learning based on the output of past ML algorithms (BRAZDIL et al., 2008). In other words, meta-learning looks at the situations of models that have been previously trained and learns how to make predictions for the new scenario it is analyzing (FACELI et al., 2011).

In general, meta-learning explores the accumulation of performance experiences from multiple applications of ML algorithms, making it flexible according to the problem studied (MAIMON; ROKACH, 2005). Due to the approach uses the characteristics of the data itself or the models implemented, there are no expensive computational costs in processing (GUERRA; PRUDÊNCIO; LUDERMIR, 2007).

1.2 Motivation

Several preprocessing techniques are necessary during the development of an ML model, given that real-world data is, in essence, noisy and heterogeneous. In other words, real-world data have several inconsistencies that make it difficult to be applied without any preprocessing to learn the models due to missing data, discrepancies, and improper information (POLYZOTIS et al., 2017).

In this scenario, some tools aim to perform data cleaning automatically, such as Learn2Clean, which sought to automate data preparation via reinforcement learning (BERTI-EQUILLE, 2019). However, none of these tools handle categorical data and, because they are iterative processes, they end up being time-consuming and computationally expensive.

As previously noticed, it is necessary to perform data compatibility with the ML algorithm, given that some models cannot handle certain different types of data. For instance, if the algorithm does not accept categorical attributes as input, it is necessary to transform them into numeric variables through one-hot-encoding[9] (OHE). Thus, executing the recommendation and due transformations based on the characteristics of the data is an interesting and little explored area of research.

[9]A technique that consists of transforming each categorical value into a column with binary values, with the value 1 as present and 0 as absent.

Due to its importance, several ML tools implement varied preprocessing operations, such as label encoding, standardization, class balancing, and data imputation. Therefore, correctly preprocessing the data is mostly a matter of choice of which operation to use in each case.

The choice for the data preparation process is currently based on the entire manual procedure for its correct construction. In the same way that AutoML seeks to facilitate the development of ML models for data scientists, this research aimed to automatize the step with the most cost of time within the ML pipeline: the data preparation.

1.3 Objectives

The primary objective of this work was to develop and evaluate a method to automatically perform the data preparation to serve as input to ML algorithms. For this, the use of meta-learning was explored, a method capable of making recommendations quickly through past experiments. In addition, the work attempted to make the system flexible, assisting scientists with little experience and allowing experts to modify the recommendations as needed. The specific objectives of this work were:

- Check the impact of preprocessing on ML algorithms;
- Evaluate and adjust the proposed approach to be successful in several domains, including real-world data.
- Optimize the proposed method to have low computational cost, to make it viable for practical use.

1.4 Dissertation organization

This chapter presents a brief contextualization, motivation, and the general and specific objectives for the conduct of this study. The next chapters are organized as follows:

- Chapter 2 presents the theoretical foundation concerning AutoML, addressing an overview of the main concepts related to automation. Initially, basic AutoML concepts are presented, followed by a more detailed explanation of each step of the pipeline (data preparation, feature engineering, model selection and configuration, and evaluation). Finally, the chapter presents some final considerations about AutoML.

- Chapter 3 presents the theoretical foundation about meta-learning, addressing the extraction of characteristics (meta-attributes) and the three forms of the meta-learning present in the literature. Finally, there is an explanation regarding the application of meta-learning in recommendation systems.
- Chapter 4 presents the bibliographic review, showing the related works to this dissertation.
- Chapter 5 details the elaboration and composition of our pipeline recommendation system named MetaPrep. There is the methodology adopted for this research, external supports, machine configurations used, and an explanation of the proposal, illustrating how the system works and how it performs the prediction and cleaning for new data.
- Chapter 6 presents and discusses the results obtained from the experimental study and compares the MetaPrep, our new approach for data preparation, with other tools in the literature. In addition, there is a case study using real data from B2W Digital, pointing out possible improvements for the proposed project.
- Chapter 7 concludes the dissertation, presenting the limitations, the main contributions, and possible future work.

Chapter 2

Automated Machine Learning

2.1 Initial considerations

Conventionally, constructing ML pipelines demand a significant human intervention in all stages, from data preparation until final evaluation (QUANMING et al., 2018). AutoML, in turn, intends to break this paradigm proposing that the whole process could be done by the computer without human interference (FEURER et al., 2015). AutoML has the potential to enable data analysts and scientists to create ML models with great efficiency and productivity, aiming to guarantee good results without human assistance (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

In this scenario, human labor is fundamental for good ML models since it defines and configures all the techniques and hyperparameters in all stages. In contrast, AutoML came with the supposition that the entire process can be executed automatically. Table 2.1, adapted from Quanming et al. (2018), illustrates the difference between the need for human interventions in conventional ML and AutoML.

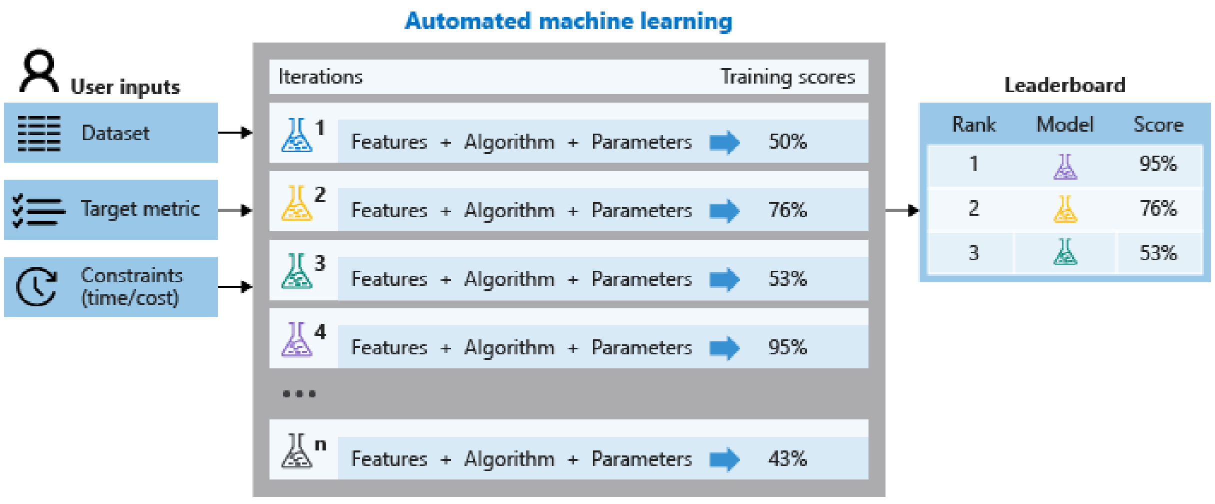
Table 2.1: Comparison between conventional ML and AutoML

	Conventional ML	**AutoML**
Data preparation	The data and the types of required cleaning are analyzed.	Automated by the computer
Feature engineering	Specialists analyze the data and perform the necessary transformations in the attributes	
Model selection	The best ML techniques based on domain knowledge are defined	
Algorithm configuration	The hyperparameters of the ML tools are adjusted to achieve better performance	
Evaluation	Evaluation of the results obtained for the proposed problems	

AutoML makes ML techniques more accessible to non-specialists interested in ML models but do not have enough time or resources to learn how to work in detail behind the methods employed. In addition, AutoML allows expert scientists or practitioners to start development from a good point. Focusing on this process of automating the entire ML pipeline, AutoML has the potential to enable analysts and data scientists to create models with efficiency and productivity, aiming to guarantee good results without human assistance (HUTTER; KOTTHOFF; VANSCHOREN, 2019).

As reported by the TPOT pipeline (c.f. Figure 1.3), other AutoML tools do not perform data preparation. Figure 2.1 demonstrates the fundamental operations that AutoML usually proposes to perform (feature engineering, model selection, and hyperparameterization). The limited focus on data preparation is noticeable in the current AutoML platforms, considering that they assume the dataset is already clean. The data preparation stage demands high data interpretability skills. Each dataset has characteristics that may impact the whole ML pipeline, which requires a correct choice for the cleaning algorithms to avoid errors. The number of techniques related to the preprocessing of the data is due to their high variability. Data can be of the most diverse types (e.g., numerical, categorical, date-time), in different scales, and even with low quality due to a high quantity of noise and missing values (GARCÍA; LUENGO; HERRERA, 2015). In this scenario, human knowledge of the various techniques is usually necessary to choose which ones to apply to the data.

Figure 2.1: Process diagram performed by AutoML


Automated machine learning
User inputs
Dataset
Target metric
Constraints (time/cost)
Iterations
Training scores
1 Features + Algorithm + Parameters 50%
2 Features + Algorithm + Parameters 76%
3 Features + Algorithm + Parameters 53%
4 Features + Algorithm + Parameters 95%
n Features + Algorithm + Parameters 43%
Leaderboard
Rank Model Score
1 95%
2 76%
3 53%

Source: Microsoft Azure documentation[1]

[1]https://docs.microsoft.com/en-us/azure/machine-learning/concept-automated-ml

2.2 Data preparation

Automating the data preparation process intends to allow developers and data scientists to reduce the amount of time consumed by this step. As a side effect, they may spend this effort in other stages, such as configuring the ML algorithm or applying the obtained knowledge.

For this automatization process to occur, it is necessary that the algorithms correctly identify the characteristics of the input data to be able to deal with it properly. However, because it is a heavy procedure due to a large amount of data and its respective variations, it is essential to identify the points that need directional attention (JOHNSON; ANDERSON; SPROULE, 2007; GARCÍA; LUENGO; HERRERA, 2015).

Guyon et al. (2015) identify some problems when performing data preparation, mentioning difficulties regarding data distribution (class balancing), the size of the datasets, identification of missing values, identification of data types (categorical or numeric), among others. Therefore, it requires an extensive and deep analysis before directing data to ML models.

The data preparation step is mandatory for any ML model. It converts disordered and dirty data into a new set with refined data compatible with the algorithms to be used (GARCÍA; LUENGO; HERRERA, 2015). Training models with unprepared data can result in outputs that offer little knowledge for the proposed problem. In the worst case, the algorithm will present errors during the execution process, and, consequently, it will not complete all the stages of the pipeline. A clear example of it is the Support Vector Machine (SVM) algorithm, which does not support categorical data in its training procedure.

The data preparation techniques used in the literature are commonly separated into three groups, namely: data integration, cleaning, and transformation (BATISTA, 2003; BERTI-EQUILLE, 2019). The following sections will provide more details about each of these groups.

2.2.1 Data integration

When data comes from several sources for creating a single dataset, they can generate many redundancies and inconsistencies in the merging process. The inconsistent information can cause processing delay and produces ML models with low generalization for the problem (GARCÍA; LUENGO; HERRERA, 2015).

Redundancies lead to a potentially significant increase in the size of the dataset. It reflects in the total runtime of the ML pipeline and makes it prone to overfit[2]. Commonly, to avoid these problems, some operations are used to integrate data from several sources, such as the unification of the sets, analysis regarding the correlation of attributes, detection of equal values, and detection of conflicts between data from different sources (GARCÍA; LUENGO; HERRERA, 2015; HAWKINS, 2004).

Due to errors during the integration process, disturbances in some attribute values (e.g., the only difference between two examples being the identifier) can produce identical repeated examples considered distinct during the integration. Instances may also have inconsistencies if the values of the attributes are outside the range established for an assessed column. However, these cases become relatively easy to verify because they are information that is usually in the metadata of the dataset (GARCÍA; LUENGO; HERRERA, 2015).

Tomlin and Welch (1986) claim that there are two main reasons for detecting duplicate examples: (1) it can be explored to reduce the size of the dataset and the cost of the algorithm processing; (2) it can be checked to avoid models considered unfeasible due to a large amount of irrelevant information. Besides, numerous duplicate instances are a sign of inefficiency in the formulation of the dataset. So, removing them can lead the user to an improved formulation if possible.

2.2.2 Data cleaning

Data cleaning aims to correct the original dataset, reduce unnecessary details, and fix some data characteristics. For example, apply imputation techniques to ensure are no missing values in the dataset allows training ML models that are sensitive or unable to execute with this issue. Among the primary operations of the data cleaning are the treatment of missing data, the detection of outliers, and class rebalancing (GARCÍA; LUENGO; HERRERA, 2015).

Missing data is a common issue in datasets used in real environments. It happens when some instances do not have any given value for one or more attributes. It is a problem for statistical analysis and one of the main difficulties for training ML models. Banks et al. (2011) confirm that less than 1% of missing data are trivial and 1 to 5% are manageable. However, 5 to 15% of missing data require more attention about the used techniques. Finally, below 15% can seriously hinder the interpretation of the data set.

[2]Overfitting occurs when the ML model fits too much the training dataset, which leads to high accuracy for the training set but does not generalize the learning to novel instances.

Table 2.2 shows a hypothetical data set with five attributes, indicating your sex, age, salary, bank balance, and whether there are frequent late payments, respectively. The missing values are represented by the character "?." This dataset will be used as an example to demonstrate the data cleaning in the following topics.

Table 2.2: Hypothetical dataset

index	sex	age	salary	balance	delays
0	Male	32	1400	1450	Y
1	Male	38	0	250	Y
2	Female	24	?	4500	N
3	Male	?	2500	2700	?
4	Female	55	5000	32000	N
5	Female	?	3200	24000	N
6	?	59	2400	8300	Y
7	Female	38	1400	800	Y

Among the most used techniques for dealing with missing data are:

- ***Case deletion:*** This technique excludes the instances containing missing values. The case deletion needs to be carefully applied, as this technique may cause a significant loss of information. This technique is better applied when missing values happen at random and small-scale (LITTLE; RUBIN, 2019). Table 2.3 shows how the case deletion would be applied to the hypothetical dataset, which excluded instances 2, 3, 5, and 6.

Table 2.3: Hypothetical dataset with case deletion

index	sex	age	salary	balance	delays
0	Male	32	1400	1450	Y
1	Male	38	0	250	Y
4	Female	55	5000	32000	N
7	Female	38	1400	800	Y

- ***Mean imputation:*** This technique consists of filling missing values of a particular attribute (characterized by each column of the dataset) with the mean of the known values of this same attribute (BANKS et al., 2011). In addition to only dealing with numerical data, this imputation technique can disturb the variation of the data, artificially increasing the significance of any statistical test. Table 2.4 presents this technique in practice, adding the mean age and salary to the dataset.

- ***Median imputation:*** This technique works similarly to the previous one, filling missing values of a particular attribute with the general median or the median of all known values.

Table 2.4: Hypothetical dataset with mean imputation

index	sex	age	salary	balance	delays
0	Male	32	1400	1450	Y
1	Male	38	0	250	Y
2	Female	24	**2271,42**	4500	N
3	Male	**41**	2500	2700	**?**
4	Female	55	5000	32000	N
5	Female	**41**	3200	24000	N
6	**?**	59	2400	8300	Y
7	Female	38	1400	800	Y

Although it also performs the imputation only on numerical data, this technique is proper when the distribution of a distinguished attribute is skewed, in other words, when the numerical distributions are tending to a specific side (BANKS et al., 2011). Table 2.5 shows the application of median imputation.

Table 2.5: Hypothetical dataset with median imputation

index	sex	age	salary	balance	delays
0	Male	32	1400	1450	Y
1	Male	38	0	250	Y
2	Female	24	**2400**	4500	N
3	Male	**38**	2500	2700	**?**
4	Female	55	5000	32000	N
5	Female	**38**	3200	24000	N
6	**?**	59	2400	8300	Y
7	Female	38	1400	800	Y

- ***Most frequent imputation:*** This technique will replace all missing values with the most frequent value in the column. The main advantage of this imputation is the possibility of being used in numerical and categorical data (BISONG, 2019). As reported in Table 2.6, the most frequent imputation performs the cleaning of the “age” and “delays” columns.

Table 2.6: Hypothetical dataset with most frequent imputation

index	sex	age	salary	balance	delays
0	Male	32	1400	1450	Y
1	Male	38	0	250	Y
2	Female	24	**1400**	4500	N
3	Male	**38**	2500	2700	**Y**
4	Female	55	5000	32000	N
5	Female	**38**	3200	24000	N
6	**Female**	59	2400	8300	Y
7	Female	38	1400	800	Y

On the data cleaning, it is also possible to deal with outliers, which are abnormal values present in the original dataset. In other words, outliers are values that significantly vary from the rest of the set. Outliers can affect the generalization of the algorithm and disturb the statistical analysis (CUNHA; CARVAJAL, 2009).

Data used in real-world applications are often voluminous and inconstant; that is, the data have numerous attributes, are collinear, redundant, and have outliers (FRANÇOIS; WERTZ; VERLEYSEN, 2011). To deal with these issues, there are techniques for detecting outliers, such as interquartile range (IQR) (VINUTHA; POORNIMA; SAGAR, 2018), isolation forest (iForest) (LIU; TING; ZHOU, 2008), and local outlier factor (LOF) (BREUNIG et al., 2000).

Furthermore, many datasets suffer from imbalanced classes, when there is a disproportionate number of instances in each class (LAURIKKALA, 2001). Some techniques have been developed in the literature to solve this problem. The two simplest and most applied are:

- ***Random oversampling:*** This technique tries to repair the distribution of classes by randomly replicating samples in minority classes, thus being a non-heuristic technique. The problem with this method is precisely the replication, which can cause overfitting by generating several duplicated data (BATISTA; PRATI; MONARD, 2004).
- ***Random undersampling:*** It is also a non-heuristic technique, but unlike the previous one, this tries to balance classes by randomly excluding samples from the majority classes (BATISTA; PRATI; MONARD, 2004).

2.2.3 Data transformation

In general, since many ML algorithms can deal with only certain types of data (for example, numeric or symbolic values), some datasets need to be transformed to have a compatible format to be used as input to ML algorithms (FACELI et al., 2011). In this context, the original attributes may have some significance in the domain in which they were acquired. At the same time, they may not generate accurate predictive models when applied in ML without some transformations (GARCÍA; LUENGO; HERRERA, 2015).

ML models usually depend on a good combination of ML algorithms and data cleaning methods, making it necessary to consider pre-processing techniques for transformation, ensuring results with more assertiveness (SCHOENFELD et al., 2018).

The transformations applied to the data will not remove or generate new attributes from the original dataset. Instead, they will transform the distribution of the original values into a new set of values with the desired properties. The hypothetical dataset in Table 2.6, after the most frequent imputation, will demonstrate the data cleaning in the following topics. Thus, for structured data, the most common transformations can be grouped into three categories (FACELI et al., 2011):

- ***Standardization:*** This technique aims to rescale the attribute values so that the average is equal to 0 and the standard deviation is equal to 1. Another standardization alternative is to rescale the values between a minimum and a maximum value (0 and 1, for example) (BISONG, 2019). These techniques are referred to as standard scaler and minmax, respectively. Table 2.7 shows the standardization using minmax.

Table 2.7: Hypothetical dataset with minmax

index	sex	age	salary	balance	delays
0	Male	**0,22**	**0,28**	**0,03**	Y
1	Male	**0,40**	**0**	**0**	Y
2	Female	**0**	**0,28**	**0,13**	N
3	Male	**0,40**	**0,50**	**0,07**	Y
4	Female	**0,88**	**1**	**1**	N
5	Female	**0,40**	**0,64**	**0,74**	N
6	Female	**1**	**0,48**	**0,25**	Y
7	Female	**0,40**	**0,28**	**0,01**	Y

- ***Normalization:*** This technique consists of rescaling each of the samples in a unitary norm, considering that the techniques present in normalization commonly use the mathematical l1, l2, or maximum norms (BISONG, 2019). Table 2.8 shows the normalization using the normalizer technique with the default parameters (l2 norm).

Table 2.8: Hypothetical dataset with normalizer (l2 norm)

index	sex	age	salary	balance	delays
0	Male	**0,01**	**0,69**	**0,71**	Y
1	Male	**0,15**	**0**	**0,98**	Y
2	Female	**0**	**0,29**	**0,95**	N
3	Male	**0,01**	**0,67**	**0,73**	Y
4	Female	**0**	**0,15**	**0,98**	N
5	Female	**0**	**0,13**	**0,99**	N
6	Female	**0**	**0,27**	**0,96**	Y
7	Female	**0,02**	**0,86**	**0,49**	Y

- ***Symbolic-numeric conversion:*** Some ML algorithms cannot handle non-numeric data, such as neural networks and SVM. So, algorithms to convert categorical data to numerical values are necessary. Two of the most famous techniques to convert symbolic into numeric data are OHE, which transforms each value of a categorical attribute into a new column with 1 or 0 according to its pertinence, and label encoding, which transforms each categorical value into a specific number (FACELI et al., 2011). Symbolic-numeric conversion techniques like OHE and label encoding also have the inverse transform, which converts the data back to the original representation. Table 2.9 shows the symbolic-numeric conversion using the label encoding technique, transforming each categorical data into a respective numeric value.

Table 2.9: Hypothetical dataset with label encoding

index	sex	age	salary	balance	delays
0	**1**	32	1400	1450	**1**
1	**1**	38	0	250	**1**
2	**2**	24	1400	4500	**2**
3	**1**	38	2500	2700	**1**
4	**2**	55	5000	32000	**2**
5	**2**	38	3200	24000	**2**
6	**2**	59	2400	8300	**1**
7	**2**	38	1400	800	**1**

2.3 Feature engineering

Feature engineering is the act of developing, selecting, and aggregating characteristics for the dataset, that is, transforming the attribute space to improve the information contained in the raw data. Such as the data preparation does, the feature engineering process refines the input data and ensures a better representation for the algorithm (ZÖLLER; HUBER, 2019). This stage reduces the redundancy of similar data and selects the most relevant information for the ML, improving the use of algorithms and models through more polished data (HE; ZHAO; CHU, 2019).

This stage of the ML pipeline focuses on attribute analysis and dimensionality of the data. Likewise, as explained by Ge et al. (2017), data dimensionality is a fundamental topic of ML. Datasets with many attributes can cause problems to the classifier due to similar, redundant, or useless data. Therefore, dimensionality reduction through the analysis of the information of the attributes is necessary to extract knowledge in the process.

The above problem is referred to in the literature as the curse of dimensionality, an expression inserted by Richard Bellman to represent adding extra dimensions to Euclidean space, which causes difficulty to ML algorithms due to its large amount of information (KEOGH; MUEEN, 2017). According to Witten et al. (2016), the dimensionality reduction produces a compact and easily interpretable representation of the dataset. It makes the data more informative and leads to more assertive models.

As explained, the initial dataset may not accurately represent the problem due to the lack of representativeness of the characteristics, besides allowing it to contain a high number of attributes, leading to the curse of dimensionality. In this way, methods of processing these attributes are used in feature engineering. Among these methods, we can find:

- ***Feature construction:*** Process for generating new attributes based on the characteristics already existing in the dataset (RAWAT; KHEMCHANDANI, 2017). For instance, it is possible to infuse domain knowledge to create new features and combine sparse classes to generate more assertive models.
- ***Feature selection:*** This process selects the most relevant attributes in the datasets, allowing better use of ML models. This method can diminish redundancies by removing low-information features such as identifiers (IDs) or other text descriptions (HE; ZHAO; CHU, 2019).
- ***Feature transformation:*** This process aims to build new sets of attributes by manipulating the original features. The generation using mathematical and statistical operations can contribute to a better representation of the context of the problem (KATZ; SHIN; SONG, 2016; TRAN; XUE; ZHANG, 2016). This method includes dimensionality reduction, performing the transformation of information through analysis, and generating statistical attributes (GE et al., 2017).

2.4 Algorithm selection and configuration

There is a diversity of ML algorithms in the literature, which may have different or similar characteristics. According to the “no free lunch” theorem[3] (WOLPERT; MACREADY, 1997), these algorithms tend to be accurate in specific scenarios while they are insufficient in others.

Choosing the learning algorithm and adjusting hyperparameter values is a routine activity of ML researchers and practitioners, seeking to guarantee that the performance of the generated models can generalize the data is also part of this task. In addition, each algorithm has its own set of hyperparameters, which may or may not be similar, for which different values generally

[3]”There is no free lunch” represents the idea that it is always necessary to give something to get something in return.

impact the final results (HOOS, 2011). Traditionally, a specialist is responsible for choosing between the available algorithms and their respective hyperparameters. However, some techniques were proposed to aid the specialist to set the hyperparameters, such as:

- ***Grid Search***: According to Zöller and Huber (2019), Grid Search is one of the most basic techniques used in the literature, which defines a finite search scope for hyperparameters in the Euclidean space, resulting in a simple sequential search. However, because it is a sequential method, the scalability is highly influenced by the dimension of the search space, as the number of combinations grows exponentially with the number of hyperparameters and the scope size defined by the user.
- ***Random Search***: This technique performs the configuration search randomly. In other words, the algorithm chooses hyperparameters randomly until a stop criterion, like maximum runtime or convergence of the results, is reached. The convergence speed is relatively faster than Grid Search and can avoid local minimums more easily (BERGSTRA; BENGIO, 2012; LI; TALWALKAR, 2020)
- ***Sequential Model-Based Optimization (SMBO)***: This technique iterates through the adjusted models and handles them to make choices about what configurations to investigate, observing the perspective of interpolating the performance between the observed parameter configurations and extrapolating to regions never seen in parameter space (HUTTER; HOOS; LEYTON-BROWN, 2011). As reported by Brochu, Cora and Freitas (2010), a commonly used procedure to apply SMBO is using Bayesian optimization.

2.5 Evaluation

After cleaning and transforming the data, choosing the learning algorithm and its configuration, the model is trained and needs to be evaluated. The evaluation is the final step of the ML pipeline. Different evaluation measures (e.g., accuracy, precision, recall) are used to obtain the reliability that the trained algorithms can generalize the studied problem. The evaluation can also be carried out in conjunction with the experts on the data and the application domain, thus achieving a more in-depth analysis of the results (CAMILO; SILVA, 2009).

According to Zhu et al. (2010), there are four possible categories that instances of classification algorithms can fall into, namely: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Using these classifications allows checking the quality of the

ML models through tests of accuracy, precision, recall, F1-Score, among other techniques. The following topics describe each category, while Table 2.10 demonstrates how to analyze in sets.

- ***True positive***: The instance belongs to a determined class and was correctly classified by the model.
- ***True negative***: The instance does not belong to a determined class and was correctly classified by the model.
- ***False positive***: The instance does not belong to a determined class and was erroneously classified by the model as belonging to it.
- ***False negative***: The instance belongs to a determined class and was erroneously classified by the model as not belonging to it

Table 2.10: Possible categories in ML classifications

Categories	Positive (Real)	Negative (Real)	Total
Positive (Predicted)	TP	FP	TP+FP (Total number of instances considered positive)
Negative (Predicted)	FN	TN	FN+TN (Total number of instances considered negative)
Total	TP+FN (Total number of instances that are positive)	FP+TN (Total number of instances that are negative)	TP+TN+FP+FN (Total dataset instances)

Accuracy is one of the most used techniques to test the efficiency of models. It is calculated by the total number of correct answers divided by the total number of samples, illustrated in Equation 2.1. A counterpoint to the accuracy evaluation is not to observe the existence or absence of overfitting (ZHU et al., 2010).

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2.1}$$

On the other hand, precision and recall serve different purposes than accuracy and are used together for the F1-Score (ZHU et al., 2010). According to Faceli et al. (2011), precision checks the proportion of items in which ML returned with the correct classification, so it can be applied when the FP is more harmful than the FN for the classification of the algorithms. Equation 2.2 shows its calculation.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{2.2}$$

Recall indicates the number of situations in which the model correctly identified TP, going in the opposite direction to precision, can be better used when FN is more harmful than FP. Equation 2.3 shows its operation (DERCZYNSKI, 2016).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.3}$$

As justified above, for statistical analysis, the F1-Score is a harmonic average between precision and recall, widely used for unbalanced datasets. F1-Score finds its best value when it reaches 1, both precision and recall are perfect, and its worst value at 0. Equation 2.4 shows its traditional formula for calculation (DERCZYNSKI, 2016; FACELI et al., 2011).

$$\text{F1-Score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.4}$$

2.6 Final remarks

Each stage of the ML pipeline has its respective specific functions and difficulties, requiring time for each part of the process. However, data preparation is a more time-consuming stage for specialists since real situations usually bring dirty data to the work environment, requiring extensive analysis and verification of the data.

Currently, data preparation is the stage of the AutoML pipeline with the largest lack in the literature, since the tools assume that the data used is already clean and structured. Nevertheless, ML does not depend solely on the data but on all the techniques employed. Feature engineering extracts the most expressive characteristics in the dataset, selection and configuration algorithm determines the best tools and hyperparameters to achieve better results, and evaluation is fundamental to identify great models.

Subsequently, AutoML is a process that aims to automate each stage of the ML, performing end-to-end automatization that does not need human intervention. Therefore, AutoML is a tool that makes ML more accessible to people who do not have specific knowledge. However, it is still a very recent process, and there are several gaps to be explored, such as the need of significant improvement on automatic data preparation.

Importantly, there is much more content to talk about each AutoML step, but the scope was limited to highlight data preparation techniques.

Chapter 3
META-LEARNING

3.1 Initial considerations

Meta-learning has differences with conventional ML, mainly on its level of adaptation. According to Jankowski, Duch and Grkabczewski (2011), meta-learning is a strategy to understand and use these adaptation properties, using an approach directed to "learning to learn." While, traditionally, ML algorithms work only with the dataset related to the problem that will be analyzed, meta-learning is based on the accumulation of experience from multiple experimental applications (called meta-examples), potentially in different contexts and domains (FACELI et al., 2011).

Meta-examples are used to create a database that will be the basis of all meta-learning, which is called meta-knowledge. The meta-examples are explored and used to derive two types of information, stored as the meta-knowledge: (1) the meta-attributes (or meta-features) of each meta-example; (2) the performance of ML algorithms on each meta-example. So, the new dataset relates its meta-attributes with the information present in the meta-knowledge, determining which algorithm and hyperparameters to use (GUERRA; PRUDÊNCIO; LUDERMIR, 2007).

Meta-knowledge shares significant similarities with conventional datasets. While datasets are separated into attributes and targets, meta-knowledge is separated into meta-attributes and meta-targets. Meta-attributes are information that is undeclared and not formally expressed in each dataset. Therefore, it is implicit in each meta-example (we can mention the number of instances, number of attributes, number of missing values, among other types of data). On the other hand, the meta-targets are the performances that the meta-examples obtained in past training, usually with the meta-target being the best experimental result (BRAZDIL et al., 2008).

In this way, meta-learning is an approach that investigates meta-data[1] to acquire meta-knowledge by mapping the characteristics of models trained in previous experiments (BRAZDIL et al., 2008). Consequently, meta-learning analyzes past results to draw conclusions to support new examples (FACELI et al., 2011).

According to Brazdil, Soares and Costa (2003), the extraction of characteristics (meta-attributes) from the datasets must contain relevant information that facilitates the generalization. Besides, it must be simple and computationally inexpensive. The techniques used for extracting characteristics are covered in the following topics.

3.2 Data characterization strategies

Meta-learning is an approach that uses information that is not explicit in the datasets, Faceli et al. (2011) determine that this information, the meta-attributes, is divided into three distinct classes:

- ***General meta-attributes***: This technique extracts simpler and more descriptive information from the datasets, such as the number of classes or instances.
- ***Statistical meta-attributes***: It is a way of extracting statistical information about the data set, which can be through standard deviation, the correlation between attributes, among others.
- ***Information-theoretic meta-attributes***: This technique aims to quantify the properties of the dataset by assigning concrete values to the data, for example, using entropy or mutual information of attributes.

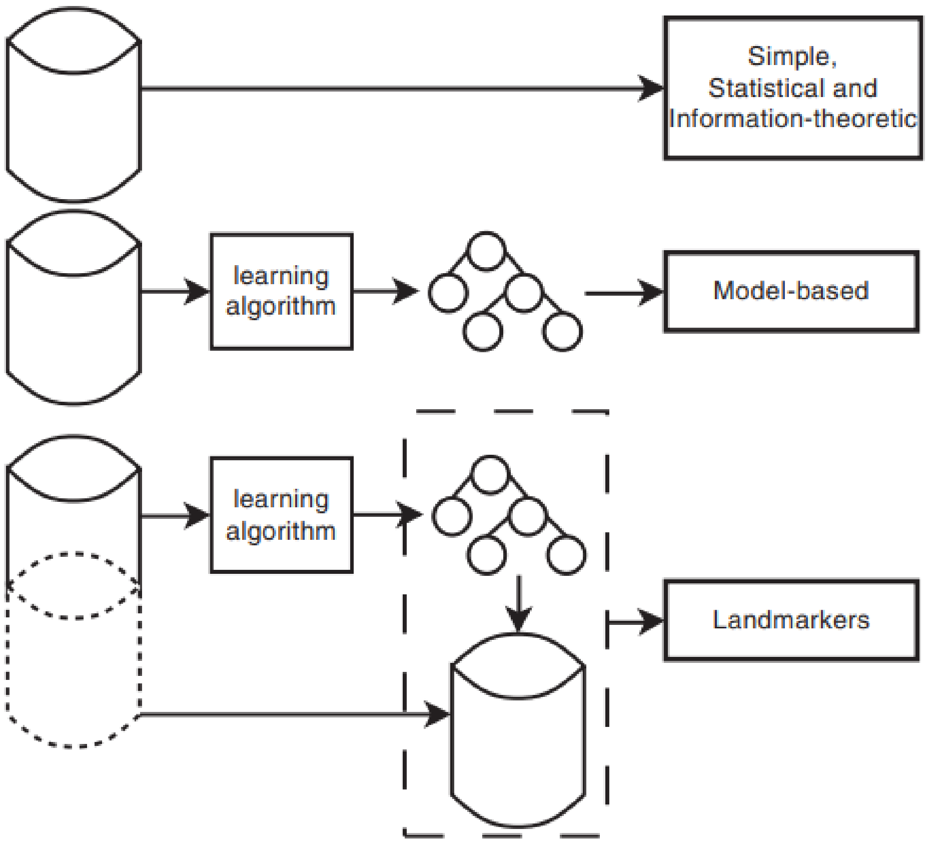
As can be seen in Figure 3.1, there are three different types of approaches to extract meta-attributes: (1) obtaining the characteristics from information that is implicit within the dataset itself; (2) acquiring information according to the structure of the classifiers used; (3) directly considering performance measures to obtain information (BRAZDIL et al., 2008). The following sections will provide more details about each approach.

3.2.1 Statistical and information-theoretic characterization

In order to extract the meta-attributes directly from the datasets, this method tries to characterize the data by establishing simple, statistical, and information-theoretic measures. Meta-

[1]Meta-data is the data that describes the structure of the main dataset, usually implicitly.

Figure 3.1: Approaches to data characterization



Source: Brazdil et al. (2008)

attributes must be defined to describe the main properties of the data set, these being different information from the attributes of the samples (CASTIELLO; CASTELLANO; FANELLI, 2005). A good set of meta-attributes in this category must: (1) help in determining performance; (2) perform the extraction without being complex or expensive (CASTIELLO; FANELLI, 2011).

As can be seen from Table 3.1, adapted from Castiello, Castellano and Fanelli (2005), the tasks aim to map the meta-attributes in order to characterize the data related to a learning problem by establishing simple, statistical and theoretical information.

3.2.2 Model-based characterization

For this method, the characterization is based on the properties of the models applied in the meta-examples (FACELI et al., 2011). Two advantages for this characterization method are: (1) as the characterization is not limited to the distribution of the data, the dataset can incorporate the complexity and assist in the performance of the hypothesis; (2) the resulting model can serve

Table 3.1: Examples of meta-attributes

Simple meta-attributes
Number of dataset instances
Number of dataset attributes
Number of nominal attributes
Number of numeric attributes
Number of output values (classes)
Statistical meta-attributes
Standard deviation of attributes
Coefficient of variation
Canonical correlation analysis
Correlation coefficient
Average kurtosis of attributes
Information-theoretic meta-attributes
Entropy of normalized classes
Entropy of normalized attributes
Joint entropy of class and attributes
Mutual information
Signal-to-noise ratio

as a basis for explaining the reasons behind the performance of the ML algorithm (VILALTA; GIRAUD-CARRIER; BRAZDIL, 2009).

For example, for this model-based characterization method, when using decision trees in the experimental sets, the model properties are used to describe the information: balance of the tree, its shape, depth, number of leaf nodes for each attribute, among other possible information (VILALTA; GIRAUD-CARRIER; BRAZDIL, 2009).

3.2.3 *Landmarking*

This method characterizes the data through the performance of simple ML algorithms, generating close meta-attributes when the performance is similar (FRANÇOIS; WERTZ; VERLEYSEN, 2011; VILALTA; GIRAUD-CARRIER; BRAZDIL, 2009). According to Prudêncio, Souto and Ludermir (2011), some meta-attributes can be very time-consuming. In this way, landmarking is a more economical approach for characterizing meta-examples and providing useful information for the meta-learning process without great computational costs.

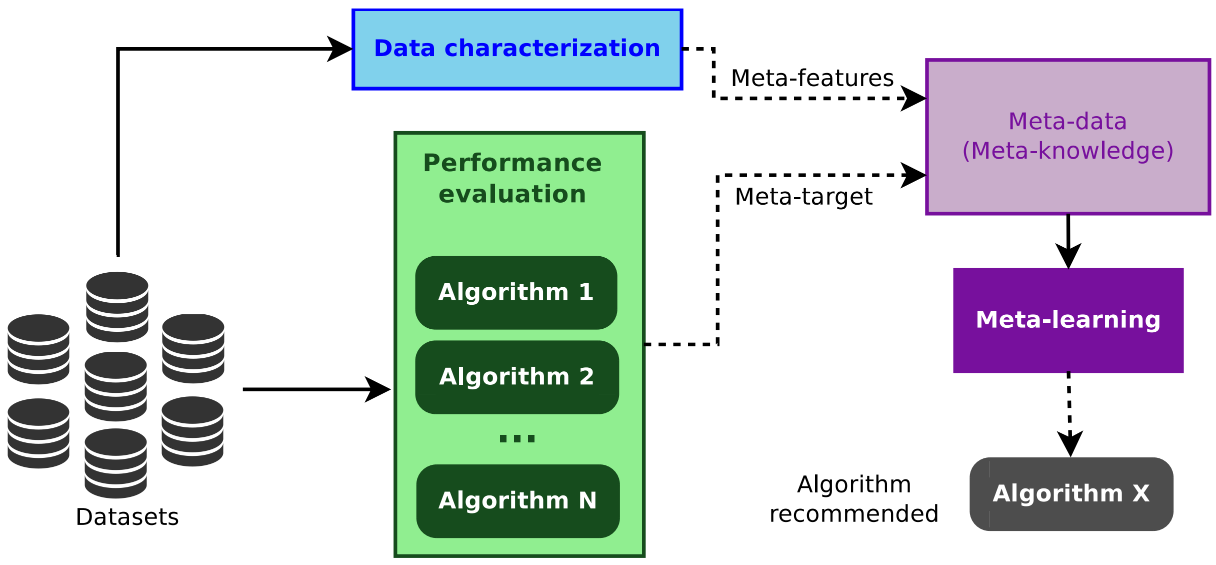
According to Faceli et al. (2011), meta-examples are characterized according to different ML algorithms, called landmarkers, generating relatively similar meta-attributes when the classifier's performance is related. Commonly, for this method, algorithms compatible with different types of data are used.

In this model of characterization of meta-examples, different evaluation measures for ML models can be operated as meta-attributes, such as accuracy, precision, F1-score, among others (FACELI et al., 2011). The description of the evaluation metrics is in Section 2.5.

3.3 Meta-learning application

To summarize, an algorithm recommendation system that uses meta-learning must focus on obtaining good meta-attributes and defining the ML models used. As shown in Figure 3.2, with the repository of datasets (or meta-examples repository), both data characterization and experimental analysis (evaluation) are performed using the desired algorithms. The results are selected to create meta-knowledge, which will give rise to a recommendation system using meta-learning (BRAZDIL et al., 2008). For data preparation, the meta-learning will select the preprocessing methods that best suit the input dataset, seeking the best solutions by looking at past experiences.

Figure 3.2: How meta-learning acquires meta-knowledge to select algorithms



Source: Mantovani (2018)

With all the experiments completed and the meta-knowledge elaborated, this base is used to construct meta-models. Meta-models can learn the relationship between the meta-attributes of the datasets and the desired output. Based on this premise, the meta-model is a great way to achieve rankings of the most promising configurations in an algorithm or even make direct performance predictions (VANSCHOREN, 2018). Usually, meta-models use adaptations of k-nearest neighbors (kNN) for their elaboration, making comparisons of the new entries with the meta-examples belonging to the meta-knowledge (BRAZDIL et al., 2008).

A critical factor for the correct application of meta-learning is good meta-knowledge with varied information, increasing the probability of obtaining good results. In this way, meta-models expect many meta-examples and tests carried out for their elaboration. (VILALTA; GIRAUD-CARRIER; BRAZDIL, 2009; BRAZDIL; SOARES; COSTA, 2003).

3.4 Final remarks

Meta-learning is a very effective technique to assist in decision-making since each possible choice of the pipeline has its weight, relevance, and consequence. In this way, the generation of a set of examples (meta-examples) to use as a basis (meta-knowledge) for judging the best trajectories is an appropriate path, using a wide range of past experiences to acquire learning.

The benefit of meta-learning is the agility to define the next steps adopted for algorithms due to meta-knowledge, making decisions more articulated by not checking every possible situation, going directly into situations closest to the problem analyzed. In the context of data preparation, the application of meta-learning assists in the selection of pre-processing techniques for ML, choosing the most appropriate pipeline for the analyzed dataset. Thus, looking for great solutions without costly architectural searches.

Chapter 4

RELATED WORK

In AutoML context, since each stage of the ML pipeline has its importance and difficulty, there are still few studies in the literature that propose techniques and methods for automating data preparation. Even though, as a general trend, projects aimed at AutoML focus on the selection and configuration of algorithms, and only some studies cover data cleaning. In this chapter, we bring some related works that aim to perform partial or total data preparation.

4.1 Researches about data preparation

In their research, Rahm and Do (2000) addressed the problems and possible approaches to data cleansing, one of the operations performs by data preparation. Initially, the main difficulties encountered were verified, citing missing values, discrepancies, duplicates, among others. Thus, they discussed the importance of the ETL process (extraction, transformation, loading) and the need for data transformations in an integrated and standardized manner. The research brought some commercial 2000s data cleaning tools, but these usually dealt with only specific domains, such as eliminating duplicates or illegal values. The work warns of interoperability problems in these tools, that is, the difficulty in combining the functionality of different methods.

The project was precise about the difficulties encountered in cleaning the data and how the ETL process impacts the final result. As described, the ETL process has three essential steps, which are: (1) Extraction of the values, to verify instance correspondence and elimination of duplication; (2) Validation and correction, used to detect problems and replace missing values or correct incorrect values; (3) Standardization, transforming the data in a standardized interval or effecting the transformation of texts through stemming or stemming.

Zhang, Zhang and Yang (2003) discussed the entire data preparation process, encompassing all the operations that are applied, in addition to emphasizing the intention of companies to perform a good data preparation for for-profit purposes. They show three main points for good data preparation in real situations, namely: (1) real data is essentially impure; (2) ML and data mining systems require quality data and; (3) quality data generates good algorithms.

This work is a good indicator on how and where to conduct your research, since although it does not bring innovations regarding theories or algorithms for data preparation, there was careful care about the desirable paths and contributions to this area of knowledge that, in the period in which the article was written, it was an area that was still emerging as something relevant to the market and academia.

In Krishnan et al. (2016) work for the development of ActiveClean, the study was performed to solve two tasks related to data preparation: outliers removal and attributes transformation. In this way, a framework was created based on pointwise gradients applied to filter potentially noisy data. The developer must initialize ActiveClean with the desired ML model, a featurization function, and the database that will be cleaned. This featurization function works mapping the data into a vector so that the algorithm can analyze it (as OHE for categorical data and bag-of-words for textual data). In each iteration, the framework suggests a sample of data to clean based on the valid data values for the desired ML model and the likelihood of it being noisy.

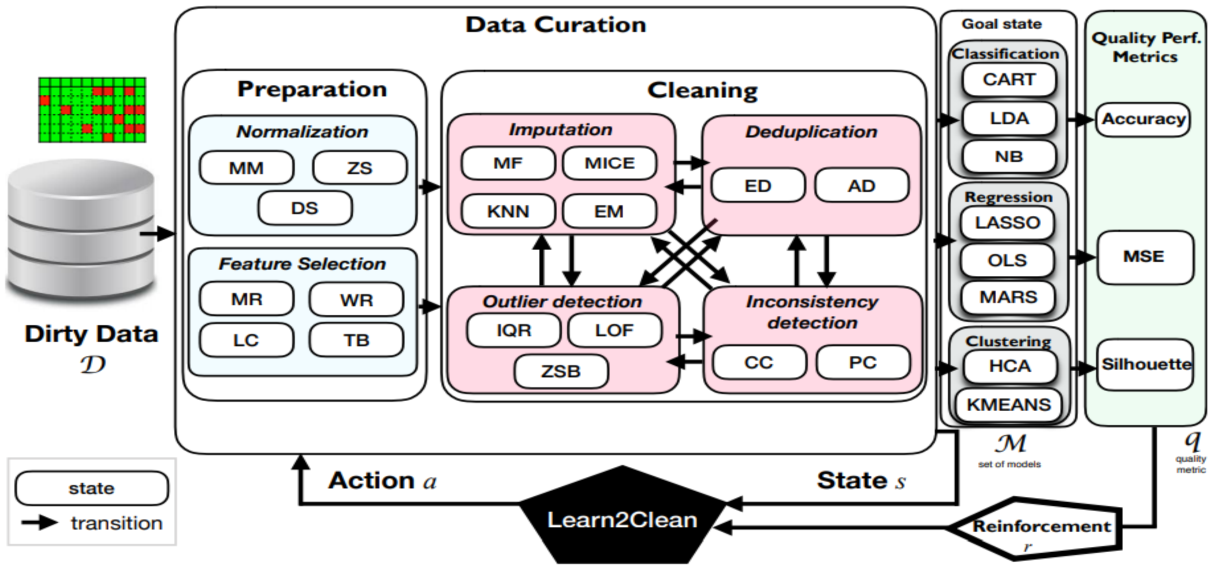
An experimental study by Krishnan et al. (2016) points to promising results, but also some limitations: (1) For some data or in sets that are already partially cleaned, the framework can lead to misleading results; (2) It is an iterative process, what can be computationally expensive; (3) There is the need for specialists since the process is not 100% automatic.

Similar to our project, Bilalli et al. (2016) studied the efficiency of using meta-learning to automate data preparation. This study experimented with 28 meta-attributes, but only 19 proved to be effective. For the experiments, some pre-processing techniques were selected including: treatment of missing values, transformation from nominal to binary, normalization, standardization and discretization.

Despite the good experimental results of Bilalli et al. (2016), attesting to the possibility of using meta-learning for automatic data preparation, their study was only an experimental prototype, so no algorithm or framework was developed. Besides, the authors also propose to rank data preparation and feature engineering techniques, disregarding the dependencies among them or the groups they belong. Consequently, they may recommend, for instance, more than one technique for discretization, which demands a closer look of an expert to decide which one to apply.

With the most complete framework for data preparation, Berti-Equille (2019) developed Learn2Clean, a tool based on Q-Learning, a reinforcement learning algorithm. This is the most complete tool for automatic data preprocessing, with operations for data cleaning, such as imputations, outlier detection, inconsistency detection and deduplication. However, like the previous work, some of the techniques used are not data preparation designed ones and came from the feature engineering stage. When a new dataset is inserted with the type of task (classification, regression or clustering) and the type of evaluation, after a series of cleaning steps such as imputations, normalizations and transformations, the result is applied in Q-Learning and the process is repeated iteratively until the best final solution is found. Figure 4.1[1] illustrates all of these steps.

Figure 4.1: Learn2Clean architecture



Source: Berti-Equille (2019)

The preliminary results of Learn2Clean validated the efficiency of reinforcement learning to automate data preparation, but even with good results, certain problems still remain: (1) There were only a few tests and none of them performed with real data; (2) Some techniques belong to feature engineering stage; (3) The platform doesn't handle categorical data; (4) It's an iterative process, so it can take a long time depending on the dataset.

As an extension of the previous work, Berti-Équille and Comignani (2021) developed CLeanEX to make the techniques used by automated data cleaning explainable. The proposed framework represents each possible technique as a node in a decision tree. In this way, the tree

[1]All the acronyms in the illustration can are in the glossary. For more details of the techniques used, access the work of Berti-Equille (2019).

is composed of branches that produce the cleaning pipelines, which have resources that can be used and exposed to explain the agent's decision.

Among the main advantages of CLeanEX, we can mention that the explanations are understandable for humans with diverse knowledge, and it can be extensible to handle causal reasoning. The framework can explain part or all of the cleaning pipeline, and it can provide a set of explanations regardless of the metric used by the ML model.

4.2 Final remarks

Although data preparation is an essential step for any ML algorithm, few studies in the literature explore data cleaning. Even though, as reported in the related works, some works were willing to carry out the automatization of this stage, demonstrating the possibility of investigation in this area. Research efforts on data preparation are still at an early stage in the context of AutoML, as there are still no consistent methods for automatizing it. The platforms existing in the literature at the moment, despite achieving good preliminary results, still take a long time to be executed.

Chapter 5

DATA PREPARATION PIPELINE RECOMMENDATION SYSTEM

Based on the review of the most related works available in the literature and the main difficulties that still permeate the data preparation, this project proposes a pipeline recommendation method for automatic data preparation using meta-learning. The developed code is open source[1], and the datasets are freely available.

The research was intended to reduce the time currently spent by ML developers when preparing data. Real-world data usually require specific techniques for their correct treatment, which requires significant analysis before entering algorithm configuration.

In this context, our proposal recommends five pipelines for the user, ranked by relevance. In addition, the developer can use the recommended pipelines and correct them manually to generate more robustness to the model. Thus, the reduction of time spent in data preparation can, for example, directly influence the total time spent to define the pipeline. So, the user may spend more time, for instance, on selecting and configuring the learning algorithm.

The methods used for the research were designed based on resources considered to be state-of-the-art. The results obtained in this work are better than those obtained without adequate pre-processing techniques and optimizing the time compared to platforms with similar proposals such as Learn2Clean.

All the necessary resources for the execution of this research project are available at the Federal University of São Carlos (UFSCar) - São Carlos campus. In addition, all the assistance for the correct execution was provided both by the university and the B2W Digital company, partner of this project.

[1]https://github.com/fernandozagatti/metaprep

UFSCar has a wide network of workstations and rooms granted for the elaboration of the project, in addition to providing the necessary programs for its proper development, which are installed and are, in the majority, in the public domain. The institution also guaranteed access to websites and articles that, for the most part, are paid for, which the entire bibliographic reference is available in the library and on the Internet through the institution's contracts with the repositories of articles and academic works.

B2W Digital, as far as it is concerned, conferred computers for the execution of the research project, in addition to authorizing access to its Virtual Private Network[2] (VPN), guaranteeing the permission to connect with real data sets of the company, allowing tests with situations that are regularly faced by companies.

5.1 Methodology

The experiments were performed on a remote server, made available by B2W Digital, with two processors Intel Xeon E3-12xx v2 (Ivy Bridge), two 16GB RAM memory (32GB RAM in total), and 50GB internal storage. The methodology to achieve the objective of this project is described below, in the following phases:

- Selection of data preparation techniques;
- Datasets (meta-examples) collection;
- Meta-knowledge construction (meta-targets and meta-attributes);
- Meta-models elaboration.

5.1.1 Data preparation techniques

The first step in the process was to define the techniques that would be used for data preparation. Thus, based on the techniques most commonly used in ML models, the following techniques were selected:

- **Imputation:** Deletion case, mean, median and most frequent;
- **Categorical-numerical transformation:** One-Hot-Encoding and label encoder;
- **Standardization and normalization:** Standard Scaler, minmax and normalizer;

[2]VPN is a private network built on the infrastructure of a public network.

- **Class balancing:** Oversampling and undersampling.

With the appropriate combinations considering all the techniques in these 4 groups, 180 objective pipelines were defined for data preparation. All pipelines can be viewed in Appendix A.

5.1.2 Datasets collection

The second step was the selection of datasets for the experiments. All datasets were collected from OpenML (FEURER et al., 2019), a collaborative platform that aims to share and develop ML research, and the list of datasets can be viewed from the identifiers present in Table 5.1. The name and description of each dataset, taken from OpenML, can be viewed in Appendix B.

Table 5.1: Datasets taken from OpenML

Identifiers of selected OpenML datasets
15, 29, 179, 188, 443, 452, 455, 473, 802, 839, 897, 930, 966, 990, 1024, 1037, 1053, 1119, 1205, 41526, 41430, 1351, 1354, 1358, 1364, 1367, 1369, 1373, 1375, 1402, 1403, 1453, 1459, 1460, 1468, 1471, 1475, 1479, 1485, 1486, 1489, 1497, 1502, 1503, 1507, 1510, 1525, 1526, 1547, 1549, 1552, 1553, 1557, 1558, 1560, 1568, 1590, 4534, 4538, 4541, 6332, 23380, 23512, 23517, 40536, 575, 40966, 40979, 40981, 40982, 40996, 40999, 41001, 41002, 41003, 41004, 41005, 41007, 41027, 42638

OpenML has a dataset filtering tool, which allows the search for sets with some features. To select the necessary datasets for the experiments, the following filter was initially applied: 1 to 10 (1..10) classes. Observing the need for larger datasets, the filtering was changed to: at least 501 (>500) instances and 1 to 10 (1..10) classes.

The choice of the final datasets took into account some selection criteria such as: to have structured data and to allow the proper application of the selected techniques. Table 5.2 shows a summary of some meta-attributes present in the meta-examples.

5.1.3 Meta-targets

As will be explained in Section 5.2.1, some datasets have characteristics that allow us to prune some pipelines since they would not be useful. By separating datasets into groups, it was unnecessary to go through all 180 pipelines, only those that make sense for that particular group.

Table 5.2: Summarization of meta-examples

	Minimum	Maximum	Mean
Instances	95	1000000	156197,06
Attributes	2	856	58,07
Categorical data	0	61	9,84
Numerical data	0	856	48,22
Classes	2	10	3,60
Instances with missing values	0	7330	448,12
Total missing values on dataset	0	68100	3439,56

For instance, if there are no missing values in our data, there is no reason to apply imputation techniques over it. Using this information, we selected five subsets of the 180 pipelines to save time to construct the meta-knowledge.

Then, each meta-example passes through its selected subset of pipelines, and the resulting cleaned data is used to determine the most effective pipeline. For this, we perform an experiment using Random Forests and ranking the pipelines by the achieved accuracy. The five top-ranked pipelines are used to establish the target of the respective meta-example.

5.1.4 Meta-attributes

For the extraction of the meta-attributes, we used Pymfe (ALCOBAçA et al., 2020), a package for the extraction of meta-features in Python. Pymfe allowed the extraction of 73 statistical and information theory-based meta-attributes, which can be viewed in Appendix C. Note that each dataset does not have 73 meta-attributes. It occurs because in many of the meta-attributes, mainly in statistical meta-attributes, it is necessary to calculate one value per attribute. For example, kurtosis is calculated from each attribute. To resolve this, after extracting these values, Pymfe separates them into kurtosis_md (median) and kurtosis_mean (average).

There was no selection of meta-attributes, extracting all attributes that Pymfe supported; as future work, it is possible to carry out an analysis of the 73 meta-attributes extracted and select only those most relevant to the meta-learning process.

For each dataset, all the extracted meta-attributes and the five best pipelines derived from previous experiments were separated, and a comma-separated values file (*csv*) containing this structured data was created with all the information. This *csv* served as meta-knowledge, which was used to create the meta-model of the recommendation system.

5.1.5 Meta-models

Meta-models are ML models that have been previously trained (offline training) in order to be able to make the appropriate pipeline recommendations based on the experiments performed. We created the meta-models using kNN on the meta-knowledge to identify the most similar datasets according to their characteristics. Specifically, we used the 1NN algorithm to avoid interferences of the hyperparameter of the meta-model in our conclusions for this first experiment. Moreover, the 1NN is intuitive, and we do not need to define a strategy to aggregate the pipelines recommended for more than one meta-example.

The choice of the kNN algorithm is due to the nature of the problem since MetaPrep's objective is to recommend the adequate data preparation pipeline according to the datasets already analyzed, that is, to check which dataset is more similar to the new entry.

In the single-model MetaPrep, all the meta-examples are used in the input dataset for the meta-model. On the other hand, the multiple-model MetaPrep uses disjoint subsets of the meta-knowledge, creating separate models. This and other decisions are more clearly explained in Chapter 5.1.1.

5.2 Recommendation system

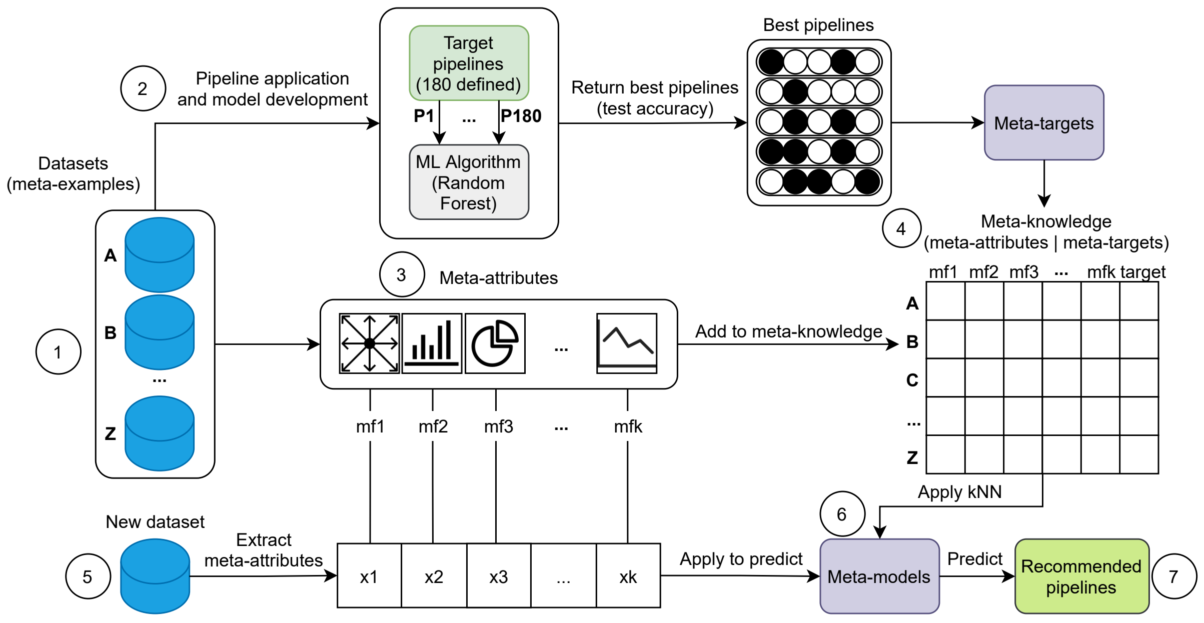
In this scenario, we propose an approach to automate the data preparation process using meta-learning techniques, a system that was called MetaPrep. This proposal consists of pre-training a meta-model capable of predicting the best pipeline for an unprecedented set of data using a large set of experiences in the meta-knowledge, as illustrated in Figure 5.1.

MetaPrep recommends the most suitable data preparation techniques based on previously observed datasets (meta-examples), starting from the training of each dataset in the 180 predefined pipelines, proceeding to extracting meta-attributes and creating meta-knowledge. Then, MetaPrep applies the kNN to create the meta-models that will make the predictions for new entries. In addition, the recommendations made by the method can be adjusted manually by data scientists if they want to.

5.2.1 MetaPrep composition

The recommendation system was developed based on experiments carried out with 80 datasets, also called meta-examples (illustrated as the step 1 in Figure 5.1), and the combi-

Figure 5.1: Our proposal: a meta-learning-based recommender system to predict best pipelines for data preparation



Source: Elaborated by the author

nation of 11 data preparation techniques, totaling 180 possible pipelines. Hence, to create the meta-model for recommending pipelines of data preparation, the system performed two auxiliary tasks: (i) characterizing the meta-example; and (ii) training the models to determine the best data preparation pipeline for each case.

Firstly (from number 2 to 4 in Figure 5.1), the 180 pipelines were applied to each of the 80 meta-examples. After performing all the data preparation techniques specified in a pipeline, a Random Forest model was trained on the refined dataset and tested using a holdout procedure. The five pipelines with the best model's accuracy become the meta-targets for each meta-example.

Then (from number 3 to 4 in Figure 5.1), we extracted the characteristics of each meta-example. These characteristics, called meta-attributes, are information that are implicitly presented in each dataset, such as the number of unique class labels, examples, and attributes, and statistics from each attribute, such as standard deviation, kurtosis, among others. In total, we used 73 meta-attributes to describe each meta-example.

From the collection of meta-attributes and meta-targets, the meta-knowledge (illustrated as the step 4 in Figure 5.1) was created as the basis for the meta-models that are the core of our pipeline recommendation system for data preparation. The creation of the meta-models was done through kNN to identify the meta-examples closest to a new entry.

We observe that this procedure may lead to identical results of several pipelines depending on each dataset's characteristics. For instance, if a dataset has no categorical attributes, all the pipelines that include categorical-numerical transformations have the same outcome among them and pipelines with the same techniques but this specific transformation. For example, the pipelines Standard Scaler ->Oversampling and One-hot-encoding ->Standard Scaler ->Oversampling are equivalent for datasets with no categorical values.

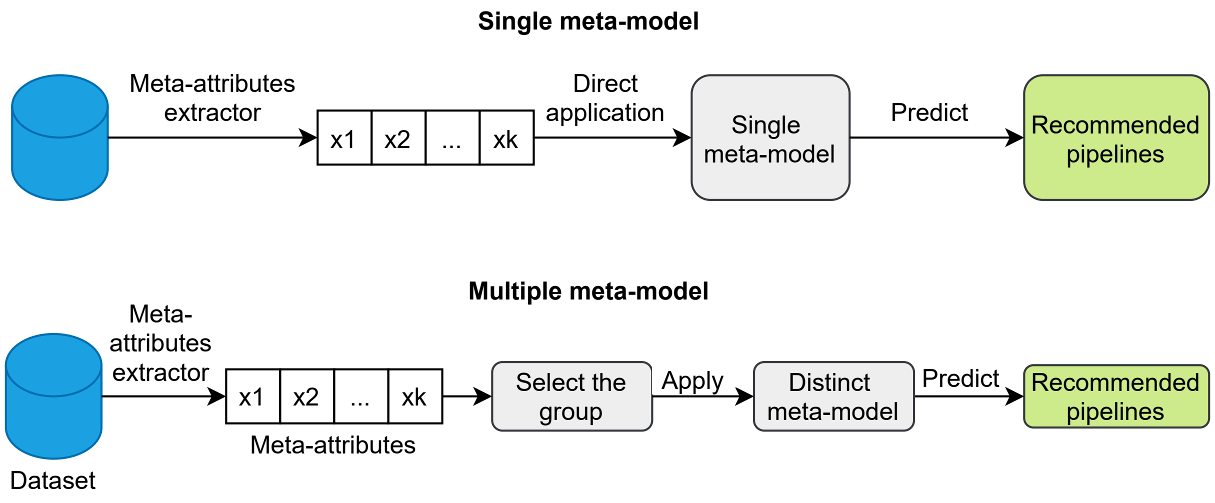
To reduce this issue's influence, we propose a variation of our method, which includes a pipeline subset selection step. The advantages of avoiding this kind of replications are twofold: (i) it reduces the number of classes the meta-model needs to predict without losing relevant pipelines; and (ii) it avoids the recommendation of a pipeline missing essential techniques. For instance, if a dataset contains missing values, we use a meta-model that assuredly recommends an imputation technique. Otherwise, the meta-model may recommend a pipeline that ignores this issue. Alternatively, if the dataset does not contain missing values, we apply a meta-model trained to predict pipelines that do not include imputation techniques.

To define the subset of pipelines, we grouped the datasets in five categories, considering the characteristics of the datasets. Consequently, we create five meta-models. In this new version, categories are mutually exclusive, then each new entry will use only one of the meta-models. The characteristics used to separate the datasets are the following:

1. ***Dataset without categorical data and missing values:*** no imputation or transformation is required.
2. ***Dataset without categorical data but with missing values:*** imputation of missing values is necessary, transformation techniques are discarded.
3. ***Dataset without missing values but with categorical data:*** only the transformation of categorical values is necessary, imputation is discarded.
4. ***Dataset with categorical data and missing values:*** this case takes both imputation and transformation, and since there are nulls in the categorical data, the imputation needs to be compatible.
5. ***Dataset with categorical data but with missing values only in numeric columns:*** this case takes both imputation and transformation, and since there are nulls only in the numerical data, it can be any type of imputation.

Considering these groups, we propose and compare the MetaPrep in two versions: single and multiple meta-models. Figure 5.2 illustrates these variations. In the single meta-model version, when the recommendation system receives a new dataset for automatic preparation, it is characterized, compared with the meta-knowledge (created using all the meta-examples). The data preparation pipelines are suggested based on the meta-targets of the most similar meta-example. The multiple meta-models version works as following. The recommendation system starts by extracting the meta-features to characterize the new dataset. Then, it checks which of the five meta-models (developed for each previously mentioned group) fit the new dataset better. Finally, instead of comparing the new entry with the entire meta-knowledge base, it is compared only with the same group's meta-examples.

Figure 5.2: Difference between single (*top*) and multiple meta-models (*bottom*) MetaPrep



Source: Elaborated by the author

5.2.2 Prediction of a new case

Once a new dataset is presented to the system, it needs to recommend a data preparation to apply to it. The new input dataset passes through the meta-attributes extraction and is used as input by the meta-models.

The extraction of meta-attributes has a special parameter called ”transformer”; by default the algorithm uses ”gray”, where all categorical-type data will be binarized with a model matrix strategy. However, it is possible to use ”None”, where the categorical attributes are not transformed and are ignored during characterization. Although the default method is more assertive because it can characterize the entire dataset without losing information, a lot of RAM is consumed in the process. For cases of low available memory, the second method is a viable alternative.

After extracting meta-attributes, there is an intermediate step to choose which meta-model will be used if we apply the multiple meta-models approach. The meta-models then predict the meta-examples that most resemble this new entry and recommend a list of five pipelines for this new dataset. We consider that, for an inexperienced user, choosing the top-ranked recommendation is a proper procedure. We used this assumption to evaluate our proposal.

Chapter 6

EXPERIMENTAL EVALUATION

The two approaches for recommending data preparation pipelines were implemented and tested in the 80 meta-examples, training each meta-example in each of the 180 pipelines and taking the five ones with the best accuracy. In this way, the recommendation system has a total of 400 possible recommendations.

Of the 180 pipelines applied in the experiments, only 79 appeared at least once among the 400 recommendations of the system. In addition, an analysis regarding the techniques used was performed:

- Among the 400 recommendations, 155 had an imputation technique:
  - Imputation case deletion: 59
  - Imputation mean: 13
  - Imputation median: 18
  - Imputation most frequent: 65
- Among the 400 recommendations, 245 had a symbolic-numeric conversion technique:
  - Label encoder: 92
  - One-hot-encoding: 153
- Among the 400 recommendations, 320 had a standardization or normalization technique:
  - Minmax: 123
  - Standard scaler: 127
  - Normalizer: 70

- Among the 400 recommendations, 362 had a class balancing technique:
  - Undersampling: 39
  - Oversampling: 323

6.1 Tests performed on open datasets

The tests were applied to 4 different datasets, which were extracted from Kaggle[1], another online platform for data scientists to publish datasets and create and explore ML models. The selected sets were:

- Titanic - Machine Learning from Disaster[2]
  - Dataset with categorical data, numerical data and missing values.
- Tic-Tac-Toe Endgame Data Set[3]
  - Dataset with only categorical data.
- Bike Buyers 1000[4]
  - Dataset with categorical and numerical data.
- Cardiovascular Disease dataset[5]
  - Dataset with only numerical data.

To compare the impact of different data preparation pipelines we performed three types of comparisons: (1) the direct application of datasets in an AutoML platform (in this case, Auto-sklearn was chosen), (2) the application in Learn2Clean and (3) the application in the two approaches proposed for our recommendation system.

Our two approaches obtained very similar results. However, as can be seen in Table 6.1, the multiple meta-models approach led to better recommendations. The approach for recommending a single meta-model often caused confusion regarding what it should recommend. These

[1]https://www.kaggle.com/
[2]https://www.kaggle.com/c/titanic
[3]https://www.kaggle.com/rsrishav/tictactoe-endgame-data-set
[4]https://www.kaggle.com/heeraldedhia/bike-buyers
[5]https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

cases could be checked in the tests, in which the single meta-model did not recommend imputations to the Titanic dataset and recommended unnecessary transformations in the cardiovascular disaster dataset.

Table 6.1: Pipelines used by each tool

Titanic - Machine Learning from Disaster	
Autosklearn	Default preprocessor
Learn2Clean	Local Outlier Factor
Single meta-model	Label Encoder ->Standard Scaler -> Oversampling
Multiple meta-models	Case deletion ->Label Encoder -> Standard Scaler ->Oversampling
Tic-Tac-Toe Endgame Data Set	
Autosklearn	Default preprocessor
Learn2Clean	N/A
Single meta-model	Imputation mean ->One-hot-encoding -> Standard Scaler ->Oversampling
Multiple meta-models	Label Encoder ->Normalizer -> Oversampling
Bike Buyers 1000	
Autosklearn	Default preprocessor
Learn2Clean	Z-score-based method ->Exact duplicate
Single meta-model	One-hot-encoding ->Standard Scaler -> Oversampling
Multiple meta-models	One-hot-encoding ->Standard Scaler -> Oversampling
Cardiovascular Disease dataset	
Auto-sklearn	Default preprocessor
Learn2Clean	Local Outlier Factor
Single meta-model	One-hot-encoding ->Standard Scaler -> Oversampling
Multiple meta-models	Standard Scaler ->Oversampling

Both Auto-sklearn, which uses one default preprocessing (Imputation mean, One-Hot-Encoding and Standard Scaler), and the Learn2Clean, which uses reinforcement learning, had difficulty handling some datasets. Auto-sklearn requires a lot of processing to carry out its training and its default preprocessing may not be ideal, while Learn2Clean cannot handle categorical data, failing to recommend a pipeline for the dataset with only categorical data (Tic-Tac-Toe).

Table 6.2 brings the accuracy and execution time for each experiment, now evaluating an end-to-end pipeline: cleaning the data, applying ML and evaluating the resulting model.

Table 6.2: Results of the end-to-end evaluation

		Total time	Accuracy
Autosklearn	**Titanic**	N/A	N/A
	Tic-Tac-Toe	N/A	N/A
	Bike Buyers	N/A	N/A
	Cardio	N/A	N/A
Learn2Clean	**Titanic**	405s	72,04%
	Tic-Tac-Toe	N/A	N/A
	Bike Buyers	359s	68,16%
	Cardio	>1800s	73,24%
Single meta-model	**Titanic**	N/A	N/A
	Tic-Tac-Toe	2.3s	98,06%
	Bike Buyers	3.9s	71,72%
	Cardio	17.9s	72,87%
Multiple meta-models	**Titanic**	7.8s	84,14%
	Tic-Tac-Toe	2.3s	98,06%
	Bike Buyers	3.9s	71,72%
	Cardio	17.9s	72,87%

Auto-sklearn was unable to perform the complete pipeline on any of the datasets, as it demands a lot of processing by the hardware, causing the kernel to halt. In parallel, Learn2Clean performs the drop of categorical columns, dealing only with numerical values in its training, and because of that it was unable to train the Tic-Tac-Toe dataset.

In contrast, the two approaches proposed in this work performed its execution in a short time, managing to recommend pipelines in seconds. As for the accuracy of the recommendations, the single meta-model had some difficulty, while the multiple meta-models variation managed to identify the techniques that should be used and those that should be excluded.

The tests demonstrated the efficiency of meta-learning for recommending pipelines, with an emphasis on the approach using multiple meta-models, being able to recommend the necessary techniques in a very short time.

6.2 Case study: people analytics

In addition to the evaluation on open datasets, we carried out a case study to verify the performance of MetaPrep on real data. Through an experimental evaluation on a dataset with "real-world problems," it is possible to evaluate the potential limitations of MetaPrep.

For this purpose, B2W Digital, an e-commerce company in Latin America, provided a dataset that is naturally dirty and could be used for this study. Due to the conditions of confidentiality, the data cannot be made publicly available and will be explained at a high level.

The dataset comprises real data from the company's employees. The purpose of the dataset is to predict the reason for leaving the company (e.g., resignation, termination of the contract, unfair dismissal, just cause). Therefore, it contains some information regarding their permanence and behavior while they were active in the organization. Among some attributes of the dataset, we can mention identification number, employment status, department, occupation, date of admission, number of completed courses, and educational level as some examples. Table 6.3 shows some meta-attributes present in this dataset.

Table 6.3: Some meta-attributes of the raw dataset

Meta-feature	Value
Number of instances	70043
Number of attributes	45
Attributes with numeric values	29
Attributes with categorical values	16
Instances with missing values	70043
Total missing values	605376

We performed all the experiments using the default parameters of MetaPrep. The initial tests were performed on raw data, that is, the dataset was applied directly to MetaPrep without performing any type of analysis in order to verify how the system behaved in situations with real data.

Initially, MetaPrep could not recommend pipelines for this specific dataset, which led us to a more in-depth analysis. By observing the dataset and the system's behavior, we could conclude that MetaPrep is not prepared to deal with datasets with low-information attributes. More specifically, attributes that have mostly missing values.

We need to note that Pymfe, the tool used to characterize the data for meta-learning, cannot deal with missing values. Our system adopts the exclusion of these instances, which can make the analysis more difficult. Because the dataset used has at least three attributes predominantly with missing values (approximately 68,000 of the 70,000 are null in each attribute), all instances would be excluded. Consequently, we would have no ways to proceed with the characterization. Besides, the dataset has redundant attributes. For example, it has the attributes ”id_occupation” and ”occupation”, equivalent to each other.

Thus, a second test was performed, manually excluding low-information attributes before applying the dataset in MetaPrep. Firstly, only those attributes that had at least 45% of the missing values were excluded. Table 6.4 shows some meta-attributes of the resulting dataset after excluding these columns, being able to visualize the differences between the new dataset and the raw data, mainly regarding the missing values.

Table 6.4: Comparison between the raw dataset and the second dataset

Meta-feature	**Before**	**After**
Number of instances	70043	70043
Number of attributes	45	36
Attributes with numeric values	29	25
Attributes with categorical values	16	11
Instances with missing values	70043	13996
Total missing values	605376	32090

Even after excluding these attributes, MetaPrep was unable to recommend pipelines. However, in this case, the limitation occurred due to hardware limitations. Since the default parameters of MetaPrep lead to high consumption of RAM and there were many categorical features to be characterized, the memory overflowed.

For the third and last test, a new dataset was built over the data used in the second experiment. Attributes that are also low-informative or have too many values to discriminate the class label, such as ids, medical records, and ill-annotated dates, were identified and removed from the dataset. Table 6.5 summarizes the difference between the meta-attributes of the new dataset compared to the raw dataset.

Table 6.5: Comparison between the raw dataset and the third dataset

Meta-feature	**Before**	**After**
Number of instances	70043	70043
Number of attributes	45	23
Attributes with numeric values	29	20
Attributes with categorical values	16	3
Instances with missing values	70043	3560
Total missing values	605376	5447

The third experiment obtained good results, managing to recommend the pipelines according to the information in the dataset. MetaPrep took a total of 356 seconds to complete the recommendation, a much higher value if compared to the experiments with open datasets. It happened mainly due to the volume of data (examples and attributes) in the dataset being higher than the Kaggle sets. Another possible factor is the high number of categorical attributes, making it difficult to process and characterize the data.

Despite having a longer processing time, the MetaPrep, using the method with multiple meta-models, managed to make a recommendation compatible with the dataset, mainly identifying the need of cleaning missing values and converting categorical-numeric. The top five recommended pipelines were, respectively:

1. Imputation most frequent ->One-hot-encoding ->Normalizer ->Oversampling
2. Imputation most frequent ->One-hot-encoding ->Standard Scaler ->Oversampling
3. Imputation most frequent ->One-hot-encoding ->MinMax ->Oversampling
4. Imputation most frequent ->One-hot-encoding ->Oversampling
5. Case deletion ->Label Encoder ->MinMax ->Oversampling

From the datasets used, Random Forest was applied to verify if the algorithm can train the data and observe its accuracy. Table 6.6 shows the results obtained. It is important to note that the raw dataset is the data without any changes, while dataset 3 is the set with the manual exclusion of low information (null and IDs data) and that MetaPrep can be applied.

Table 6.6: Results with Random Forest

Datasets	Accuracy
Raw dataset	N/A
Dataset 3	N/A
Dataset 3 + Pipeline 1	80,93%
Dataset 3 + Pipeline 2	80,92%
Dataset 3 + Pipeline 3	80,82%
Dataset 3 + Pipeline 4	80,92%
Dataset 3 + Pipeline 5	79,04%

It is possible to observe that the Random Forest algorithm cannot apply the training to the raw data or the data after the manual exclusion, since both datasets still had inconsistencies such as missing values and categorical data. In addition, despite the results in the pipeline tests being similar, we can observe a growing increase in accuracy as the rank goes up to the first position.

Chapter 7
CONCLUSIONS

This work addressed a current and real need to automate the data preparation on AutoML platforms, which cannot handle naturally noisy data. In this context, the preliminary results showed the applicability of meta-learning for selecting data preparation techniques, managing to recommend pipelines in very short times, and having the possibility of being customized by the developer.

Although the initial results are promising, it is still necessary to conduct a more accurate analysis regarding the attributes of low-variance. As seen in the case study, many missing values can hinder the data characterization and not allow the recommendation of the pipelines. Another point to note is the hardware since some datasets can cause the RAM to overflow. Even though MetaPrep has the parameter “transformer,” as explained in Section 5.2.2, methods to optimize the algorithm without activating this configuration are an ideal way.

7.1 Main contributions

We propose MetaPrep, a method to automate the data preparation process using meta-learning. Our proposal consists of pre-training a meta-model capable of predicting the best pipelines for a previously unseen dataset using a large set of varied datasets. The main contributions of this work are:

- We present a novel meta-learning-based algorithm to automate the data preparation for ML. Although we found a similar idea in the literature, our method is the first to recommend the entire data preparation pipeline based on meta-learning. Besides, we implemented our approach to be easily used in other applications and adapted it to work with many ML platforms with little effort.

- To the best of our knowledge, the state-of-the-art method to automate data preparation is the reinforcement learning-based Learn2Clean. Our method achieves accuracy rates similar to Learn2Clean (BERTI-EQUILLE, 2019), but with a significantly reduced runtime.
- As we recommend a set of five data preparation pipelines, our method is effective for data scientists with different experience levels. While beginners may use the best-ranked pipeline as an out-of-the-box recommendation, more experienced researchers or practitioners may use their knowledge to decide one of the five recommendations and make their modifications.

7.2 Future work

As a future work, it is still possible to execute a more in-depth analysis in relation to the 5 best pipelines chosen by each meta-example, and this choice can be made with other evaluation methods such as F1-Score or ROC Curve, not just accuracy, obtaining pipelines with less overfitting.

As seen in the case study, a possible extension of MetaPrep is the addition of a module for automatic verification of attributes with low-information (for example, identifiers and columns with many empty values), which would be able to make recommendations more easily and assertively. In addition, extending the analysis to other types of data, such as datetimes, will allow better use of the data for the data preparation.

Further, it is possible to carry out a study in relation to the 79 pipelines that were used at least once and the reason why the remaining 101 were not chosen. We also study the possibility of adding more datasets and more preprocessing techniques, making the meta-models more complete and accurate. On the other hand, a more costly extension is the analysis for more ML algorithms, opening the possibility for the system to carry out two queries, recommending the data preparation pipeline and the algorithm for the user.

Appendices

Appendix A

PIPELINES-TARGET

Listed here are all 180 pipelines-target from the combination of the data preparation techniques defined in Section 5.1.1.

1. No preparation
2. Deletion case
3. Mean imputation
4. Median imputation
5. Most frequent imputation
6. Label Encoder
7. One-Hot-Encoding
8. MinMax
9. StandardScaler
10. Normalizer
11. Undersampling
12. Oversampling
13. Deletion case ->Label Encoder
14. Deletion case ->One-Hot-Encoding
15. Mean imputation ->Label Encoder
16. Mean imputation ->One-Hot-Encoding
17. Median imputation ->Label Encoder
18. Median imputation ->One-Hot-Encoding
19. Most frequent imputation ->Label Encoder
20. Most frequent imputation ->One-Hot-Encoding
21. Deletion case ->MinMax

22. Deletion case ->StandardScaler
23. Deletion case ->Normalizer
24. Mean imputation ->MinMax
25. Mean imputation ->StandardScaler
26. Mean imputation ->Normalizer
27. Median imputation ->MinMax
28. Median imputation ->StandardScaler
29. Median imputation ->Normalizer
30. Most frequent imputation ->MinMax
31. Most frequent imputation ->StandardScaler
32. Most frequent imputation ->Normalizer
33. Deletion case ->Undersampling
34. Deletion case ->Oversampling
35. Mean imputation ->Undersampling
36. Mean imputation ->Oversampling
37. Median imputation ->Undersampling
38. Median imputation ->Oversampling
39. Most frequent imputation ->Undersampling
40. Most frequent imputation ->Oversampling
41. Label Encoder ->MinMax
42. Label Encoder ->StandardScaler
43. Label Encoder ->Normalizer
44. One-Hot-Encoding ->MinMax
45. One-Hot-Encoding ->StandardScaler
46. One-Hot-Encoding ->Normalizer
47. Label Encoder ->Undersampling
48. Label Encoder ->Oversampling
49. One-Hot-Encoding ->Undersampling
50. One-Hot-Encoding ->Oversampling
51. MinMax ->Undersampling
52. MinMax ->Oversampling
53. StandardScaler ->Undersampling
54. StandardScaler ->Oversampling
55. Normalizer ->Undersampling
56. Normalizer ->Oversampling

57. Deletion case ->Label Encoder ->MinMax
58. Deletion case ->Label Encoder ->StandardScaler
59. Deletion case ->Label Encoder ->Normalizer
60. Deletion case ->One-Hot-Encoding ->MinMax
61. Deletion case ->One-Hot-Encoding ->StandardScaler
62. Deletion case ->One-Hot-Encoding ->Normalizer
63. Mean imputation ->Label Encoder ->MinMax
64. Mean imputation ->Label Encoder ->StandardScaler
65. Mean imputation ->Label Encoder ->Normalizer
66. Mean imputation ->One-Hot-Encoding ->MinMax
67. Mean imputation ->One-Hot-Encoding ->StandardScaler
68. Mean imputation ->One-Hot-Encoding ->Normalizer
69. Median imputation ->Label Encoder ->MinMax
70. Median imputation ->Label Encoder ->StandardScaler
71. Median imputation ->Label Encoder ->Normalizer
72. Median imputation ->One-Hot-Encoding ->MinMax
73. Median imputation ->One-Hot-Encoding ->StandardScaler
74. Median imputation ->One-Hot-Encoding ->Normalizer
75. Most frequent imputation ->Label Encoder ->MinMax
76. Most frequent imputation ->Label Encoder ->StandardScaler
77. Most frequent imputation ->Label Encoder ->Normalizer
78. Most frequent imputation ->One-Hot-Encoding ->MinMax
79. Most frequent imputation ->One-Hot-Encoding ->StandardScaler
80. Most frequent imputation ->One-Hot-Encoding ->Normalizer
81. Deletion case ->Label Encoder ->Undersampling
82. Deletion case ->Label Encoder ->Oversampling
83. Deletion case ->One-Hot-Encoding ->Undersampling
84. Deletion case ->One-Hot-Encoding ->Oversampling
85. Mean imputation ->Label Encoder ->Undersampling
86. Mean imputation ->Label Encoder ->Oversampling
87. Mean imputation ->One-Hot-Encoding ->Undersampling
88. Mean imputation ->One-Hot-Encoding ->Oversampling
89. Median imputation ->Label Encoder ->Undersampling
90. Median imputation ->Label Encoder ->Oversampling
91. Median imputation ->One-Hot-Encoding ->Undersampling

92. Median imputation ->One-Hot-Encoding ->Oversampling
93. Most frequent imputation ->Label Encoder ->Undersampling
94. Most frequent imputation ->Label Encoder ->Oversampling
95. Most frequent imputation ->One-Hot-Encoding ->Undersampling
96. Most frequent imputation ->One-Hot-Encoding ->Oversampling
97. Deletion case ->MinMax ->Undersampling
98. Deletion case ->MinMax ->Oversampling
99. Deletion case ->StandardScaler ->Undersampling
100. Deletion case ->StandardScaler ->Oversampling
101. Deletion case ->Normalizer ->Undersampling
102. Deletion case ->Normalizer ->Oversampling
103. Mean imputation ->MinMax ->Undersampling
104. Mean imputation ->MinMax ->Oversampling
105. Mean imputation ->StandardScaler ->Undersampling
106. Mean imputation ->StandardScaler ->Oversampling
107. Mean imputation ->Normalizer ->Undersampling
108. Mean imputation ->Normalizer ->Oversampling
109. Median imputation ->MinMax ->Undersampling
110. Median imputation ->MinMax ->Oversampling
111. Median imputation ->StandardScaler ->Undersampling
112. Median imputation ->StandardScaler ->Oversampling
113. Median imputation ->Normalizer ->Undersampling
114. Median imputation ->Normalizer ->Oversampling
115. Most frequent imputation ->MinMax ->Undersampling
116. Most frequent imputation ->MinMax ->Oversampling
117. Most frequent imputation ->StandardScaler ->Undersampling
118. Most frequent imputation ->StandardScaler ->Oversampling
119. Most frequent imputation ->Normalizer ->Undersampling
120. Most frequent imputation ->Normalizer ->Oversampling
121. Label Encoder ->MinMax ->Undersampling
122. Label Encoder ->MinMax ->Oversampling
123. Label Encoder ->StandardScaler ->Undersampling
124. Label Encoder ->StandardScaler ->Oversampling
125. Label Encoder ->Normalizer ->Undersampling
126. Label Encoder ->Normalizer ->Oversampling

127. One-Hot-Encoding ->MinMax ->Undersampling
128. One-Hot-Encoding ->MinMax ->Oversampling
129. One-Hot-Encoding ->StandardScaler ->Undersampling
130. One-Hot-Encoding ->StandardScaler ->Oversampling
131. One-Hot-Encoding ->Normalizer ->Undersampling
132. One-Hot-Encoding ->Normalizer ->Oversampling
133. Deletion case ->Label Encoder ->MinMax ->Undersampling
134. Deletion case ->Label Encoder ->MinMax ->Oversampling
135. Deletion case ->Label Encoder ->StandardScaler ->Undersampling
136. Deletion case ->Label Encoder ->StandardScaler ->Oversampling
137. Deletion case ->Label Encoder ->Normalizer ->Undersampling
138. Deletion case ->Label Encoder ->Normalizer ->Oversampling
139. Deletion case ->One-Hot-Encoding ->MinMax ->Undersampling
140. Deletion case ->One-Hot-Encoding ->MinMax ->Oversampling
141. Deletion case ->One-Hot-Encoding ->StandardScaler ->Undersampling
142. Deletion case ->One-Hot-Encoding ->StandardScaler ->Oversampling
143. Deletion case ->One-Hot-Encoding ->Normalizer ->Undersampling
144. Deletion case ->One-Hot-Encoding ->Normalizer ->Oversampling
145. Mean imputation ->Label Encoder ->MinMax ->Undersampling
146. Mean imputation ->Label Encoder ->MinMax ->Oversampling
147. Mean imputation ->Label Encoder ->StandardScaler ->Undersampling
148. Mean imputation ->Label Encoder ->StandardScaler ->Oversampling
149. Mean imputation ->Label Encoder ->Normalizer ->Undersampling
150. Mean imputation ->Label Encoder ->Normalizer ->Oversampling
151. Mean imputation ->One-Hot-Encoding ->MinMax ->Undersampling
152. Mean imputation ->One-Hot-Encoding ->MinMax ->Oversampling
153. Mean imputation ->One-Hot-Encoding ->StandardScaler ->Undersampling
154. Mean imputation ->One-Hot-Encoding ->StandardScaler ->Oversampling
155. Mean imputation ->One-Hot-Encoding ->Normalizer ->Undersampling
156. Mean imputation ->One-Hot-Encoding ->Normalizer ->Oversampling
157. Median imputation ->Label Encoder ->MinMax ->Undersampling
158. Median imputation ->Label Encoder ->MinMax ->Oversampling
159. Median imputation ->Label Encoder ->StandardScaler ->Undersampling
160. Median imputation ->Label Encoder ->StandardScaler ->Oversampling
161. Median imputation ->Label Encoder ->Normalizer ->Undersampling

162. Median imputation ->Label Encoder ->Normalizer ->Oversampling
163. Median imputation ->One-Hot-Encoding ->MinMax ->Undersampling
164. Median imputation ->One-Hot-Encoding ->MinMax ->Oversampling
165. Median imputation ->One-Hot-Encoding ->StandardScaler ->Undersampling
166. Median imputation ->One-Hot-Encoding ->StandardScaler ->Oversampling
167. Median imputation ->One-Hot-Encoding ->Normalizer ->Undersampling
168. Median imputation ->One-Hot-Encoding ->Normalizer ->Oversampling
169. Most frequent imputation ->Label Encoder ->MinMax ->Undersampling
170. Most frequent imputation ->Label Encoder ->MinMax ->Oversampling
171. Most frequent imputation ->Label Encoder ->StandardScaler ->Undersampling
172. Most frequent imputation ->Label Encoder ->StandardScaler ->Oversampling
173. Most frequent imputation ->Label Encoder ->Normalizer ->Undersampling
174. Most frequent imputation ->Label Encoder ->Normalizer ->Oversampling
175. Most frequent imputation ->One-Hot-Encoding ->MinMax ->Undersampling
176. Most frequent imputation ->One-Hot-Encoding ->MinMax ->Oversampling
177. Most frequent imputation ->One-Hot-Encoding ->StandardScaler ->Undersampling
178. Most frequent imputation ->One-Hot-Encoding ->StandardScaler ->Oversampling
179. Most frequent imputation ->One-Hot-Encoding ->Normalizer ->Undersampling
180. Most frequent imputation ->One-Hot-Encoding ->Normalizer ->Oversampling

Appendix B

All datasets description

Here are all the datasets used by the experiments, reported in Section 5.1.2, well as their respective descriptions.

ID	Dataset name	Dataset description
15	breast-w	Breast Cancer Wisconsin (Original) Data Set. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The target feature records the prognosis (malignant or benign).
29	credit-approval	This dataset is interesting because there is a good mix of attributes – continuous, nominal with small numbers of values, and nominal with larger numbers of values. There are also a few missing values.

179	adult	Prediction task is to determine whether a person makes over 50K a year. Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))
188	eucalyptus	The objective was to determine which seedlots in a species are best for soil conservation in seasonally dry hill country. Determination is found by measurement of height, diameter by height, survival, and other contributing factors.
443	analcatdata_broadway	analcatdata A collection of data sets used in the book ”Analyzing Categorical Data,” by Jeffrey S. Simonoff, Springer-Verlag, New York, 2003. The submission consists of a zip file containing two versions of each of 84 data sets, plus this README file. Each data set is given in comma-delimited ASCII (.csv) form, and Microsoft Excel (.xls) form.

452	analcatdata_broadwaymult	analcatdata A collection of data sets used in the book ”Analyzing Categorical Data,” by Jeffrey S. Simonoff, Springer-Verlag, New York, 2003. The submission consists of a zip file containing two versions of each of 84 data sets, plus this README file. Each data set is given in comma-delimited ASCII (.csv) form, and Microsoft Excel (.xls) form.
455	cars	The Committee on Statistical Graphics of the American Statistical Association (ASA) invites you to participate in its Second (1983) Exposition of Statistical Graphics Technology. The purposes of the Exposition are (l) to provide a forum in which users and providers of statistical graphics technology can exchange information and ideas and (2) to expose those members of the ASA community who are less familiar with statistical graphics to its capabilities and potential benefits to them. The Exposition wil1 be held in conjunction with the Annual Meetings in Toronto, August 15-18, 1983 and is tentatively scheduled for the afternoon of Wednesday, August 17.

473	cjs	The effects of the Growth Regulators Paclobutrazol (PP 333) and Flurprimidol (EL-500) on the Number and Length of Internodes in Terminal Sprouts Formed on Trimmed Silver Maple Trees.
802	pbcseq	Binarized version of the original data set (see version 1). It converts the numeric target feature to a two-class nominal target feature by computing the mean and classifying all instances with a lower target value as positive ('P') and all others as negative ('N').
839	kdd_el_nino-small	Binarized version of the original data set (see version 1). It converts the numeric target feature to a two-class nominal target feature by computing the mean and classifying all instances with a lower target value as positive ('P') and all others as negative ('N').
897	colleges_aaup	Binarized version of the original data set (see version 1). It converts the numeric target feature to a two-class nominal target feature by computing the mean and classifying all instances with a lower target value as positive ('P') and all others as negative ('N').

930	colleges_usnews	Binarized version of the original data set (see version 1). It converts the numeric target feature to a two-class nominal target feature by computing the mean and classifying all instances with a lower target value as positive ('P') and all others as negative ('N').
966	analcatdata_halloffame	Binarized version of the original data set (see version 1). The multi-class target feature is converted to a two-class nominal target feature by re-labeling the majority class as positive ('P') and all others as negative ('N'). Originally converted by Quan Sun.
990	eucalyptus	Binarized version of the original data set (see version 1). The multi-class target feature is converted to a two-class nominal target feature by re-labeling the majority class as positive ('P') and all others as negative ('N'). Originally converted by Quan Sun.
1024	cjs	Binarized version of the original data set (see version 1). The multi-class target feature is converted to a two-class nominal target feature by re-labeling the majority class as positive ('P') and all others as negative ('N'). Originally converted by Quan Sun.

1037	ada_prior	The task of ADA is to discover high revenue people from census data. This is a two-class classification problem. The raw data from the census bureau is known as the Adult database in the UCI machine-learning repository. The 14 original attributes (features) include age, workclass, education, education, marital status, occupation, native country, etc. It contains continuous, binary and categorical features. This dataset is from ”prior knowledge track”, i.e. has access to the original features and their identity.
1053	jm1	This is a PROMISE data set made publicly available in order to encourage repeatable, verifiable, refutable, and/or improvable predictive models of software engineering.
1119	adult-census	Dataset from the MLRR repository: http://axon.cs.byu.edu:5000/. Note: this dataset is identical to the version stored in UCI, but only includes the training data, not the test data. See [adult (2)](http://openml.org/d/1590) for the complete data.
1205	BNG(Australian)	-
41526	test_dataset	-

41430	DiabeticMellitus	This data was collected from combine primary and secondary sources, through questionnaire, verbal interview and some part of the hospital's record department's data, from the selected government's hospitals in the Northwestern states of Nigeria.
1351	BNG(anneal,1000,1)	-
1354	BNG(anneal,5000,1)	-
1358	BNG(anneal,10000,5)	-
1364	BNG(anneal.ORIG,5000,5)	-
1367	BNG(anneal.ORIG,10000,5)	-
1369	BNG(kr-vs-kp,1000,1)	-
1373	BNG(kr-vs-kp,5000,5)	-
1373	BNG(kr-vs-kp,10000,1)	-
1402	BNG(lymph,1000,1)	-
1403	BNG(lymph,1000,5)	-
1453	PieChart3	pie chart 3
1459	artificial-characters	This database has been artificially generated. It describes the structure of the capital letters A, C, D, E, F, G, H, L, P, R, indicated by a number 1-10, in that order (A=1,C=2,...). Each letter's structure is described by a set of segments (lines) which resemble the way an automatic program would segment an image. The dataset consists of 600 such descriptions per letter.

1460	banana	An artificial data set where instances belongs to several clusters with a banana shape. There are two attributes At1 and At2 corresponding to the x and y axis, respectively. The class label (-1 and 1) represents one of the two banana shapes in the dataset.
1468	cnae-9	This is a data set containing 1080 documents of free text business descriptions of Brazilian companies categorized into a subset of 9 categories.
1471	eeg-eye-state	All data is from one continuous EEG measurement with the Emotiv EEG Neuroheadset. The duration of the measurement was 117 seconds. The eye state was detected via a camera during the EEG measurement and added later manually to the file after analyzing the video frames. '1' indicates the eye-closed and '0' the eye-open state. All values are in chronological order with the first measured value at the top of the data.
1475	first-order-theorem-proving	The attributes are a mixture of static and dynamic features derived from theorems to be proved. See the paper for full details.

1479	hill-valley	Each record represents 100 points on a two-dimensional graph. When plotted in order (from 1 through 100) as the Y coordinate, the points will create either a Hill (a “bump” in the terrain) or a Valley (a “dip” in the terrain).
1485	madelon	MADELON is an artificial dataset, which was part of the NIPS 2003 feature selection challenge. This is a two-class classification problem with continuous input variables. The difficulty is that the problem is multivariate and highly non-linear.
1486	nomao	Nomao collects data about places (name, phone, localization...) from many sources. Deduplication consists in detecting what data refer to the same place. Instances in the dataset compare 2 spots.

1489	phoneme	The aim of this dataset is to distinguish between nasal (class 0) and oral sounds (class 1). Five different attributes were chosen to characterize each vowel: they are the amplitudes of the five first harmonics AHi, normalised by the total energy Ene (integrated on all the frequencies): AHi/Ene. The phonemes are transcribed as follows: sh as in she, dcl as in dark, iy as the vowel in she, aa as the vowel in dark, and ao as the first vowel in water.
1497	wall-robot-navigation	The data were collected as the SCITOS G5 robot navigates through the room following the wall in a clockwise direction, for 4 rounds, using 24 ultrasound sensors arranged circularly around its 'waist'.
1502	skin-segmentation	The Skin Segmentation dataset is constructed over B, G, R color space. Skin and Nonskin dataset is generated using skin textures from face images of diversity of age, gender, and race people.
1503	spoken-arabic-digit	This dataset contains time series of mel-frequency cepstrum coefficients (MFCCs) corresponding to spoken Arabic digits. Includes data from 44 males and 44 females native Arabic speakers.

1507	twonorm	This is an implementation of Leo Breiman's twonorm example[1]. It is a 20 dimensional, 2 class classification example. Each class is drawn from a multivariate normal distribution with unit variance. Class 1 has mean (a,a,..a) while Class 2 has mean (-a,-a,..-a). Where a = 2/sqrt(20). Breiman reports the theoretical expected misclassification rate as 2.3%. He used 300 training examples with CART and found an error of 22.1%.
1510	wdbc	Breast Cancer Wisconsin (Diagnostic) Data Set (WDBC). Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The target feature records the prognosis (benign (1) or malignant (2)). [Original data available here](ftp://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/)
1525	wall-robot-navigation	Wall-Following Robot Navigation Data Data Set (version with 2 Attributes)
1526	wall-robot-navigation	Wall-Following Robot Navigation Data Data Set (version with 4 Attributes)

1547	autoUniv-au1-1000	AutoUniv is an advanced data generator for classifications tasks. The aim is to reflect the nuances and heterogeneity of real data. Data can be generated in .csv, ARFF or C4.5 formats.
1549	autoUniv-au6-750	AutoUniv is an advanced data generator for classifications tasks. The aim is to reflect the nuances and heterogeneity of real data. Data can be generated in .csv, ARFF or C4.5 formats.
1552	autoUniv-au7-1100	AutoUniv is an advanced data generator for classifications tasks. The aim is to reflect the nuances and heterogeneity of real data. Data can be generated in .csv, ARFF or C4.5 formats.
1553	autoUniv-au7-700	AutoUniv is an advanced data generator for classifications tasks. The aim is to reflect the nuances and heterogeneity of real data. Data can be generated in .csv, ARFF or C4.5 formats.
1557	abalone	A 3-class version of abalone dataset.
1558	bank-marketing	Reduced version (10 % of the examples) of bank-marketing dataset.
1560	cardiotocography	A 3-class version of Cardiotocography dataset.
1568	nursery	4-class version of the original Nursery dataset

1590	adult	Prediction task is to determine whether a person makes over 50K a year. Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))
4534	PhishingWebsites	In this dataset, we shed light on the important features that have proved to be sound and effective in predicting phishing websites. In addition, we propose some new features.
4538	GesturePhaseSegmentationProcessed	The dataset is composed by features extracted from 7 videos with people gesticulating, aiming at studying Gesture Phase Segmentation.
4541	Diabetes130US	This data has been prepared to analyze factors related to readmission as well as other outcomes pertaining to patients with diabetes.
6332	cylinder-bands	Cylinder bands UCI dataset - Process delays known as cylinder banding in rotogravure printing were substantially mitigated using control rules discovered by decision tree induction.

23380	cjs	The effects of the Growth Regulators Paclobutrazol (PP 333) and Flurprimidol (EL-500) on the Number and Length of Internodes in Terminal Sprouts Formed on Trimmed Silver Maple Trees.
23512	higgs	Higgs Boson detection data. The data has been produced using Monte Carlo simulations. The first 21 features (columns 2-22) are kinematic properties measured by the particle detectors in the accelerator. The last seven features are functions of the first 21 features; these are high-level features derived by physicists to help discriminate between the two classes.
23517	numerai28.6	The data is cleaned, regularized and encrypted global equity data. The first 21 columns (feature1 - feature21) are features, and target is the binary class you're trying to predict.
40536	SpeedDating	This data was gathered from participants in experimental speed dating events from 2002-2004.
575	kdd_coil_4	This data set is from the 1999 Computational Intelligence and Learning (COIL) competition. The data contains measurements of river chemical concentrations and algae densities.

40966	MiceProtein	The data set consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex.
40979	mfeat-pixel	One of a set of 6 datasets describing features of handwritten numerals (0 - 9) extracted from a collection of Dutch utility maps. The maps were scanned in 8 bit grey value at density of 400dpi, scanned, sharpened, and thresholded. Corresponding patterns in different datasets correspond to the same original character. 200 instances per class (for a total of 2,000 instances) have been digitized in binary images.
40981	Australian	Australian Credit Approval. This is the famous Australian Credit Approval dataset, originating from the StatLog project. It concerns credit card applications. All attribute names and values have been changed to meaningless symbols to protect the confidentiality of the data.
40982	steel-plates-fault	A dataset of steel plates' faults, classified into 7 different types. The goal was to train machine learning for automatic pattern recognition.

40996	Fashion-MNIST	Fashion-MNIST is a dataset of Zalando's article images, consisting of a training set of 60,000 examples and a test set of 10,000 examples.
40999	jungle_chess_2pcs_endgame_elephant_elephant	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41001	jungle_chess_2pcs_endgame_complete	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41002	jungle_chess_2pcs_endgame_rat_panther	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41003	jungle_chess_2pcs_endgame_rat_elephant	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41004	jungle_chess_2pcs_endgame_lion_elephant	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41005	jungle_chess_2pcs_endgame_rat_rat	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41007	jungle_chess_2pcs_endgame_lion_lion	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
41027	jungle_chess_2pcs_raw_endgame_complete	This dataset is part of a collection datasets based on the game "Jungle Chess" (a.k.a. Dou Shou Qi).
42638	titanic	-

Appendix C

ALL USED META-ATTRIBUTES

Here are all the meta-attributes used by the system, reported in Section 5.1.4, well as their respective descriptions.

rows_null - Compute the number of instances that have at least one null value.
total_null - Compute the total number of null values in the dataset.
sattr_to_inst - Compute the ratio between the number of attributes.
cat_to_num - Compute the ratio between the number of categoric and numeric features.
freq_class - Compute the relative frequency of each distinct class.
inst_to_attr - Compute the ratio between the number of instances and attributes.
nr_attr - Compute the total number of attributes.
nr_bin - Compute the number of binary attributes.
nr_cat - Compute the number of categorical attributes.
nr_class - Compute the number of distinct classes.
nr_inst - Compute the number of instances (rows) in the dataset.
nr_num - Compute the number of numeric features.
num_to_cat - Compute the number of numerical and categorical features.
attr_conc - Compute concentration coef. of each pair of distinct attributes.
attr_ent - Compute Shannon's entropy for each predictive attribute.
class_conc - Compute concentration coefficient between each attribute and class.
class_ent - Compute target attribute Shannon's entropy.
eq_num_attr - Compute the number of attributes equivalent for a predictive task.
joint_ent - Compute the joint entropy between each attribute and class.
mut_inf - Compute the mutual information between each attribute and target.
ns_ratio - Compute the noisiness of attributes.

can_cor - Compute canonical correlations of data.

cor - Compute the absolute value of the correlation of distinct dataset column pairs.

cov - Compute the absolute value of the covariance of distinct dataset attribute pairs.

eigenvalues - Compute the eigenvalues of covariance matrix from dataset.

g_mean - Compute the geometric mean of each attribute.

gravity - Compute the distance between minority and majority classes center of mass.

h_mean - Compute the harmonic mean of each attribute.

iq_range - Compute the interquartile range (IQR) of each attribute.

kurtosis - Compute the kurtosis of each attribute.

lh_trace - Compute the Lawley-Hotelling trace.

mad - Compute the Median Absolute Deviation (MAD) adjusted by a factor.

max - Compute the maximum value from each attribute.

mean - Compute the mean value of each attribute.

median - Compute the median value from each attribute.

min - Compute the minimum value from each attribute.

nr_cor_attr - Compute the number of distinct highly correlated pair of attributes.

nr_disc - Compute the number of canonical correlation between each attribute and class.

nr_norm - Compute the number of attributes normally distributed based in a given method.

nr_outliers - Compute the number of attributes with at least one outlier value.

p_trace - Compute the Pillai's trace.

range - Compute the range (max - min) of each attribute. roy_root - Compute the Roy's largest root.

sd - Compute the standard deviation of each attribute.

sd_ratio - Compute a statistical test for homogeneity of covariances.

skewness - Compute the skewness for each attribute.

sparsity - Compute (possibly normalized) sparsity metric for each attribute.

var - Compute the variance of each attribute.

w_lambda - Compute the Wilks' Lambda value.

REFERENCES

ALCOBAçA, E. et al. Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, v. 21, n. 111, p. 1–5, 2020. Available at: <http://jmlr.org/papers/v21/19-348.html>.

BANKS, D. et al. *Classification, Clustering, and Data Mining Applications: Proceedings of the Meeting of the International Federation of Classification Societies (IFCS), Illinois Institute of Technology, Chicago, 15–18 July 2004*. [S.l.]: Springer Science & Business Media, 2011.

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD explorations newsletter*, ACM New York, NY, USA, v. 6, n. 1, p. 20–29, 2004.

BATISTA, G. E. d. A. P. A. *Pré-processamento de dados em aprendizado de máquina supervisionado*. Thesis (Doctor in Philosophy) — Universidade de São Paulo, 2003.

BERGSTRA, J.; BENGIO, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, v. 13, n. 2, 2012.

BERTI-EQUILLE, L. Learn2clean: Optimizing the sequence of tasks for web data preparation. In: ACM. *The World Wide Web Conference*. [S.l.], 2019. p. 2580–2586.

BERTI-ÉQUILLE, L.; COMIGNANI, U. Explaining automated data cleaning with cleanex. In: *IJCAI-PRICAI 2020 Workshop on Explainable Artificial Intelligence (XAI)*. [S.l.: s.n.], 2021.

BILALLI, B. et al. Automated data pre-processing via meta-learning. In: SPRINGER. *International Conference on Model and Data Engineering*. [S.l.], 2016. p. 194–208.

BISONG, E. Introduction to scikit-learn. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform*. [S.l.]: Springer, 2019. p. 215–229.

BRAZDIL, P. et al. *Metalearning: Applications to data mining*. [S.l.]: Springer Science & Business Media, 2008.

BRAZDIL, P. B.; SOARES, C.; COSTA, J. P. D. Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, Springer, v. 50, n. 3, p. 251–277, 2003.

BREUNIG, M. M. et al. Lof: identifying density-based local outliers. In: *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*. [S.l.: s.n.], 2000. p. 93–104.

BROCHU, E.; CORA, V. M.; FREITAS, N. de. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR*, abs/1012.2599, 2010. Available at: <http://arxiv.org/abs/1012.2599>.

CAMILO, C. O.; SILVA, J. C. d. Mineração de dados: Conceitos, tarefas, métodos e ferramentas. *Universidade Federal de Goiás (UFC)*, p. 1–29, 2009.

CASTIELLO, C.; CASTELLANO, G.; FANELLI, A. M. Meta-data: Characterization of input features for meta-learning. In: SPRINGER. *International Conference on Modeling Decisions for Artificial Intelligence*. [S.l.], 2005. p. 457–468.

CASTIELLO, C.; FANELLI, A. M. Computational intelligence for meta-learning: A promising avenue of research. In: *Meta-Learning in Computational Intelligence*. [S.l.]: Springer, 2011. p. 157–177.

CHU, X. et al. Data cleaning: Overview and emerging challenges. In: *Proceedings of the 2016 international conference on management of data*. [S.l.: s.n.], 2016. p. 2201–2206.

CUNHA, S. B. D.; CARVAJAL, S. R. *Estatistica Basica-a Arte de Trabalhar com Dados*. [S.l.]: Elsevier Brasil, 2009.

DERCZYNSKI, L. Complementarity, f-score, and nlp evaluation. In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. [S.l.: s.n.], 2016. p. 261–266.

DRORI, I. et al. Alphad3m: Machine learning pipeline synthesis. In: *AutoML Workshop at ICML*. [S.l.: s.n.], 2018.

FACELI, K. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: LTC, 2011.

FEURER, M. et al. Efficient and robust automated machine learning. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2015. p. 2962–2970.

FEURER, M. et al. Openml-python: an extensible python api for openml. *arXiv:1911.02490*, 2019.

FRANÇOIS, D.; WERTZ, V.; VERLEYSEN, M. Choosing the metric: a simple model approach. In: *Meta-Learning in Computational Intelligence*. [S.l.]: Springer, 2011. p. 97–115.

GARCÍA, S.; LUENGO, J.; HERRERA, F. *Data preprocessing in data mining*. [S.l.]: Springer, 2015.

GE, Z. et al. Data mining and analytics in the process industry: The role of machine learning. *IEEE Access*, IEEE, v. 5, p. 20590–20616, 2017.

GUERRA, S.; PRUDÊNCIO, R.; LUDERMIR, T. Meta-aprendizado de algoritmos de treinamento para redes multi-layer perceptron. *Anais do VI Encontro Nacional de Inteligência Artificial*, p. 1022–1031, 2007.

GUYON, I. et al. Design of the 2015 chalearn automl challenge. In: IEEE. *2015 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2015. p. 1–8.

HAWKINS, D. M. The problem of overfitting. *Journal of chemical information and computer sciences*, ACS Publications, v. 44, n. 1, p. 1–12, 2004.

HE, X.; ZHAO, K.; CHU, X. *AutoML: A Survey of the State-of-the-Art*. 2019.

HOOS, H. H. Automated algorithm configuration and parameter tuning. In: *Autonomous search*. [S.l.]: Springer, 2011. p. 37–71.

HUTTER, F.; HOOS, H. H.; LEYTON-BROWN, K. Sequential model-based optimization for general algorithm configuration. In: SPRINGER. *International conference on learning and intelligent optimization*. [S.l.], 2011. p. 507–523.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. *Automated Machine Learning-Methods, Systems, Challenges*. [S.l.]: Springer, 2019.

JANKOWSKI, N.; DUCH, W.; GRKABCZEWSKI, K. *Meta-learning in computational intelligence*. [S.l.]: Springer, 2011.

JOHNSON, B. A.; ANDERSON, W. M.; SPROULE, W. D. *Data preparation for media browsing*. [S.l.]: Google Patents, Jan. 2 2007. US Patent 7,159,174.

KATZ, G.; SHIN, E. C. R.; SONG, D. Explorekit: Automatic feature generation and selection. In: IEEE. *2016 IEEE 16th International Conference on Data Mining (ICDM)*. [S.l.], 2016. p. 979–984.

KEOGH, E.; MUEEN, A. Curse of dimensionality. In: ____. *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer US, 2017. p. 314–315. ISBN 978-1-4899-7687-1. Available at: <https://doi.org/10.1007/978-1-4899-7687-1_192>.

KRISHNAN, S. et al. Activeclean: interactive data cleaning for statistical modeling. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 9, n. 12, p. 948–959, 2016.

LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In: SPRINGER. *Conference on Artificial Intelligence in Medicine in Europe*. [S.l.], 2001. p. 63–66.

LE, T. T.; FU, W.; MOORE, J. H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. *Bioinformatics*, Oxford University Press, v. 36, n. 1, p. 250–256, 2020.

LI, L.; TALWALKAR, A. Random search and reproducibility for neural architecture search. In: PMLR. *Uncertainty in Artificial Intelligence*. [S.l.], 2020. p. 367–377.

LITTLE, R. J.; RUBIN, D. B. *Statistical analysis with missing data*. [S.l.]: John Wiley & Sons, 2019.

LIU, F. T.; TING, K. M.; ZHOU, Z.-H. Isolation forest. In: IEEE. *2008 Eighth IEEE International Conference on Data Mining*. [S.l.], 2008. p. 413–422.

MAIMON, O.; ROKACH, L. Data mining and knowledge discovery handbook. Springer, 2005.

MANTOVANI, R. G. *Use of meta-learning for hypeparameter tuning of classification problems*. Thesis (Doctor in Philosophy) — Universidade de São Paulo, 2018. Available at: <https://doi.org/10.11606/t.55.2018.tde-15102018-092202>.

NORVIG, P.; RUSSELL, S. *Inteligência Artificial: Tradução da 3a Edição*. [S.l.]: Elsevier Brasil, 2014.

OLSON, R. S. et al. Evaluation of a tree-based pipeline optimization tool for automating data science. In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. New York, NY, USA: ACM, 2016. (GECCO '16), p. 485–492. ISBN 978-1-4503-4206-3. Available at: <http://doi.acm.org/10.1145/2908812.2908918>.

OLSON, R. S. et al. Applications of evolutionary computation: 19th european conference, evoapplications 2016, porto, portugal, march 30 – april 1, 2016, proceedings, part i. In: ____. [S.l.]: Springer International Publishing, 2016. chap. Automating Biomedical Data Science Through Tree-Based Pipeline Optimization, p. 123–137. ISBN 978-3-319-31204-0.

POLYZOTIS, N. et al. Data management challenges in production machine learning. In: *Proceedings of the 2017 ACM International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2017. (SIGMOD '17), p. 1723–1726. ISBN 9781450341974. Available at: <https://doi.org/10.1145/3035918.3054782>.

PRUDÊNCIO, R. B.; SOUTO, M. C. D.; LUDERMIR, T. B. Selecting machine learning algorithms using the ranking meta-learning approach. In: *Meta-learning in computational intelligence*. [S.l.]: Springer, 2011. p. 225–243.

PYLE, D. *Data preparation for data mining*. [S.l.]: morgan kaufmann, 1999.

QUANMING, Y. et al. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.

RAHM, E.; DO, H. H. Data cleaning: Problems and current approaches. *IEEE Data Eng. Bull.*, v. 23, n. 4, p. 3–13, 2000.

RAWAT, T.; KHEMCHANDANI, V. Feature engineering (fe) tools and techniques for better classification performance. *International Journal of Innovations in Engineering and Technology (IJIET)*, v. 8, n. 2, 2017.

ROSA, J. L. G. *Fundamentos da inteligência artificial*. [S.l.]: LTC, 2011.

SCHOENFELD, B. et al. Preprocessor selection for machine learning pipelines. *arXiv preprint arXiv:1810.09942*, 2018.

TOMLIN, L.; WELCH, J. S. Finding duplicate rows in a linear programming model. *Operations Research Letters*, Elsevier, v. 5, n. 1, p. 7–11, 1986.

TRAN, B.; XUE, B.; ZHANG, M. Genetic programming for feature construction and selection in classification on high-dimensional data. *Memetic Computing*, Springer, v. 8, n. 1, p. 3–15, 2016.

VANSCHOREN, J. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

VILALTA, R.; GIRAUD-CARRIER, C.; BRAZDIL, P. Meta-learning-concepts and techniques. In: *Data mining and knowledge discovery handbook*. [S.l.]: Springer, 2009. p. 717–731.

VINUTHA, H.; POORNIMA, B.; SAGAR, B. Detection of outliers using interquartile range technique from intrusion dataset. In: *Information and Decision Sciences*. [S.l.]: Springer, 2018. p. 511–518.

WITTEN, I. H. et al. *Data Mining: Practical machine learning tools and techniques*. [S.l.]: Morgan Kaufmann, 2016.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, IEEE, v. 1, n. 1, p. 67–82, 1997.

ZHANG, S.; ZHANG, C.; YANG, Q. Data preparation for data mining. *Applied artificial intelligence*, Taylor & Francis, v. 17, n. 5-6, p. 375–381, 2003.

ZHU, W. et al. Sensitivity, specificity, accuracy, associated confidence interval and roc analysis with practical sas implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, v. 19, p. 67, 2010.

ZÖLLER, M.; HUBER, M. F. Survey on automated machine learning. *CoRR*, abs/1904.12054, 2019. Available at: <http://arxiv.org/abs/1904.12054>.