SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

MASTER
COMPUTER ENGINEERING

A bundle of services to develop better
Machine Learning applications
Miguel Ângelo Machado Guimarães

Miguel Ângelo Machado Guimarães. A bundle of services to develop better
Machine Learning applications

10/2022

SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

MASTER
COMPUTER ENGINEERING

A bundle of services to develop better Machine Learning applications

Miguel Ângelo Machado Guimarães

10/2022

This work does not include the comments and suggestions proposed by the Jury.

SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

M

MASTER
COMPUTER ENGINEERING

A bundle of services to develop better Machine Learning applications

Miguel Ângelo Machado Guimarães

Davide Rua Carneiro

This work does not include the comments and suggestions proposed by the Jury.

Agradecimentos

Em qualquer projeto existe um conjunto de pessoas que o tornam possível, e sem elas, este projeto não estaria concluído.

Agradeço em primeiro lugar ao meu orientador, professor Dr. Davide Carneiro, pela maneira como me acolheu como seu discípulo. Agradeço pela confiança depositada, pelas reuniões que se disponibilizou, correções e sugestões, assim como pela revisão e orientação do projeto.

Agradeço por tudo o que aprendi. E principalmente, por tudo que tive oportunidade de fazer.

Aos meus colegas, pelo suporte, motivação e companhia.

Obrigado à Escola Superior de Tecnologia e Gestão por me ter acolhido no meu percurso profissional, a todo o corpo docente, e pela disponibilidade destes. Pela infraestrutura, na qual se alojou o projeto online 24h por dia. No fundo, por contribuíres para o projeto se tornar uma realidade.

Ao CIICESI, Centro de Inovação e Investigação em Ciências Empresariais e Sistemas de Informação, pela bolsa de investigação que deu origem ao presente trabalho e ao FEDER, Fundo Europeu de Desenvolvimento Regional, pelo financiamento que permitiu esta bolsa ser possível.

Um agradecimento especial também para quem não contribuiu de forma direta para o sucesso deste projeto, mas ainda assim, sem eles também não seria possível, incluindo a família.

"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful. ... all models are approximations. Essentially, all models are wrong, but some are useful. However, the approximate nature of the model must always be borne in mind... To find out what happens to a system when you interfere with it you have to interfere with it (not just passively observe it)."

GEORGE E. P. BOX

RESUMO

Inteligência Artificial (IA) é um tema na moda atualmente. Machine Learning (ML) é a área mais comum de aplicação de IA, e como o nome indica, o objetivo é fazer com que a máquina aprenda.

Essa aprendizagem pode ser a simulação de tarefas repetitivas do Homem, para, por exemplo, testar cenários hipotéticos ou até mesmo substituir a mão de obra humana. Pode inclusivamente, ser uma simulação a nível físico como a nível mental, ou seja, envolver o deslocamento de algum objeto, ou ainda o raciocínio ou o resultado deste de um indivíduo.

Estes sistemas inteligentes podem até superar o intelecto do Homem. No entanto, é necessário haver restrições da sua aplicação em determinados domínios mais sensíveis onde exista um direito à explicação, como refere o Regulamento Geral sobre a Proteção de Dados 2016/679 (RGPD), em que qualquer decisão que tenha por base um sistema inteligente tem de ser justificada.

Como refere o Regulamento Europeu para a Inteligência Artificial, principalmente no ponto 3.5, o uso de IA pode afetar significativamente um elevado número de fatores relacionado com os direitos fundamentais do ser humano. Existe, portanto, a necessidade de assegurar o direito à dignidade humana, respeito pela privacidade, não discriminação e igualdade de género. É necessário garantir também que todos os intervenientes afetados por um sistema de IA tenham as mesmas condições de trabalho e de segurança.

De facto, grande parte das aplicações de ML têm como intuito auxiliar o ser humano, como, por exemplo, ajudar o gestor de alguma empresa a tomar uma decisão e/ou explicá-la. O problema é que os algoritmos conhecidos por oferecerem uma melhor *performance*, tais como redes neuronais que são uma abordagem inspirada no funcionamento do sistema nervoso dos mamíferos, são também aqueles cujo funcionamento ou o porquê de tomarem determinadas previsões é mais difícil de decifrar.

Nesse sentido, motivado pelas novas normas do RGPD e por questões éticas, e com um caso real de aplicação no domínio de deteção de fraude fiscal, um dos objetivos deste trabalho é explicar o porquê das previsões elaboradas pelos algoritmos conhecidos por *black-box*. Não obstante, o trabalho pode ser aplicado a outros algoritmos em que falte a componente explicativa, e outros domínios que necessitem de uma decisão apoiada numa explicação.

A solução proposta é o desenvolvimento de raiz de um sistema inteligente na área XAI (Explainable Artificial Intelligence), que seja incorporado e contribua para um sistema de ML já existente com justificações plausíveis e transparentes sobre as previsões dadas por outros modelos de ML.

Outro desafio destes sistemas inteligentes é a necessidade de um constante retreino de modelos, dado que novos dados chegam ao sistema, para não ficarem obsoletos com o tempo por já não conseguirem eficazmente realizar uma previsão. Contudo, uma maior quantidade de dados não significa necessariamente novos padrões, correndo-se o risco de se desperdiçar recursos a re-treinar um modelo cuja *performance* não é superior à sua anterior versão.

Para abordar este problema, propõe-se o uso de *meta-learning* para prever a *performance* de um modelo de ML com base nas características do *dataset* (caracterizadas por *meta-features*). Resumidamente, será construído um meta-modelo com base nas *meta-features* de vários *datasets*, que terá a capacidade de prever uma métrica de erro de um futuro modelo de ML, e.g. RMSE, MSE, R^2, MAE, incluindo o tempo que demora a treinar o modelo, permitindo assim decidir quanto ao re-treino ou não do modelo.

Este conjunto de serviços para ML permitirá desenvolver melhores modelos, quer do ponto de vista ético, quer do ponto de vista da sua eficiência.

Palavras-chave: *Interactive Machine Learning. Meta-Learning. Error Prediction. Explainable Artificial Intelligence*.

ABSTRACT

Artificial Intelligence (AI) is a fashionable topic these days. Machine Learning is the most common application of AI, and it's widely used in many domains in order to give predictions and help a decision agent to decide. The problem is that the algorithms that are known to offer better performance, such as neural networks, inspired by the human brain, are also more difficult to understand.

Therefore, it's necessary to have restrictions of its application in more sensitive areas in which there is a right to an explanation, as stated in the General Data Protection Regulation 2016/679 (GDPR).

In this sense, motivated by the new rules of the GDPR and by ethical issues, one of the objectives of this work is to explain why of such algorithms known as black-box make certain predictions. The proposed approach can be applied to other algorithms that lack the explanatory component, and in domains that need a decision supported by an explanation.

The proposed solution is the development of an intelligent system from scratch in the field of Explainable Artificial Intelligence (XAI), which is incorporated into and contributes to an existing ML system with plausible and transparent justifications about the predictions given by the other ML models.

Another challenge of these intelligent systems is the need for constant re-training of models, given that new data are always arriving, so that they don't become obsolete over time. However, a greater amount of data not necessarily means new patterns, running the risk of wasting resources on re-training a model whose performance is not better to its previous version.

To address this problem, we propose the use of meta-learning to predict the performance of a ML model based on dataset characteristics (meta-features). In short, a Meta-model is constructed based on the meta-features of many datasets, which will have the ability to predict an error metric of a future ML model, e.g. RMSE, MSE, R^2, MAE, including the time it takes to train the model, thus making it possible to decide on re-training or not the model.

This set of services for ML will allow developing better ML systems, from the point of ethical view and its efficiency.

Keywords: Interactive Machine Learning. Meta-Learning. Error Prediction. Explainable Artificial Intelligence.

Contents

List of Figures

List of Tables

List of Acronyms

AI Artificial Intelligence

AL Active Learning

ANN Artificial Neural Network

API Application Programming Interface

AR Action Research

CD Continuous Delivery

CI Continuous Integration

CLI Command-line Interface

CSV Comma-separated values

CURL Client URL

DM Data Mining

DRF Distributed Random Forest

DT Decision Tree

ESTG Escola Superior de Tecnologia e Gestão do IPP

EU European Union

GDPR General Data Protection Regulation

HCC Human Centered Computing

HTTP Hypertext Transfer Protocol

IML Interactive Machine Learning

IQR Interquartile range

JSON JavaScript Object Notation

KMDSS Knowledge Management and Decision Support Systems

LIME Local Interpretable Model-Agnostic Explanations

MAE Mean Absolute Error

ML Machine Learning

MRD Mean Residual Deviance

MSE Mean Squared Error

NBDT Neural-Backed Decision Trees

OL Online Learning

OS Operating System

REST Representational State Transfer

RMSE Root Mean Squared Error

RMSLE Root Mean Squared Logarithmic Error

R^2 R Squared

SCM Software Configuration Management

SHAP SHapley Additive exPlanations

SMEs Small and medium-sized enterprises

SVM Support Vector Machine

UI User Interface

WSGI Web Server Gateway Interface

XAI eXplainable Artificial Intelligence

XDT Explainable Decision Tree

XML eXtensible Markup Language

Chapter 1

Introduction

The recent growth of Machine Learning (ML) on business tasks and value-added operations is undeniable. Machine Learning algorithms have outpaced human decision performance in many domains, and this has convinced decision makers to rely increasingly on them [1]. Thus, Machine Learning is nowadays used in virtually all spheres of our existence, controlling our routines in pervasive and transparent ways, and often used to take decisions that have a measurable impact in our lives[2, 3].

The increase in the use of ML has been accompanied by the increase in the volume of data used. Generally, the more complex the problem/domain is and the larger the dataset, the more complex the models learned are. This poses two main challenges: 1) models are harder to understand and interpret by human decision-makers; and 2) the training of models becomes more expensive in terms of time and resources.

The first challenge poses an interpretability problem: a human user obtains a decision/prediction from a model, but often lacks the necessary information to properly judge and evaluate the outcome. Namely, "how good is it?", "how good are neighboring decisions?", "what is the rationale behind it?", "how does the model behave and how can it's behavior be understood and predicted?". This is especially true when so-called black box models are used, such as Deep Learning, or ensembles such as Random Forests. Coincidentally, these are among the most popular models nowadays.

The second challenge is related to efficient resource management in Machine Learning [4]. Indeed, as the size of datasets increases, so do the necessary resources to train the models and the corresponding costs [5], both in terms of infrastructure/services, time and energy. These costs are also higher in data streaming scenarios or in domains with concept drift [6]. In these cases, frequent updates of the model are necessary, to ensure that it remains up to date with recent data. On the one hand, this encompasses significant costs as the training of models are computationally intensive tasks. On the other, it may ultimately be impractical, if the training of the model takes so much time that it is already outdated when it is finished.

In this work, we propose an integrated approach to deal with both these challenges. While the approach is generic, it is instantiated for a specific case study on tax fraud detection. The key takeaways from this work are thus:

- We propose a way to build instance-level explanations for any Machine Learning model, using a proxy explainable model that does not require access to the original data but only to the model being explained;

- Explanations are symbolic in the sense that they are based on the concepts of the domain, thus being easier to understand by Human practitioners;
- We propose an approach for predicting model performance, that is especially useful for deciding when to update a model in data streaming scenarios.

1.1 Explainability in Machine Learning

Machine Learning problems and algorithms have been growing in complexity in the past years, mostly due to the increase in the volume and complexity of data. As a consequence, models and their behavior are increasingly harder to fathom by Humans.

This creates challenges for researchers, regulatory bodies of industry-specific applications and other decision makers where there is a need to guarantee principles such as non-discrimination, adhering to regulations, complying with scientific domain, or following certain legislation.

Thus, the ability of human agents to *understand* the decision process of models or the rationale behind their predictions is nowadays a common requirement in the field of Machine Learning. It has been addressed by a relatively recent field known as eXplainable Artificial Intelligence (XAI) [7]. While XAI is a very broad field that can encompass multiple knowledge domains, in this dissertation we focus on the aspects more related to Machine Learning models. In this scope, it is important to first make the distinction between two important terms: explainability and interpretability.

While these concepts are often used interchangeably, they represent different notions according to the direction in which information flows [8, 9]. Explainability relates to the ability of the model to detail why and how a given prediction is being made, or how the model behaves internally. Interpretability, on the other hand, is the ability of the human user to understand the explanations provided. Asides from these, the research community has put forward other desirable properties such as transparency - the ability to visualize the inner workings of the model, the ability to use domain-knowledge in the explanation, or its scientific consistency [9].

Depending on the requirements, there can also be varying degrees of explanation. For instance, it is possible to explain a decision process without actually understanding the model which generated such decision, or the intricate relationships between cause and effect in the decision process [1]. An example is the use of saliency maps to explain the classification of an image: while such an explanation may be used to understand a prediction, it is not helpful to understand how a Deep Learning model works.

Building explanations is naturally easier in some models, namely those based on statistical or rule-based algorithms (e.g. Decision Trees). It is much harder and less intuitive in the so-called "black-box" models (e.g. Deep Learning), that are characterized by high complexity and abstraction levels.

Nonetheless, different approaches are being undertaken in both explainable and black-box models. These approaches are sometimes specific to a given algorithm, or generic and applicable to a broad range of them.

One of the most interesting examples is the use of counterfactuals or evidence based on the interpretability of the model. These require a deep understanding of the Machine Learning model being used and how changes in the input may alter the outcome [10]. These decisions are characterized by the complete categorization of a specific decision and how the decision would be altered given some changes in the input.

This is a generic idea which may have different implementations depending on the algorithm being studied. In the literature this approach can be found in linear classification algorithms [11], where a linear Machine Learning algorithm is exploited to find how changes in coefficients or inputs change the final decision, as well as in black box models such as multilayer perceptrons [12].

Indeed, explanations are often more valuable when it comes to black box models, as there is here, clearly, a trade-off between interpretability and accuracy [13]: models that are generally more accurate, such as Deep Learning, are usually also harder to explain. As a consequence, when interpretability and/or explainability are critical project requirements, other less accurate models are often used.

New explainable methods can, however, allow for these complex models to be explained and thus allow their use even in domains where explanations are critical. A common strategy is to use an additional or external algorithm or methodology to provide the necessary explanation, while still using the underlying original model.

Examples of generic and domain-agnostic approaches that follow this strategy are LIME [14] and SHAP [15]. LIME approximates inputs to predictions using a local linear explanation model while SHAP uses metrics represented by SHAP values to denote the expectation of the prediction change in the model based on a determined feature. LIME and SHAP are not mutually exclusive and can be used together [15]. These approaches have been used to explain algorithms' decisions in fields such as intrusion detection [16] or anomaly detection [17].

Other approaches are dependent on the type of black box algorithms being explained. For instance, in the case of imagery and neural networks, the DeepLIFT [18] approach tries to explain which are the most relevant pixels in an image through the analysis of the weight activation in a neural network alike algorithm. Layer-Wise Relevance Propagation [19] is yet another neural networks specific method that interprets the predictions similarly to DeepLIFT. More detailed examples will be discussed in section 2.2.

There are, however, additional concerns to be taken into consideration especially when it comes to streaming scenarios, in which ML algorithms must not be stationary in time but be updated. This creates additional concerns with model deployment and adds the need to monitor models and audit models throughout their lifecycle. Even more specifically, there is a need to guarantee that local and global explainability and model trustworthiness is maintained between deployments [20].

Explainable AI will always have to be a two-way road, in which models need to be able to explain decisions in a way that human users can understand them. While some types of explanation can be generic and apply to any problem, domain-dependent ones are probably more valuable as they will add insights that are specific to that problem, and thus more relevant to support decision-making. This is related with the notion of completeness, which is the extent to which an explanation allows a complete understanding of all the domains for each attribute in the decision-making process [21].

The general perception is that all models can be explained to some extent. However, some are easier to explain than others. In any case, explanations should consider the mental model of the user and the domain of the application, something that is not always considered by existing methods. This is the approach to explainability followed in this work, which will be further detailed in chapter 4.

1.2 Measure the Quality of a ML Model

Every Machine Learning model needs data. Thus, this is a fundamental part of the system. More often than not, the quantity and quality of the data determines the success of an algorithm. In fact, the Machine

Learning community knows that usually more data beats better algorithms. E.g., in a Netflix challenge, one team of students, using a very simple algorithm, almost beat the accuracy of Netflix's proprietary algorithm by adding more information about the movie genres from IMDB [22]. On the other hand, more data does not always result in a better model, as more data does not mean more quality data [23]. Especially when it comes with increased costs [24].

Thus, there is a need to know when more data are good to a ML model and when not, as it has a big effect on the model performance. Also, it can be as good to have some kind of function that for all the data that we have for a particular problem, what are and how many do we have to incorporate in the training of a ML model in order to have the best possible model for that problem and provide efficient predictions. In a time that ML is applied everywhere, this is a crucial point for the success of an application, specially in sensitive domains, like prognosticate if some patient has or not some disease.

Until this point, we only discussed one problem: the importance and the effect that data has on ML algorithms. We were assuming that all data was available at once, then only one ML model was needed. In the literature, having all data available at once is considered full batch learning [25]. Even in this case, it can be particular useful to know which of the data available can be the best possible model.

However, the problem gets more difficult in Interactive Machine Learning (IML) and online learning scenarios, like when the data comes from streaming. In this case, it's even more important to know if new data will get a better model.

In the last years, IML has met some advances in different domain areas [26]. In contrast to traditional one-shot batch systems, in which the model is trained only once, in IML and data streaming scenarios, one common task is to update or re-train models when data change.

A small period between model updates may lead to a system that adapts faster to changes in the data, like in concept drift scenarios [27], but also one more sensitive to noise. It will also be a more costly system in the sense that it will require more computational resources and time for the constant re-training of the models.

On the other hand, a system with a large period between updates, although requiring less computational resources, may perform worse over time.

Thus, there is a need to understand how well a model would perform if more data were added to it, to achieve the best possible model with less data in the shortest amount of time, and consequently with less computational power. For that, we developed a novel approach to predict the error of a future model using meta-learning.

1.3 Case Study

Sections 1.1 and 1.2 detailed the relevance of the two main problems being addressed in this work: 1) to predict the performance of a Machine Learning model before its training; and 2) to explain the predictions of a model in a way that can be understood by a human. However, nowadays, explanations are not only desirable from a perspective of interpretability but are starting to become a legal requirement. In the context of the General Data Protection Regulation (GDPR), the European Union (EU) recently regulated on algorithmic decision-making and, specifically, addressed the issue of a "right to explanation"[28]. There are particularly sensitive domains in which algorithmic decisions significantly affect one's life, such as credit scoring, sentencing, or fraud detection.

This section describes the Case Study that motivates this work, which is inserted in these sensitive areas of application of Machine Learning. Specifically, this work was developed in the context of the Neurat funded project (31/SI/2017 - 39900). One of the goals of the Neurat project is to develop a Machine Learning environment for tax fraud detection. There are however some particular characteristics, namely:

- There are two types of variable in this problem: static variables (which are obtained through the feature extraction process from the raw data), and dynamic variables (which are proposed by human users);
- Labeled data is scarce since most of the dynamic variables cannot be extracted from the raw data and must be provided manually by human users (some of them reflect subjective or abstract concepts);
- Users' actions change the dataset over time as they validate the models' predictions (contributing to the labeled dataset) or provide their own contribution, for example;
- New raw data is added regularly;
- Models must be updated frequently to adapt to new data. However, new data is not necessarily different from existing data.

The system can thus be described as a cooperative environment in which ML tools and human experts (auditors) work together to increase the efficiency of tax audits. However, the use of ML, and in particular of supervised methods, requires vast amounts of labeled data. The problem is that data can only be labeled by auditors and, it comes at a high cost: auditors must undergo extensive training and their time is very limited. As a consequence, they are able to review but a small portion of the transactions of a company, usually by sampling, and thus provide a small amount of labeled data.

Thus, an Active Learning (AL) approach is being followed to implement it [29]. Generally, AL approaches aim to make Machine Learning less expensive by reducing the need for labeled data. To achieve this, a so-called *Oracle*, which may be a human expert or some automated artifact, is included in an cycle in which a ML model is continuously improved by training on a growing pool of labeled data. In this case, the Oracle are the auditors.

However, we introduce several major changes to the "traditional" AL scheme (Figure 1.1). First, we consider a pool of models rather than a single model. In practice, one model is maintained for each dynamic variable in the problem. This is necessary since dynamic variables, as opposed to static ones, cannot be extracted from the raw data. Thus, one model is maintained to predict each variable. When models are updated, so are the predictions for these variables. Predictions are then validated or changed by the auditors, in what constitutes one way of incorporating human knowledge into the system.

Secondly, we add an additional input to the Oracle. When assessing one instance of data i, the auditor has also access to a prediction p for each dynamic variable, and a corresponding explanation e. The former is provided by each model f associated to each dynamic variable, while the latter is provided by the Explanation Interface as a result of $f(i)$. Now, when the auditor receives the instance to label (that is, when the auditor performs an audit action), he also receives the labels proposed by the system for the data not yet validated, as well as an intelligible explanation for it, tailored for this specific domain.

Finally, we also introduce a fork in the typical AL process in the moment of training new models. While, typically, models are updated at regular intervals or when a certain amount of new labeled data exist,

we include a module for predicting the performance of a future model if it is trained with a given dataset. The goal is to avoid training new models if this is not expected to result in a significant performance improvement.

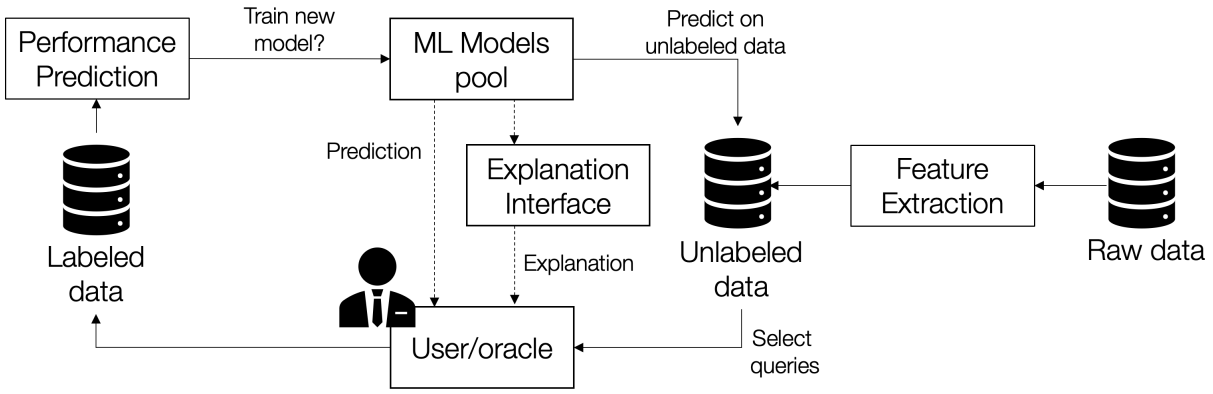

Figure 1.1: High-level view of the main elements of the proposed system.

1.4 Objectives

Machine Learning is one of the most trending topics nowadays. The reason is that it is so useful to the human being that it becomes almost impossible to ignore. Mainly for companies looking for a competitive advantage in the market. Thus, being more and more present in our everyday life, even if we do not notice it. What goes even more unnoticed is the fact that every Machine Learning model needs computational power. And of course, it also needs data. But how much data are necessary to build the best Machine Learning model possible, and how many times do we need to re-train a model so that it does not become obsolete as data change? This kind of questions are the ones that can reduce unnecessary costs to a company.

On the other hand, how useful can a prediction be if no explanation is given? It's possible to apply some of the best ML models to any domain and still give an explanation? And what kinds of explanation a specific application need? These are some of the questions that concern the implementation of an ML system in a company.

The following objectives try to give an answer to the previous questions and are the purpose that brings this work to life, they are:

- Implementation of an approach for optimizing the speed of learning in online learning systems;
- Avoid unnecessary training of ML models in streaming data scenarios, using meta-learning;
- Explain the predictions of ML models, mainly those known as black-box, because it is not possible to extract an explanation of its predictions based on the model.

1.5 Research Methodology

The goals of this scientific research were to develop and test a bundle of services to help build better ML systems, while reducing costs and taking legal requirements seriously.

However, the authors did not know exactly how to achieve these goals or what to do.

Thus, a methodology was followed that allows for research but also to act in an iterative way, as is the case of the Action Research (AR) [30]. It is the perfect methodology for tackling challenging problems, as it does not plan the entire project, but reflects, acts and observes in light of the latest experiences (figure 1.2).

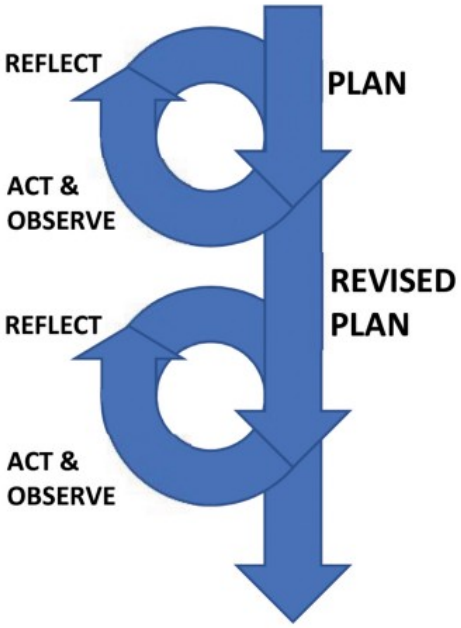

Figure 1.2: Kemmis and McTaggart's action research spiral [31]

1.6 Document Organization

The dissertation is divided into 7 chapters. The first chapter presents the problem we are trying to solve, and the proposed solution. The second chapter provides the background in the area in which this work is inserted: Machine Learning (ML), starting by a summary of the five tribes of ML [32], going through different types of learning scenarios and summarizes what already exists to solve the problems addressed in this work. Afterwards, chapter three shows the architecture developed to the case study that inspires this work: a tax fraud detection, nonetheless can be applied in any other domain. Chapter four describes in detail the Explainable Decision Tree (XDT), how it works and how it can be used with other systems through its API. Chapter five describes in detail how Model Performance Prediction works, as well as how it can be incorporated into other ML systems. In chapter six, the results obtained are presented. Finally, in chapter seven, the limitations of this work and future directions are discussed.

Chapter 2

State of the art

2.1 Machine Learning

2.1.1 The Five Tribes of Machine Learning

In [32] the author introduces five different approaches and ways of thinking about ML in which we already have decades of research available, they are: symbolists, connectionists, evolutionaries, bayesians and analogizers.

For symbolists, all learning can be obtained by manipulating symbols, i.e. like mathematicians solves equations by replacing expressions with other expressions [32]. In that line of thought, it's required presenting some initial knowledge to the system, and it's possible to transform this knowledge into learning in order to solve new problems.

Connectionists correctly assume that learning is what a human brain does. So, what they try to do is reverse engineer it in order to have an artificial brain that can, theoretically, learn as a human brain does. This is done by adjusting the strengths between artificial neurons in order to bring the output closer to what it should be [32].

Evolutionary approach is that of natural selection. Indeed, they have an irrefutable argument, that natural selection made us. So, with genetic algorithms they try to resolve problems the same way nature does, by crossing and mutating the most adaptive chromosomes.

Bayesians think of all learned knowledge is uncertain and the solution for learning is probabilistic inference. With this in mind, it's possible to incorporate new evidence into our beliefs as it increases the probability of learning [32].

Finally, for analogizers, the recognizing similarities between data plays an important rule in the learning process, and therefore they judge new events with old events rational.

2.1.2 Types of Learning

Interactive Learning

In what regards the relationship between Human specialists and Artificial Intelligent systems, there are mainly two trends. The most common one is that the interaction between the Human and the System occurs at the moment of generating the data. The Human may be involved in the process of data labeling

and thus contributes to the knowledge that the AI system will model. This is thus an *ante-hoc* interaction, that is, before the training of the model.

However, the action of the user is sometimes necessary or desirable during the use of the system. When this happens in an iterative fashion and leads to an improvement of the models over time through the interaction with the user, it is deemed Interactive Learning [33]. Interactive Learning is interesting in several different situations. It may happen in scenarios in which not all the data is available at the beginning of the process. In these cases, new data may arrive that needs to be labeled by the Human expert while the system is already in use, and the models need to be updated. It may also happen when the data changes significantly during the use of the system, a phenomenon known as concept drift [34]. Or, it may happen that the problem we are trying to solve is computationally hard, and the involvement of Human experts may help speed up or simplify the learning process [35]. Interactive Learning may also be a solution for scenarios of very large datasets which cannot be dealt with in a single run of an algorithm.

In terms of interaction, and as opposed to traditional Machine Learning systems, this is thus a *post-hoc* interaction between the Human expert and the system. That is, the interaction occurs after the training, through the evaluation of the predictions of the model by a Human expert. The general idea of interactive learning can thus be depicted as in Figure 2.1: the Human expert analyses the prediction of a model for a given input and provides her/his feedback, which is then incorporated into the model somehow to iteratively improve it over time.

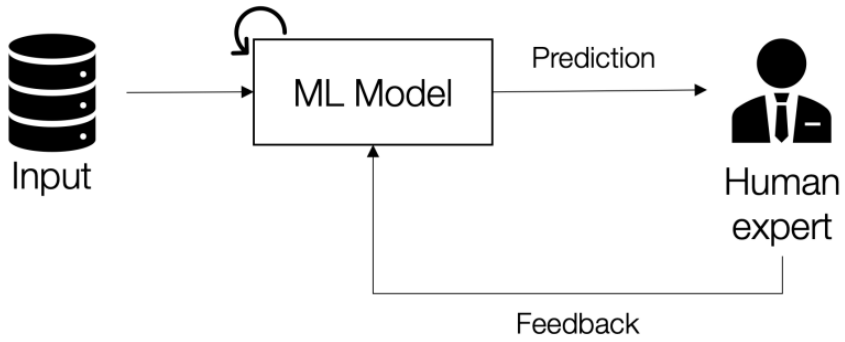

Figure 2.1: General diagram of an interactive learning system.

Different approaches exist to integrate Human knowledge and skills into Machine Learning models. For instance, in [35] the authors modify the Ant Colony Optimization algorithm to allow humans to ”guide” the learning of the algorithm by playing alongside. The authors used a snake-like game to allow Humans to play and thus point the algorithm towards a more rapid learning gradient. Interactive approaches have also demonstrated that even users who are not domain experts can often construct good classifiers, without any help from a learning algorithm, using a simple two-dimensional visual interface [36]. The authors show that if a few attributes can support good predictions, users generate accurate classifiers, whereas domains with many high-order attribute interactions favor standard Machine Learning techniques. Healthcare in particular, in which datasets are often small, can benefit especially from an Human-in-the-loop approach, by allowing domain experts to provide knowledge and expertise to datasets which are often so small that they cannot be used by automated Machine Learning algorithms [37]. Specific applications exist, including in the annotation of medical imaging [38] or in knowledge discovery in bioinformatics [39]. Sections 1.3 and 3 describes one such Interactive Learning system, developed for the domain of financial fraud detection.

Active Learning

As ML is a relatively recent field, there are some incongruities with some concepts in the practitioners. Active Learning (AL) is often considered to be IML and vice versa. However, AL can also be seen as a technique for IML by some authors [40]. IML aims to enable non ML experts to train a ML model through "rapid, focused, and incremental model updates" [41]. AL can contribute to that, as the core idea of AL is to help ML systems learn without the need of Big Data. This is due to the fact that a Human agent is asked to give knowledge to the system. This is done by iteratively selecting key instances to be labeled by an oracle, e.g., a human/data annotator, which job is the categorization and labeling of data for AI applications.

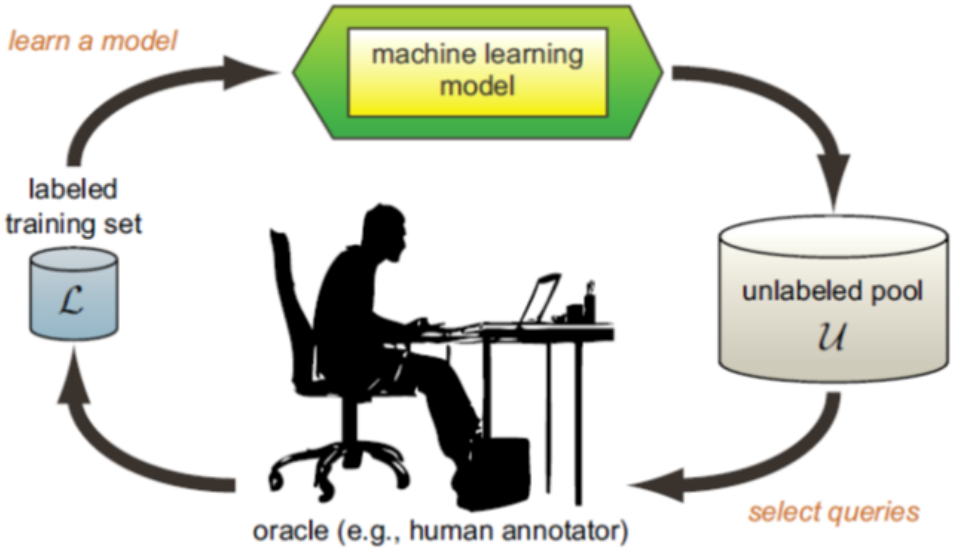

Figure 2.2: Active Learning [42]

Figure 2.2 shows that after the training of a ML model, some outputs with high uncertainty return to the unlabeled pool. On the other hand, the human annotates some instances of the unlabeled pool and input those labelled instances for the training set.

The point of this approach is that the job of a data annotator is reduced to only label the instances that are highly uncertain by the ML model, thus, avoid classifying those samples with high certainty predicted by the ML model. Consequently, save effort and cost of a data annotator at the same time that builds a better ML system with less available data.

Online Learning: Learning in Streaming Scenarios

Current ML algorithms face several issues that stem from the new nature of data (e.g., volume, complexity, velocity, variety) [43]. One of the most significant is to learn from high-speed streaming data. In these scenarios, learning becomes continuous, and models are often initially trained with small amounts of data, and are later updated or re-trained when new data are available.

There are, however, additional issues. Namely, there is the need to decide which data will be used to train future models or when will future models be trained or updated. There is here clearly a trade-off between the use of computational resources and how outdated a model is.

A small period between model updates or re-trains may lead to a system that adapts faster to changes in the data, but also to one more sensitive to noise. It will also be a more costly system in the sense that it will require more computational resources and time for the constant re-training of the models. Ultimately, depending on the complexity of the model and the velocity at which time changes, it may be impossible to update models in due time.

On the other hand, a system with a large period between updates will require less computational resources, but may perform worse over time as models become obsolete and no longer reflect the patterns in the data.

When learning from streaming data, a key assumption from traditional batch ML systems is violated: that the training and test data are drawn from the same feature space and have the same distribution.

One of the recently proposed techniques for dealing with this is Transfer Learning: the goal is to extract knowledge from a task and use it to improve learning on a different task. Different elements can be transferred, including instances, feature representation or parameters. Various approaches exist, including inductive, transductive and unsupervised transfer learning [44].

These challenges are more significant when there is concept drift [34]. That is, when data, concepts, variables, or patterns change over time. In these scenarios, algorithms must learn and forget concepts incrementally, and the act of forgetting becomes as relevant as the act of learning new concepts.

To deal with these scenarios, the notion of Evolving Ensemble has been proposed [45]. This denotes an Ensemble whose weights are fine-tuned using some optimization mechanism, generally of biological inspiration (e.g. genetic algorithm) [46]. However, when concept drift is too significant, models eventually have to be replaced. Adaptive Random Forests address this issue by training a new model in the background when concept drift is detected, which later replaces the original model [47].

We propose an approach that differs from the previous ones in the sense that our main goal is to predict the performance of a given model when trained on a given set of data. Thus, rather than updating models at regular intervals or when a certain amount of new data exist, we provide an additional decision layer, that is the predicted performance of the future model. The main goal is to avoid re-training a model if it is not expected to perform better than the previous one, thus more efficiently managing resources.

The following subsection describes the most important concept to predict the performance of a future ML model: meta-learning.

2.1.3 Meta-Learning

Meta-learning [48] is a rich field, that can be applied to distinct domain areas [49]. For example, in [50] Meta-Learning was used to give advice about which classification method is appropriate for a particular dataset.

Many researchers hold different perspectives of what the word “meta-learning” exactly means, but one of them can be explained as the ability to learn how to learn [48]. In short, meta-learning is an umbrella term for a system that uses meta-data with the aim to learn more about the data itself.

Other types of learning can be viewed as a type of meta-learning too, such as Ensemble Learning, that combines several base models in order to produce one optimal predictive model. E.g., the stacking generalization method [51] works by combining several base-level learning algorithms and using a meta-level learner to learn a linear function of the models produced by the base-level algorithms.

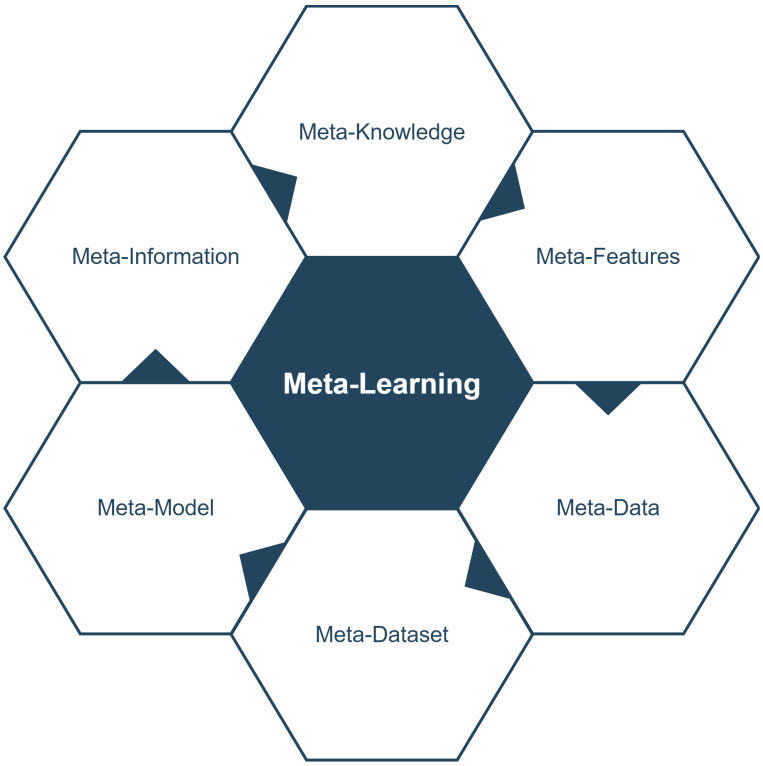

Figure 2.3: Meta-learning concepts.

Figure 2.3 presents the concepts that are most related to meta-learning. The list is eclectic and by no means exhaustive. Therefore, in this section, we describe these concepts with an emphasis on the degree of relevance to the present work.

Meta-Data

Meta-data is data about data which characterize the original data source. It is widely used for building, maintaining, and managing data warehouses [52]. In the Data Warehouse Architecture, meta-data plays an important role as it specifies the source, usage, values, and features of data warehouse data. It also defines how data can be changed and processed [53].

To understand the importance of this subject and the extents of analyzing together many data from different sources and domains, one can look at an example by Matt Blaze, an encryption expert who directs the Distributed Systems Lab at the University of Pennsylvania. Concerning the power and peril of the meta-data that NSA collects, Blaze states that:

"On a massive scale, when you look at everyone's metadata, it becomes even more powerful, more revealing, perhaps, than content. Why? Because unlike with content, there is no real limitation on how much of it can be effectively processed... And the more metadata there is, the more revealing it is." [54].

Meta-Features

One important concept in meta-learning are meta-features, which are like hyperparameters: parameters whose values are used to control the learning process. Meta-features are the attributes that bring the meta-dataset to life.

The meta-features characterize a dataset and the relation between the available knowledge and the needed knowledge to build the best ML model in the shortest time possible. Meta-features can be clustered in categories, namely: (i) features that describe the attributes, (ii) relationships between attributes, and (iii) relationships between attributes and the target.

Those features can be very simple measures, such as the number of classes or targets, to more complex ones, such as statistical information (e.g., mean kurtosis of attributes, mean skewness of attributes, etc.) or information-theoretic characteristics (e.g., noise signal ratio, class entropy, etc.).

Meta-Dataset

Meta-features are used to build the so-called meta-dataset. The meta-dataset is like any other dataset used to create a ML model, that is composed by the meta-features, that represent the independent variables, and meta-labels, which represent the dependent variables in the different ML problems.

#	meta-feature 1	meta-feature 2	meta-feature 3	meta-label
meta-example 1	0.5	0.7	0.4	1
meta-example 2	0.7	0.6	0.5	0
meta-example 3	0.6	0.5	0.8	1

Table 2.1: Example of a meta-dataset with three rows and columns

The typical structure of a meta-dataset seen in table 2.1 has instances (from now on called meta-examples) with two types of data. The meta-features characterize a dataset (ML problem), or a part of one, in terms of the quality and distribution of its data. The meta-labels, on the other hand, contain the performance metrics of a model trained for that particular ML problem. This makes the correspondence between data characteristics and resulting model performance, which will then be learned by the meta-model.

2.2 Explainability

Explanation has been a central feature of AI systems for legal reasoning since their inception [55]. Paradigms underlying this problem fall within the so-called XAI field, which is widely acknowledged as a crucial feature for the practical deployment of AI models [56]. AI researchers and practitioners have focused their attention on XAI to help them better trust and understand models at scale [57].

In a legal dispute, there will be two parties and one will win and one will lose. Losers have a right to an explanation of why their case was unsuccessfull [55]. Given such an explanation, the losers may be satisfied and accept the decision, or may consider if there are grounds to appeal [55]. Explanation is essential for any legal application that is to be used in a practical setting [55].

Explainability is a prerequisite for building trust and adoption of AI systems in high stakes domains requiring reliability and safety such as healthcare and automated transportation, and critical industrial applications with significant economic implications such as predictive maintenance, exploration of natural resources, and climate change modeling [57]. The challenges for the research community include [57]: (i) defining model explainability; (ii) formulating explainability tasks; and (iii) designing measures for evaluating the performance of models in explainability tasks.

The right to explanation is viewed as a promising mechanism in the broader pursuit by government and industry for accountability and transparency in algorithms, artificial intelligence, robotics, and other automated systems [58]. GDPR does not, in its current form, implement a right to explanation, but rather what is termed a limited "right to be informed" [58].

The requirement for an explanation can be of two types in what regards their temporal relation with the training of the model. They can be *ante-hoc*, when the explanation is a requirement at the beginning of the project, and an explainable model is used. Or, they can be *post-hoc*, when the requirement is established after the training of the model and a black-box model was used.

Explanations can also be model-agnostic, when they are general and ignore the internal structure of the model, or model-specific, when they depend on the structure of the model. Finally, explanations can also be categorized according to heir scope: they are deemed local when they explain a single instance of data or prediction, or global when they explain the behavior of the model [59].

For every type of explanation, there are nowadays techniques or approaches that can be used to implement them. When explainability is an initial requirement, inherently explainable models can be used such as Decision Trees [60], Linear Models [61] or Case-based Reasoning [62]. When explanations must be implemented after the training of the model, several approaches can be used.

Indeed, one common source of ethical problems is the use of complex models, that are difficult to understand, otherwise known as black-box models. Unfortunately, more accurate models such as deep learning or ensembles are generally harder to explain [63]. In this regard, deep learning systems are probably the best examples of black-boxes in the sense that they are the most complex models. Still, as with any other model, deep learning models can be explained in the sense that it is possible, for instance, to detail the whole process of training of the model, or to explain a particular prediction in terms of all the calculations involved. The problem is that the number of calculations is so large and so devoid of meaning concerning the overall picture, that this kind of explanation is really not helpful for a human. A distinction should thus also be made between explainability and interpretability: the former being the ability of the model to explain itself, and the latter the ability of the human to understand the explanation.

For model agnostic explanations [64], techniques such as Partial Dependency Plots [65], Feature Relevance [66] or Feature Interaction [67] can be used. Another well-known *post-hoc* model agnostic methods include LIME (Local Surrogate) and SHAP (SHapley Additive exPlanations) [68]. Both methods also still use a black-box model for predictions, but they perform minor changes in the input to evaluate changes in prediction, and thus understand how one relates to the other. For instance, if a change in a variable does not result in a change in the prediction, that variable is probably not a very relevant one. Changes are, however, small enough for the new data point still be in the vicinity of the original one.

Specific methods also exist for connectionist algorithms, such as Artificial Neural Network (ANN) or deep learning, mostly focused on image classification tasks. These include different versions of Saliency Maps, which are similar to SHAP and LIME, but for pixel data, such as Vanilla Gradient or SmoothGrad [69].

"Gathering information, and above all developing trust, have become the key source of sustainable competitive advantage."

HEIL, BENNIS AND STEPHENS (2000)

Specifically, the main objective off all these techniques is to develop trust in a ML system. Trust is

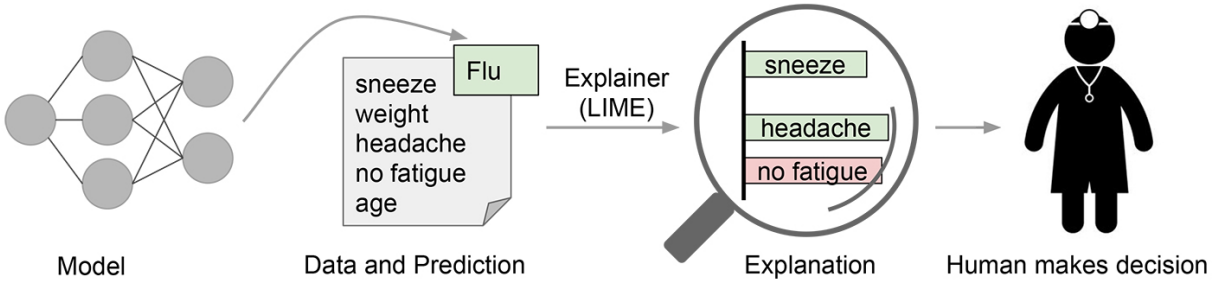

Figure 2.4: Explaining individual predictions. A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient's history that led to the prediction. Sneeze and headache are portrayed as contributing to the "flu" prediction, while "no fatigue" is evidence against it. With these, a doctor can make an informed decision about whether to trust the model's prediction. [71]

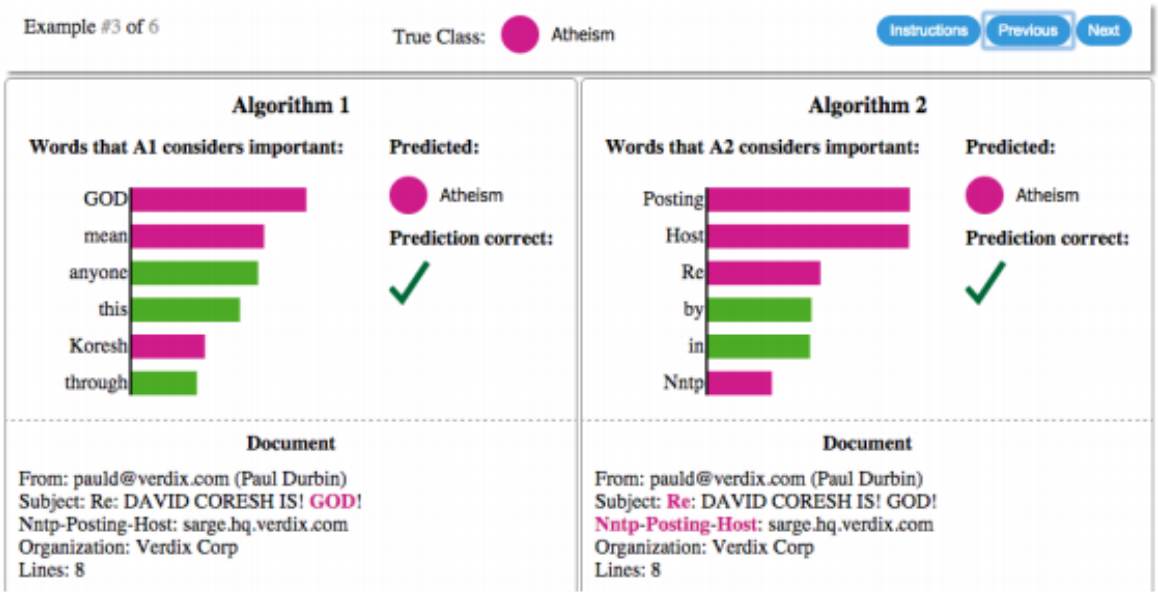

Figure 2.5: Explaining individual predictions of competing classifiers trying to determine if a document is about "Christianity" or "Atheism". The bar chart represents the importance given to the most relevant words, also highlighted in the text. Color indicates which class the word contributes to (green for "Christianity", magenta for "Atheism"). [71]

the main ingredient to effectively transfer knowledge in organizations [70]. Interpersonal trust makes knowledge exchanges less costly and increases the likelihood that newly acquired knowledge is sufficiently absorbed so as to be useful to the recipient [70]. The same happens in a decision support agent that uses a ML system to help take decisions or to give advise to someone, if the agent trust in the model, then he can take or help someone take better decisions in a more efficient way.

In [71] the autors present Local Interpretable Model-Agnostic Explanations (LIME) to explain the behavior of the system without accessing to its internal parameters, thus transform a black-box model into a white-box one. LIME gives local explanations as he acts on the neighborhood of its input values. Therefore, is possible to build trust or not in the model and in his predictions, given the reason why the model behaves (figures 2.4-2.7).

In figure 2.4 is given an explanation for a classification model that predicts that a patient has flu. In figure 2.5 are highlighted in magenta the words that contribute to the prediction of two Support Vector

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 2.6: Explaining an image classification prediction made by Google's Inception neural network. The top 3 classes predicted are "Electric Guitar" (p = 0.32), "Acoustic guitar" (p = 0.24) and "Labrador" (p = 0.21). [71]

(a) Husky classified as wolf (b) Explanation

Figure 2.7: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task. [71]

Machine (SVM) models to text classification. In figures 2.6 and 2.7 is shown the explanation for the classification of images using deep learning.

Using LIME, the authors show that even non-experts are able to identify irregularities when explanations are present [71]. Further, LIME can complement existing systems, and allow users to assess trust even when a prediction seems "correct" but is made for the wrong reasons (figures 2.4-2.7) [71].

Chapter 3

Architecture

This chapter describes the autonomous learning system approach proposed by the Neurat Project [72] for financial fraud detection. Figure 3.1 provides a general overview of the system, from a data flow perspective.

This problem involves the concept of Knowledge Management and Decision Support Systems (KMDSS), in the sense that there is a huge amount of data that arrives frequently and needs to be transformed and processed in order to create valuable information so decision-makers can explain the reasons behind their decisions. In this case study, the goal is to support the decision-making process by an Auditor in the fraud detection task.

It also involves the concept of Human Centered Computing (HCC) in the sense that it focuses on an intelligent learning environment with human interaction, sometimes called “human-in-the-loop” [73]. Thus, this is an Interactive Machine Learning scenario in which the user also needs to introduce knowledge in the framework by adding more information, e.g., suggest more features, and by validating previous knowledge, as depicted in Figure 3.1.

The system has as main data input SAF-T (Standard Audit Files for Tax) files [74]. The SAF-T file is an XML file describing accounting data from organizations, that is sent to national tax authorities and/or external auditors for audit and compliance purposes. These files go through a feature extraction process that creates some relevant features for audit, and through a Rule-based System whose main goal is to further enrich the data based on a set of rules. At the end of this process data is stored in the so-called unlabeled dataset.

From this point on resides the interactive nature of the system. The auditor picks instances from the unlabeled data, analyzes them, provides her/his own structured feedback, and saves the changes. Auditor feedback may include changing the values in the database and/or providing details and a justification about the rationale followed. Cases that have been processed by the auditor are regarded as having been validated by an expert and are then added to the labeled dataset. This dataset constitutes the input for the training of models that can predict, for instance, the likeliness of fraud of a given instance.

The auditor can also suggest new features and new values for existing features (when they are enumerations) which will, if approved, be added to the feature extraction process. These user-defined features have however a particularity: they may not result from the feature extraction process. When it is not possible to calculate them in the feature extraction phase (because they do not depend directly on the input data, for instance) they can be "guessed" by specifically trained models. That is, once there is enough

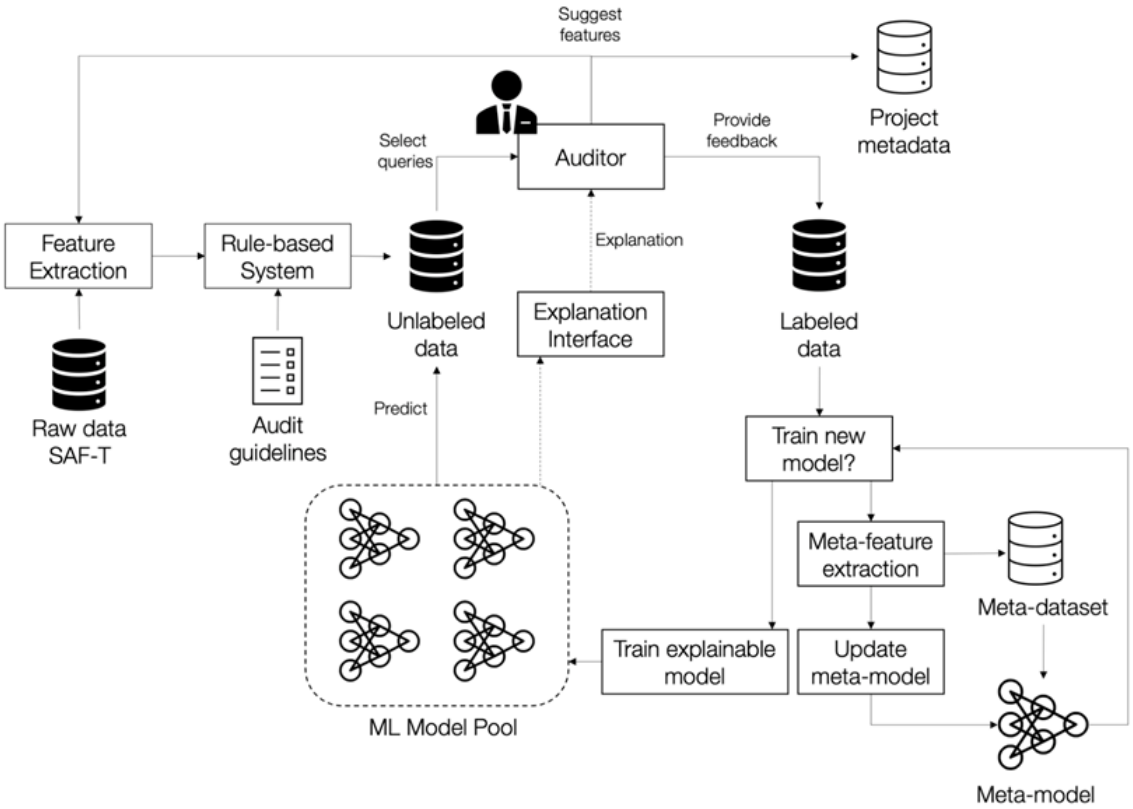

Figure 3.1: Overview of the main functionalities that constitute the developed system.

labeled data, models can be trained to predict the value of each of these user-defined variables. These predictions are later validated by the auditors. This means that in any given time there are multiple models in the so-called model-pool: one for predicting the main dependent variable (likeliness of fraud) and others for predicting the values of specific user-defined variables. The system also includes an explanatory interface, for creating human-friendly justifications for the predictions, based on the XDT developed, explained in chapter 4.

One key aspect about this system is thus that the labeled dataset changes over time, as auditors provide their feedback. This has as main implications that models run the risk of becoming outdated. One solution to this problem would be to frequently re-train these models, in order to keep them up-to-date. However, given the potential number of models and the increasing amount of data, this operation becomes computationally very expensive with time. Moreover, most of the times it can happen that the new data is not significantly different from the existing one, and the new models are also similar to the previous ones in terms of performance. Training new models in this situation is thus a waste of resources. The approach followed for addressing this issue consists in trying to predict the key performance indicators of a new model before its training, so that it is only trained if the statistical properties of the new data promise to result in a potentially better model. This approach was developed in this work and is described in chapter 5.

Chapter 4

eXplainable Decision Tree

The main goal of the XDT is to generate elements that can be used to create different types of intelligible explanations for human users. This means that explanations should be domain-dependent and in-line with the users' conceptual models.

To achieve this, we are using a modified version of the CART algorithm[75], an algorithm of the Decision Tree category. A Decision Tree is, in itself, an explainable model: it can be analyzed visually to understand which variables and values are used at each level to take a decision. However, this may be difficult, for example, if the tree is too large. There is also additional information that can be provided that is not explicitly in the tree's structure. In this section, we detail the explainable elements that are generated by the system, to support the Human auditor in decision-making.

This algorithm allows building a Decision Tree from a group of observations. Each node of the tree contains boolean rules about the observations (e.g. value of variable x is greater than y) and each leaf contains the result of the prediction for a given path in the tree. While the tree is being built, the training set is increasingly split at each node, leading to smaller sub-sets of the data. This splitting process ends when one or more stopping criteria are met, which may include a minimum size of the split or a minimum degree of variance/purity.

Variance denotes how much the values for the dependent variable of a split are spread around their mean value (in regression tasks), while purity considers the relative frequency of classes: if all classes have roughly the same frequency, the node is deemed "impure". The Gini index is used in the CART algorithm to measure impurity [76].

Formula 4.1, as proposed by [77], describes the relationship between the outcome y and features x. Each instance of the training set is attributed to a single leaf node (subset R_m). $I\{x \in R_m\}$ is a function that returns 1 if x is in the subset R_m or 0 otherwise. In a regression problem, the predicted outcome $\widehat{y} = c_l$ of a leaf node R_1 is given by the average value of the instances in that same node. The algorithm can be used for both classification and regression tasks.

$$\widehat{y} = \widehat{f}(x) = \sum_{m=1}^{M} c_m I\{x \in R_m\} \tag{4.1}$$

The core of the Explanatory Interface is thus a tree-based model, the so-called *explanatory model*. A different explanatory model is trained to explain the predictions of each model in the pool.

If there is access to the original dataset, the explanatory model can be trained with the original data or with a subset of it.

However, if there is no access to the original dataset as in cases of proprietary models, the explanatory interface is able to generate a synthetic dataset given some basic meta-data about the original dataset (e.g. minimum and maximum value for each variable). First, the data corresponding to the independent variables are generated, at random. The main goal is to cover the search space as much as possible. However, it may also happen that data instances that are unrealistic, in the sense that they would not happen in real life, are generated. While this may lead to an increased complexity of the explainable model, there are no other disadvantages in the sense that, if those instances never happen in real life, no explanations will ever be requested for them either. If they were, these would eventually be nonsensical. These random dataset is then fed to the original model, so that it can predict on each of its instances. The resulting synthetic dataset is then used to train the explainable model. Figure 4.1 details the process of training an explainable model when there is no access to the original dataset by the explanation interface. When this access exists, the original dataset is used directly to train the explainable model.

Whether the explainable model was trained with or without access to the original data, its goal is to be used in parallel with the original model. That is, whenever a new prediction is issued by the original model, its corresponding explainable model does a prediction as well, in order to generate the elements that are used to create explanations.

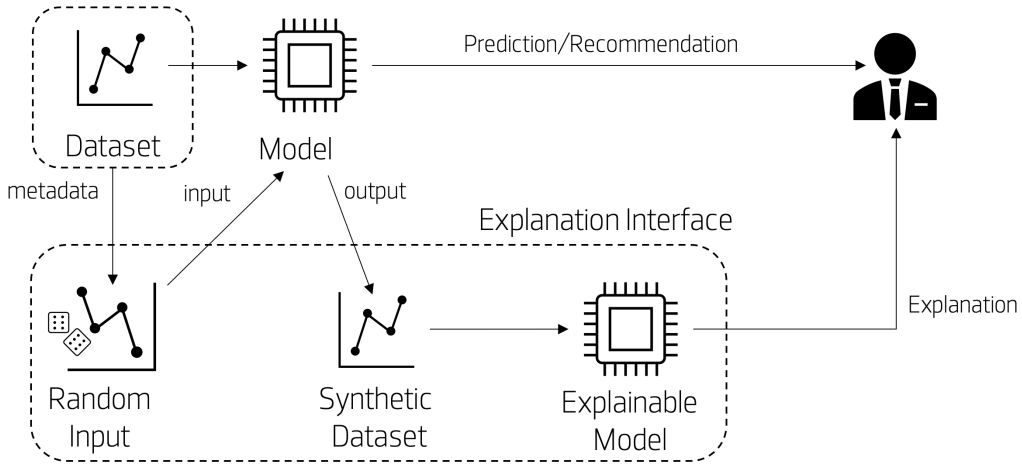

Figure 4.1: Process for generating explanations when there is no access to the original dataset.

The algorithm to create the explainable models works as follows. When the tree is being built and each split generated, additional information is stored in the node which includes: the boolean rule that generates the split (mentioning the variable and the value interval), the prediction $\widehat{y}$ based on that split (i.e. the average or most frequent value, depending on the problem), measures of dispersion or purity (variance, standard deviation and Gini index), and the indexes of the instances in the split.

These values are then used to provide a notion of *confidence* and *support* to the decision-maker. Confidence is given by dispersion and purity measures: the lower the dispersion or the higher the purity, the higher the confidence on the decision is. Support is given by the number of instances in the split: the higher the number of instances, the higher the support is.

This information on the nodes allows to incorporate a group of explainable elements in the user interface. Figure 4.2 shows a prototype of the graphical user interface that is used to provide explanations.

When an auditor wants to analyze a specific instance, she/he selects that instance and is redirected to this interface, which receives the data of the instance, the prediction, and an explanation. The user interface has three main areas, marked in the Figure as (a) - Explore, (b) - Decision path and (c) - Last results.

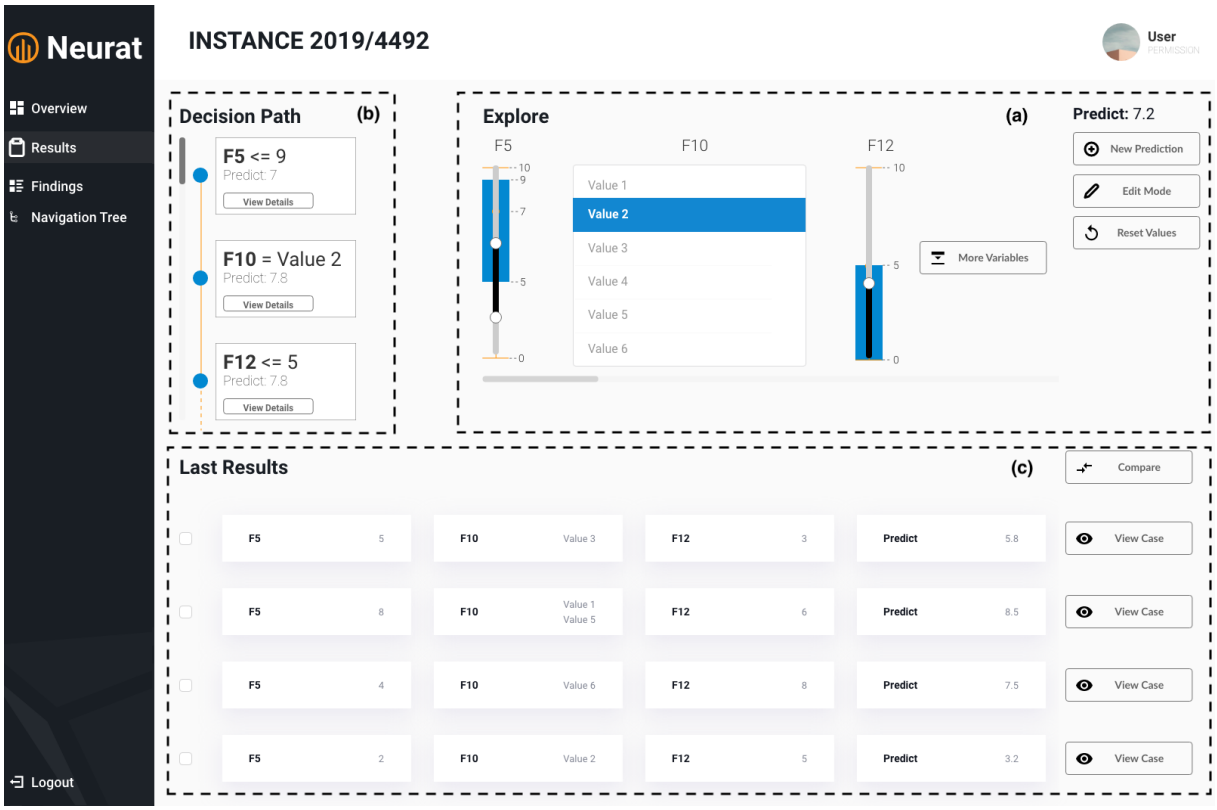

Figure 4.2: Prototype of the main screen of the application, with some explainable elements created, and three main areas highlighted: Explore (a), Decision Path (b) and Last Results (c).

Area (a) allows the user to explore the search space and analyze each feature according to their relative importance. Features and values are collected from the internal nodes when traversing the tree to make a prediction. In this context, feature relevance is based on how much that split/feature decreases dispersion/purity. For each feature that the interface shows the following elements (depending on whether the variable is numeric or nominal): the domain of the feature (range/enumeration of possible values), the interval/values for which the prediction holds (blue bar or values highlighted in blue), and the value of the feature in the instance being audited (gray dot).

This allows the auditor to gain a sense of how *risky* the decision is. If the value of a given feature is very close to the upper or lower limits of the blue bar, it indicates that a slight change of this feature towards the limit would significantly alter the prediction of the tree. Likewise, the size of the blue bar is also related to this sense of risk: the shorter the bar the more risky the decision is. In the case of a nominal feature, multiple values can be highlighted to show for which values of the enumeration the prediction holds. The risk of the decision grows with fewer highlighted values.

In Figure 4.2, the graphical interface is shown in "Edit Mode". This means that the user may change the values of the variables to perform a counterfactual analysis. That is, what would be the prediction if the value of a feature had been v_2 instead of v_1. These "what-if" scenarios allow the auditor to interact with the tree and to understand how predictions would change under different scenarios. This contributes significantly to the interpretability and interactivity of the explanation. The user does this by changing the

value of the features by means of a slider, or by selecting a value from a list. The scenarios created by the user can be added to area (c), to be compared. The user can also reset area (a), returning all the values and the associated prediction to the initial state of the instance being audited.

There is also a pagination mechanism that controls the amount of information provided to the user, to avoid overload. Indeed, depending on the training set, the number of levels/nodes/features on a tree may be too large to be efficiently analyzed by a Human. In that sense, in this interface we show only the n most relevant features. The user can then choose to request additional features (and the associated prediction) by clicking on the "More variables" button. These are gradually added upon request by decreasing relevance.

In the left side of the interface there is the area marked as (b). This area shows the path followed through the tree to make the prediction. Like in (a), this area may not show the whole path as it implements the same pagination mechanism: when features are added to (a) they are also added to (b). This element allows the user to understand (part of) the reasons for a given prediction: "because feature f_1 is smaller or equal than v_1 and feature f_2 equals v_2".

In this area the user may also click on a specific node to see its details (Figure 4.3). The details show, in the left side, the information for the feature that is also visible on (a). On the center and right, the "details" modal provides information regrading the *confidence* and *support* of the prediction. The graphical representation shows the prediction (blue dot) and the interval given by the standard deviation. A smaller interval indicates an increased confidence as instances in this split are more closely distributed around the mean, and vice-versa.

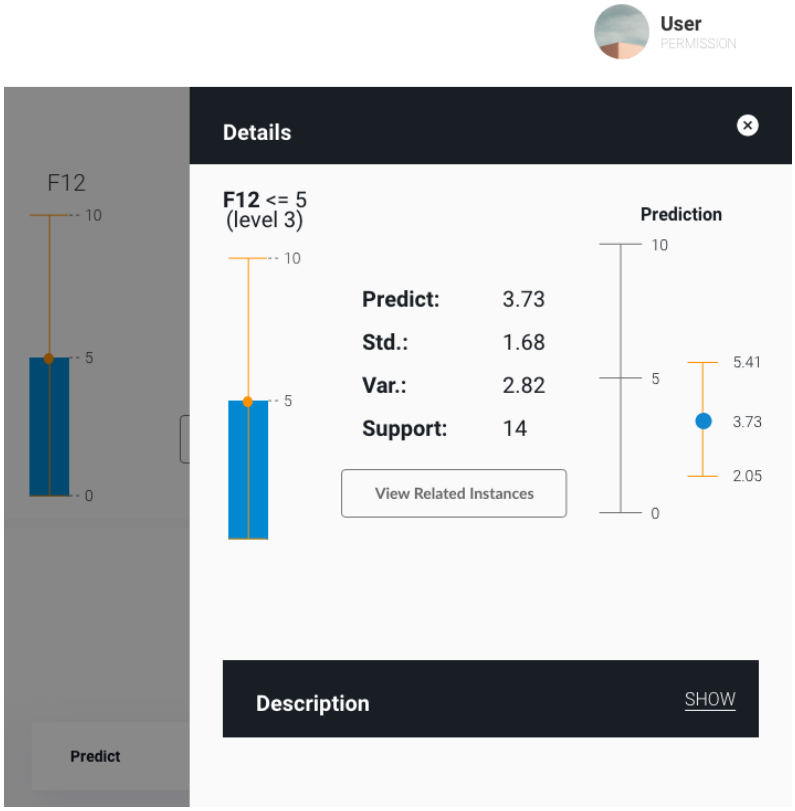

Figure 4.3: Details of a split node, with confidence and support measures.

The central part of the modal shows values which include the support (number of instances in this split) and a button that allows the user to access the instances that fall into this split. The user may thus

visualize the instances, which are shown sorted by similarity to the current instance in descending order. Similarity is calculated based on a weighted sum of differences, given by the euclidean distance for numerical variables and by the cosine similarity for the vector of nominal data (if any). While visualizing specific instances the user may add them to a list for comparison (area (c)).

As the user moves down the path, splits become smaller but confidence increases. It is up to the user to decide how far down to travel: an early stop may lead to a more general decision (with high support and potential low confidence), while going further down will lead to low support but high confidence. Finally, in area (c) the user has access to a list of previous prediction results (the scenarios that were simulated) and/or to actual instances that were visualized by the user and added for comparison. This allows to more easily compare a group of scenarios or real cases and their results.

4.1 REST API

The purpose of this section is to help developers that want to use the XDT through the interaction with a well documented Representational State Transfer (REST) Application Programming Interface (API) via the open-source Swagger framework.

First things first, to start the execution of the API, one is advised to create a virtual environment to install all the dependencies needed, e.g. with the following commands:

For Windows systems:

```
pip install virtualenv
virtualenv venv
venv/scripts/activate
pip install -r requirements.txt
```

For Linux based systems:

```
pip3 install virtualenv
virtualenv venv
source venv/bin/activate
pip3 install -r requirements.txt
```

After the execution of the previous steps, with the dependencies listed in appendix F, one should see the word "venv" on the left side of the command line, meaning that the virtual environment is activated, and now it's possible to run the API with:

```
python rest_plus.py (on Windows)
python3 rest_plus.py (on Linux)
```

The last command will start the execution of the RESP API that allow the developers to see and test the endpoints provided through the Swagger UI (figures 4.4-4.5 and appendix B.1-B.5).

In figure 4.5 the models to deal with the API are shown, used to send POST requests, and the responses given by the endpoints. E.g. the dataset model (figure B.3) not only characterize a dataset, representing the type of dataset, features and instances, but also can indicate if it's ready to use (in that case one should

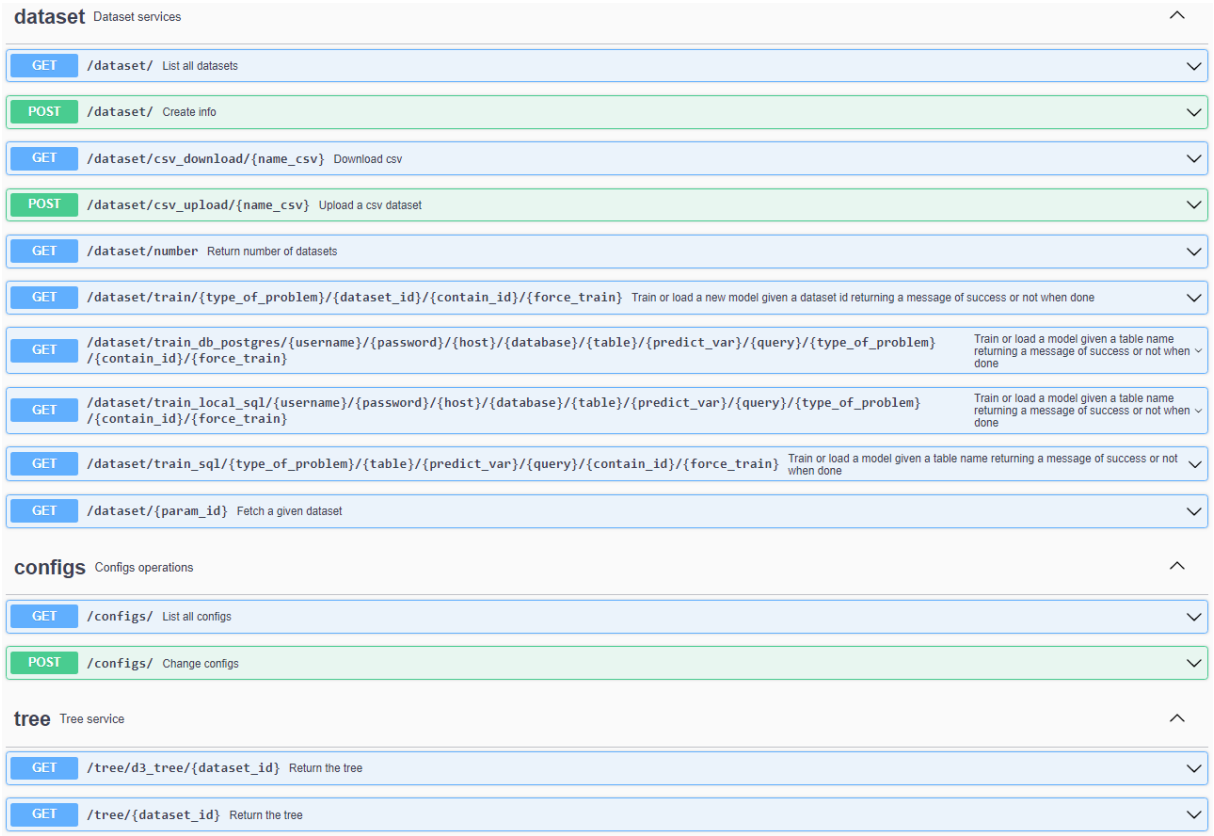

Figure 4.4: XDT API endpoints

upload a dataset to API and send this kind of information in a POST request). More information about a specific model can be seen as a user interacts with the Swagger interface and open the associated links.

In short, it's possible to train a dataset in three different sources and formats: CSV files, PostgreSQL and MySQL databases (figure 4.4). The stop criteria and other configurations related to the training of the decision tree can be changed (figure B.4). Finally, the trained tree can be shown in two different formats: the one used in the training, and one util transformation to be possible to see a visual representation of the tree using the D3 React library (figure E.1).

To initiate the XDT frontend, it's needed to have the NodeJs installed and the package manager NPM, and run the following commands:

```
npm install
npm start
```

With this frontend, one can see and interact with the trained trees given the name of the dataset, with different types of visualizations provided by the D3 library, e.g. changing the type of connections between the nodes, open and collapse nodes, vertical or horizontal orientation, etc.

Others endpoints provided by API are for predictions, test, and for dataset services (for download and upload datasets), shown in appendix B.

In appendix C is shown an example of a tree and in D examples of predictions and explanations given by the decision tree. In short, the JSON attribute "human_counterfactual_explanation" compiles all the given reasons for the prediction done by the decision tree, detailed in the previous JSON attributes

Figure 4.5: XDT API models

and explains to the final user why the prediction is what it is and why is not another one exploring the counterfactual analysis definition.

Moreover, one can integrate the XDT in existing ML systems, e.g. to explain the predictions given by a black-box model like a deep learning one.

Chapter 5

Model Performance Prediction

Every ML model needs computational power. And of course, it also needs data. But how much data are necessary to build the best ML model possible, and how many times do we need to re-train a model so that it does not become obsolete as data change? This kind of questions are the ones that can reduce unnecessary costs for a company.

The main goal of the Performance Prediction module is to predict one or more performance metrics of a future model, if it is trained from a dataset with given statistical properties. The underlying assumption is that certain features of a dataset can be associated to the quality of the resulting dataset. For instance, it is generally accepted that a dataset in which there is a significant amount of missing data leads to a poorer model. Likewise, a larger number of variables and instances tends to result in better models. Other examples could be given, related for instance with the complexity of the data, its distribution, the relationship between its variables, or numerous statistical and information-theoretic features that described the characteristics of a dataset.

This chapter describes the approach implemented for predicting the performance metrics of a future model. The process by which the approach was implemented and its results validated is detailed in Figure 5.1.

The process starts by consuming a large amount of datasets, called the training datasets. These datasets are first pre-processed, namely to normalize the values of the dependent variable of each one. Then, these datasets go through a process of meta-feature extraction. Meta-features describe important statistical and information-theoretic characteristics of the dataset, that are expected to be related in some degree to the quality of a model trained with the dataset.

In the beginning of the project, the meta-features were calculated using the Python language, simple ones, like number of instances and features of the datasets, percentage of missing values and features bad represented (i.e. imbalanced classed, e.g. 90% of one category and only 10% of another). But one recent Python library was found to be particularly useful for the meta-feature extraction, the pymfe library [78]. With the help of this library, it's possible to extract more than 100 meta-features, but only a subset of 59 were used in this work because not all datasets can extract the same meta-features. In [79] a complete list and description of all meta-features is available, and in [80] the API documentation.

The meta-features used in the present work are of three types: (i) simple measures such as the number of instances and columns; (ii) statistics-based, such as the sparsity of the attributes and the number of outliers; and (iii) information-theoretic features such as the information gain of attributes. In (i), those

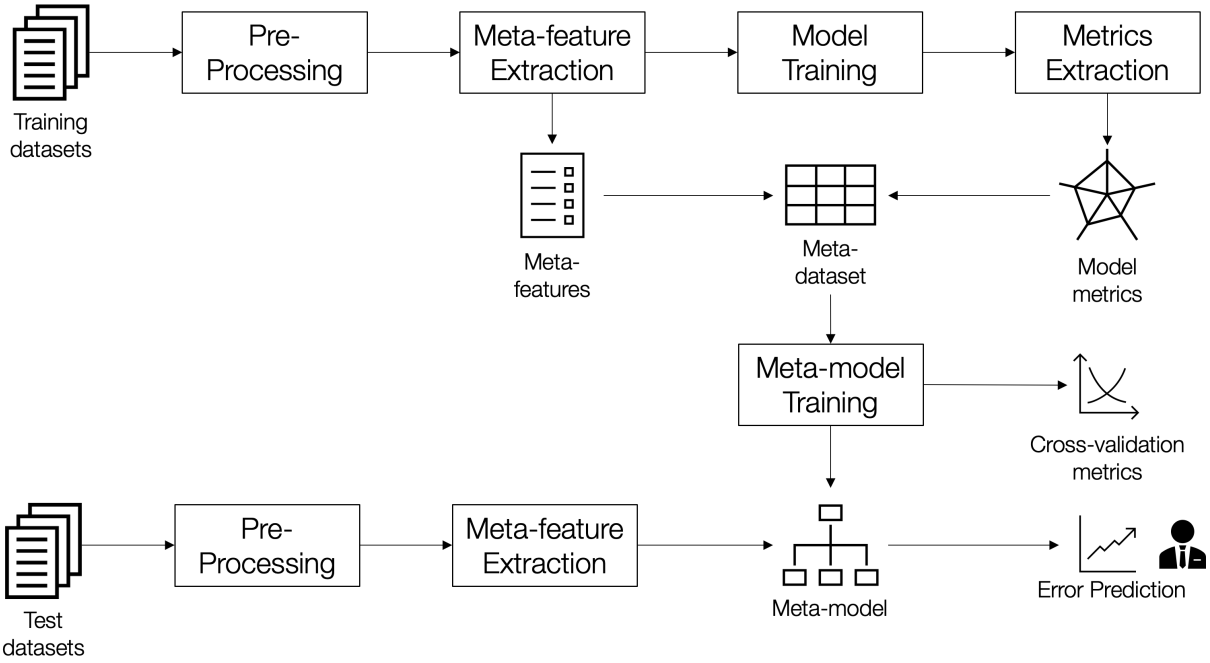

Figure 5.1: Methodology followed to validate the proposed approach for model performance prediction.

measures are also advantageously combined to give other measures more appropriate for specific tasks, for example by taking products, ratios, or logarithms.

Once the meta-features are extracted, an actual model is trained with the dataset, and its performance metrics are extracted (e.g. RMSE, mae, r^2) and the time (as a measure of complexity) needed to train each of those models. Models were trained using the H2O library [81]. A Random Forest Algorithm was trained using 20 trees with a maximum depth of 20 levels.

The meta-features of all the datasets are then combined with the performance metrics of the corresponding models, to generate what we call the meta-dataset. In this meta-dataset, meta-features are the independent variables while performance metrics and the model training time are the dependent variables. Typically, the meta-dataset has one line for each dataset/model combination, although we followed a different approach for validation purposes, as detailed in Section 6.2.

Once the dataset has a large enough dimension, the so-called meta-model is trained. This meta-model is trained from the meta-dataset by selecting all the independent variables and one of the dependent variables, corresponding to the performance metric that one intends to predict. The performance of the meta-model itself is obtained through cross-validation.

The previously described approach considers the training of a number of models that can be relatively large. However, once the meta-model is trained, it can be used to predict the performance metric of future models. Given that the models in the pool will only be re-trained if the meta-model predicts a performance improvement, the number of models trained in the long-term is expected to be smaller than if the models in the pool were updated on a regular basis. Moreover, if the meta-model is robust enough, it can be used to predict the performance on any kind of problem.

The goal is, indeed, that it can be used in any Machine Learning problem, independently of the nature of the problem. This can be done by simply extracting the same meta-features of the dataset of the new problem, and using the meta-model for predicting the intended performance metrics. This is detailed in the lower part of Figure 5.1, in which a group of datasets that were not used for training, deemed test

datasets, are used to assess the performance of the meta-model.

The difference, in this case, is that the models are actually trained so that the predictions of the meta-model can be compared against the actual performance metrics of the models. In a real use of the system, such as that described in the previous section, the meta-model would be used for predicting the performance of a model when trained on a specific set of data, and the model would only be trained if the prediction was for better performance metrics than that of the current model. Thus a reduction in the number of models trained, and consequently in the amount of necessary computational resources, is achieved. Section 6.2 details the process through which this approach was validated.

This process was followed to train a meta-model for each relevant performance metric, the only change is to select the respective dependent variable, (e.g. RMSE, MAE, r^2, RMSLE, training time). Thus, the process results in multiple meta-models, which can be used to predict the different performance metrics of a future model for a given ML problem.

5.1 Find the Best Meta-Model

There are two ways of thinking to solve the problem of predicting the error of a future ML model based on previous data that describes the relationship between the meta-features extracted for a particular dataset and the error metrics for the model trained. The first, more achievable one, is to simply analyze the previous data of one particular domain, build the meta-model based on this data alone, and therefore, we can only try to predict the error of future ML models of the same domain.

Because, if we try to predict the error of some other domain, we will fail badly, as the correlation between the meta-features and the error metrics may be completely different from the other domain, and we don't have training data that also describes the influence of the meta-features on the error metrics of this particular domain. Also, the error metrics could be in a completely different range (e.g. 0 to 1 against 0 to 1,000,000), and the model built for only a particular range will not give predictions with different ranges. This goes against the rationale of a ML model, predicting unusual possibilities.

However, this still is a valid approach and can be useful in batch scenarios (e.g. to know how many and which data can build the best possible model) as so in streaming data scenarios, in which new data of the same domain arrive at the ML system, and we need to find if it will be useful to spend resources to train a new model.

Howsoever, the other way of thinking and the one followed in this work, was to build a meta-model that can, theoretically, predict the future error of every ML model training with a dataset that only a subset was used in the training of the meta-model, but also, try to predict datasets that the meta-model never "seen" before.

To this end, several meta-datasets with different configurations and approaches were built and therefore, several meta-models were created and object of inference. More precisely, of the 47 datasets used, several train-test splits were performed in order to find the best possible meta-model.

Besides that, mainly two approaches were tested, namely: (i) each dataset was only extracted meta-features with all instances once, and (ii) a streaming scenario was simulated in which, for each dataset, data were considered at 200-instances intervals. Multiple instances of each dataset were added to the meta-dataset (one for each new 200 instances of the original dataset), with the goal of having a larger meta-dataset.

#	RMSE	R^2	MRD	RMSLE
meta-model 1	0.017450	0.980312	0.000304	0.014650
meta-model 2	0.096803	0.297332	0.009371	0.081110
meta-model 3	0.016280	0.983094	0.000265	0.013624
meta-model 4	0.007925	0.908762	0.000063	0.007467

Table 5.1: 10-fold cross validation metrics for four different meta-models

Table 5.1 describes the 10-fold cross validation metrics obtained by the meta-models built with different configurations. The "meta-data-1" refers to a meta-dataset with 974 meta-examples using 43 datasets and the simulated streaming scenario previously described, with meta-instances added at 200-instances intervals. The other 4 datasets were used to test.

The "meta-data-2" used only one meta-example from each of the 43 datasets, considering all the instances once, as the opposite approach followed by "meta-data-1", previously described.

The "meta-data-3" was created following the same methodology as "meta-data-1" but using only 37 datasets. It resulted in 959 meta-examples. As previously mentioned, the other datasets were used to test, in this case 10.

Finally, "meta-data-4" used only one meta-example from each 46 datasets, considering all the data of each. The purpose of this meta-dataset is to use as many data sources as possible in the training of the meta-model to simulate the arrival of more data sources to the meta-feature extraction module. Thus, only was left to train-test split validation the tax fraud detection problem to compare the same case study with other train-test splits.

In short, two approaches were tested between all meta-datasets: (i) a 10-fold cross-validation aimed at test the proposed approach to different problems and understand what meta-dataset-built approach performs better, and (ii) a train-test split validation to the case study of tax fraud detection.

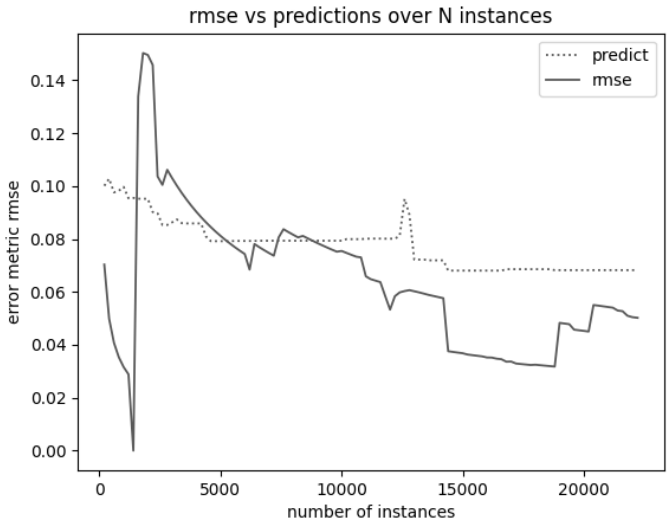

Figure 5.2: Evolution of observed vs. predict RMSE over time in an IML setting for fraud detection with "meta-data-1".

In Figures 5.2 - 5.4, one can see the predictions and the observed RMSE error metric for the tax fraud detection case study for each meta-model trained, in which the approach that perform better was the

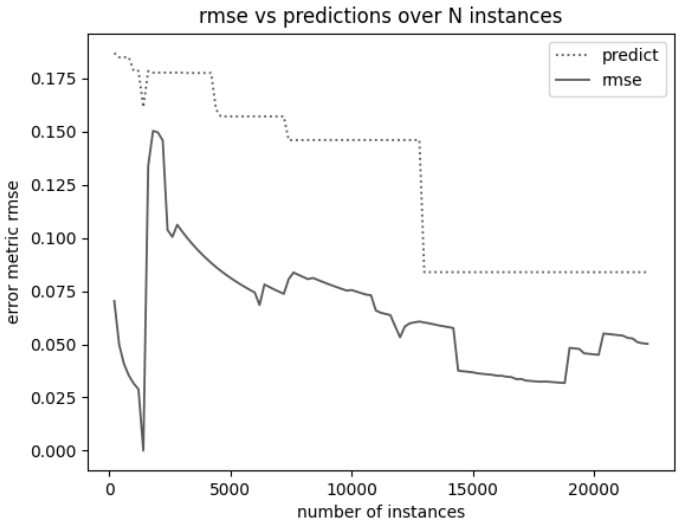

Figure 5.3: Evolution of observed vs. predict RMSE over time in an IML setting for fraud detection with "meta-data-3".

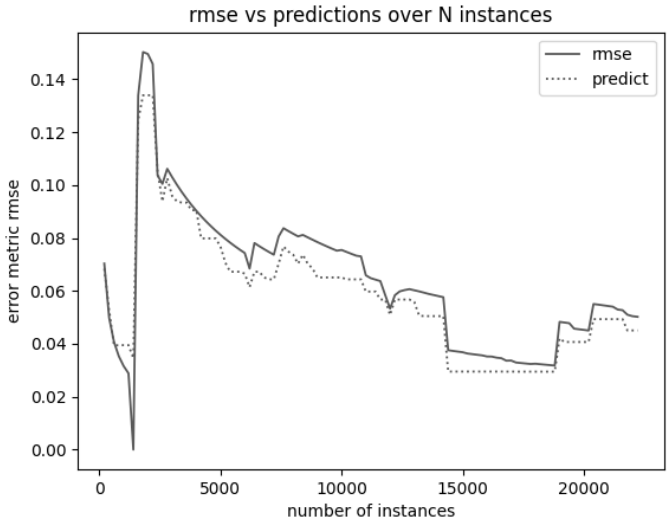

Figure 5.4: Evolution of observed vs. predict RMSE over time in an IML setting for fraud detection with "meta-data-4".

meta-model 4, that uses only one meta-example for each dataset, created based on all the instances of each dataset, using 46 distinct datasets (figure 5.4).

In this study, one can conclude that with the increase of the datasets in which meta-features are extracted, the meta-model performs better when only the last meta-example was used for each input dataset. This result encourages future works to continue pursuing this approach and investigate more datasets to be added to the meta-feature extraction module.

"Measure what can be measured, and make measurable what cannot be measured."

GALILEO GALILEI

Chapter 6

Validation and Results

6.1 eXplainable Decision Tree

In appendix E one can see different types of visualization of trees to different datasets with the help of D3 library to React. E.g, In order to explain the traffic prediction in New York City center, represented by one or more datasets uploaded to the REST API and consequently trained a decision trees to each dataset, one can visualize and interact with the decision tree in many ways. In figure E.2 a horizontal orientation of the tree is viewed, opposed to a vertical orientation in figure E.3. Other view settings can be seen in E.1, or with the execution of the frontend with the commands show in section 4.1.

A shaded node represent a node that has more nodes or leafs, and therefore it is possible to expand to see the children or collapse to a better overall picture. On the other hand, a white node represent a leaf, that is, a node that unique represent a prediction, and is no longer divided due to stopping criteria. Regardless of the type of the node, each node gives a prediction next to a dozen useful attributes to understand the distribution of data, and of course, an explanation to that prediction based on the path from the root of the tree until the present node, and other explanation techniques (as discussed in chapter 4 and shown in appendix C and D).

Visualizing the tree, a user can understand why the model behave the way it behaves, and how predictions are given and why. E.g. in figure 6.1, in addition to the forecast, XDT explains that the number of people (or census) is approximately 19 (see the attribute “predict” in the leaf node represented by the white circle) because the clouds are approximately below 270. It also issues an alert on the response given by the API: if the value of clouds was 7 units lower, the new forecast would be 12, which in this case remained the same (for this split point in the first level of the tree). Continuing the explanation, the census of the New York City center is approximately 11 in the second level of the tree for this particular path to get to the leaf node, because the “tempnormal” feature exceeds 260.38 units. The explanation continues until the pressure attribute, which is below 263.92 units and the prediction given by the leaf node is 18.33, which means approximately 19 people, supported by three past cases and this leaf node does not have any outliers. Also, with this visualization it is easy to see the other possible paths of the tree, which is useful to know this kind of extra information about the forecast.

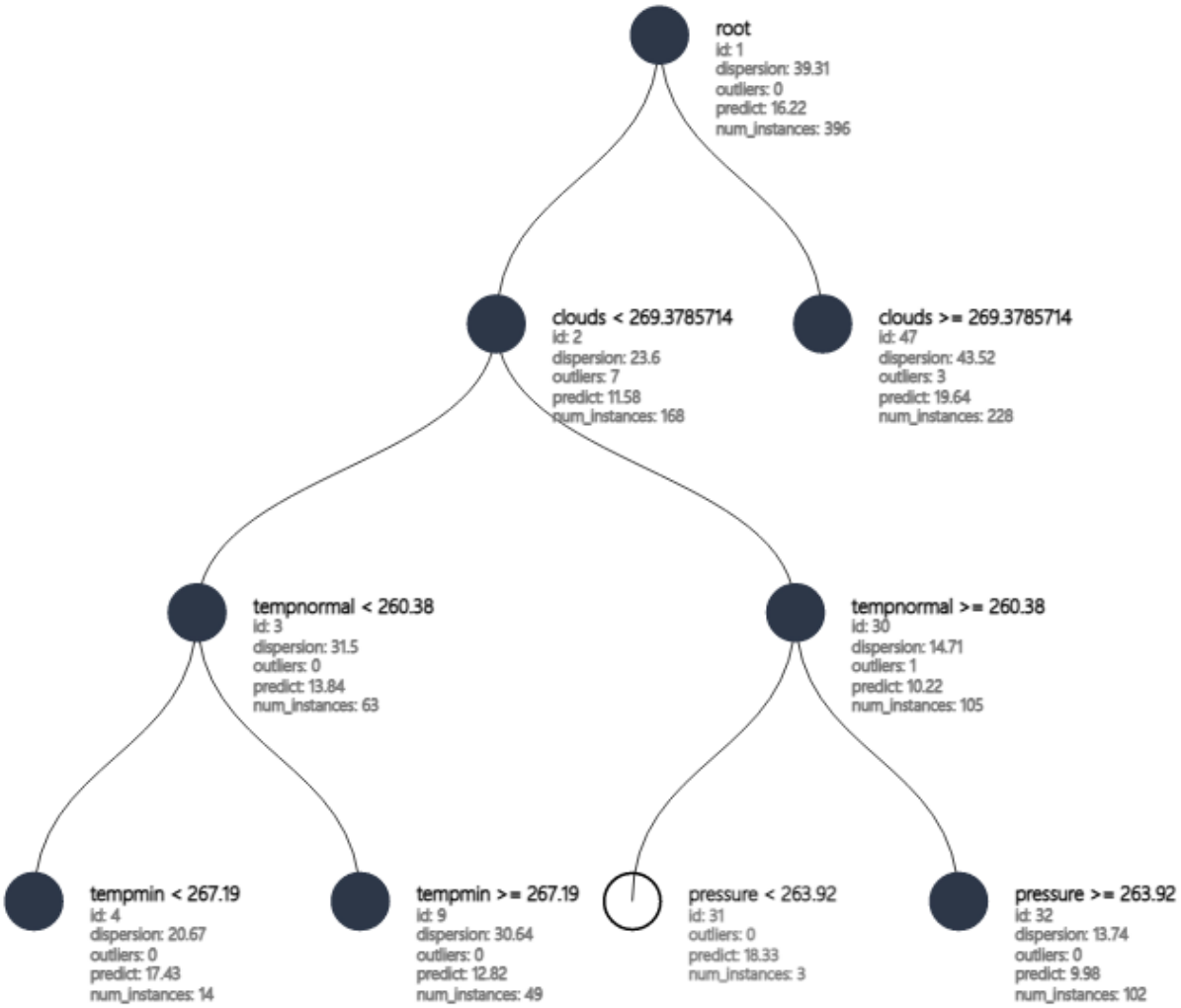

Figure 6.1: XDT to traffic prediction for New York City center

6.2 Meta-Learning

The methodology proposed for creating a meta-dataset for predicting model performance, detailed in Chapter 5, was instantiated to assess its suitability to solve the proposed challenge. This section details the results obtained.

A total of 50 datasets, detailed in Table A.2 (in Appendix A), were used in the process, covering both regression and classification problems, as well as streaming and batch scenarios. Of these, 44 were used for training the meta-model and 6 were used for testing it. Concerning the test datasets, 3 are batch datasets while the other 3 are streaming datasets. Given that streaming and batch ML problems are fundamentally different, the meta-model was evaluated differently for each case.

We also acknowledge that 44 datasets for training is a relatively small number, as it would result in a meta-dataset with only 44 instances. In order to increase the number of instances, the following approach was followed. Each dataset was streamed in blocks of 200 instances. A first model is trained with the first 200 instances, and additional models are trained by adding 200 instances to the previous dataset. That is, for a dataset with 650 instances, 4 models are trained: the first with 200 lines, the second with 400, the third with 600, and the last one with the whole 650.

Given the number and size of the datasets used for training and the approach described, the resulting

meta-dataset contains a total of 673 instances.

The meta-dataset is composed of 111 columns. Of these, 105 correspond to the meta-features extracted from the datasets, while the remaining six correspond to relevant performance metrics: training time, RMSE, MAE, MSE, RMSLE and r^2. This means that the meta-dataset can be used to train six different models, one to predict each of these individual metrics.

Table 6.1 describes some of the meta-features considered. The criteria for selecting a sample of the meta-features was based on the relative importance of each one during the training of the meta-model.

Table 6.1: Ten most relevant meta-features, out of 105 used to build the meta-model.

Meta-feature	**Scaled Importance**	**Description**
linear_discr	100	Linear Discriminant classifier
mut_inf.sd	48.38	Standard deviation of mutual information
sparsity.sd	29.87	Standard deviation of sparsity metric
eq_num_attr	20.21	Attributes equivalent for a predictive task
one_itemset.sd	14.15	Standard deviation of one itemset
elite_nn.sd.relative	10.21	Performance of Elite Nearest Neighbor
one_itemset.mean	10.05	Mean of one itemset
ns_ratio	9.73	Noisiness of attributes
attr_ent.mean	8.29	Mean of Shannon's entropy
mut_inf.mean	7.25	Mean of mutual information

To assess the quality of the meta-model during the training phase, a 10-fold cross-validation approach was followed. Metrics were computed for each holdout prediction, and averaged at the end, resulting in the following values: $RMSE = 0.000355$, $R^2 = 0.98$ and $MAE = 0.007$.

Next, the performance of the meta-model was assessed on the test datasets. To this end, a different approach was followed depending on whether it was being evaluated on a batch or streaming dataset. In any case, the same meta-model already trained was used as the basis for evaluation.

For batch datasets, its meta-features were extracted and the performance of a model (RMSE) was predicted by the meta-model. An actual Random Forest was then trained with each batch dataset, and its actual RMSE value was obtained. Table 6.2 shows, for each dataset, the RMSE predicted by the meta-model and the actual RMSE of the trained Random Forest, which are relatively similar, showing that the meta-model does a good job predicting the error.

Table 6.2: Observed vs. predicted RMSE for the three test batch datasets.

Dataset	**Observed RMSE**	**Predicted RMSE**
Wine quality	0.102	0.111
Diabetes	0.400	0.321
Breast Cancer	0.161	0.093

Streaming datasets were tested differently, to simulate a realistic streaming scenario. Specifically, for each dataset, data were streamed in blocks of 200 instances. For each block, its meta-features were extracted, and the meta-model was used to predict the performance of a model trained on that block. Then, a Random Forest was actually trained with the block. The RMSE of the resulting model was compared with the RMSE predicted by the meta-model (Figure 6.2), to assess the quality of the meta-model. For each model trained, a new instance was added to the meta-dataset, as previously described. The meta-model was updated at 10 block intervals (2000 instances of streamed data).

So, the main difference between the batch and the streaming scenarios is that in the latter the meta-model is updated as new data are received.

Given the size of the Wine Quality dataset, it was tested following both the batch and the streaming approach.

Figure 6.2 shows the observed vs. the predicted RMSE for the 3 streaming datasets and the Wine Quality one. The data points are generally close to the diagonal, which indicates a relatively good predictive performance. However, in the AWS dataset there are some instances in which the prediction is slightly off, and in the Wine Quality dataset there seems to be a tendency to overestimate the value of the RMSE. Nonetheless, the values are close to the diagonal.

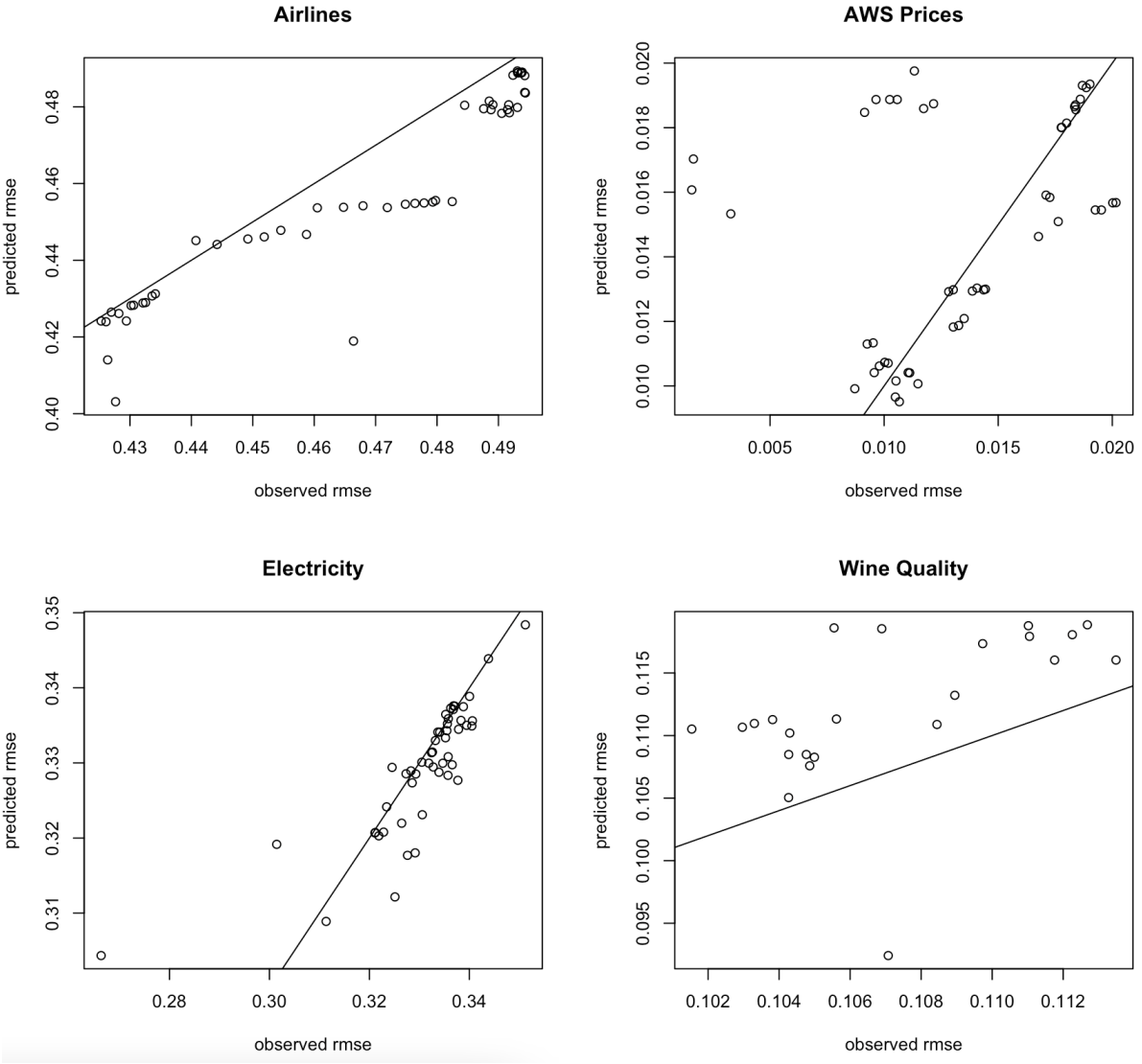

Figure 6.2: Observed vs. predicted RMSE for the 3 streaming datasets and the wine quality dataset, with models trained at 200-instance intervals. The solid line represents the diagonal.

Table 6.3 shows how the performance of the meta-model evolved over time, as new versions of it were trained, including the streaming data received so far. It can be seen that, even for streaming data, the performance of the meta-model is maintained or improves over time, as new data arrives and is incorporated into the meta-model.

Figure 6.3, on one hand, shows the evolution of RMSE for the Wine Quality dataset over time, as

Table 6.3: Evolution of the RMSE of the meta-model as more instances of data are incorporated.

Dataset	***N*=2000**	***N*=4000**	***N*=6000**	***N*=8000**	***N*=10000**
Airlines	0.018	0.005	0.020	0.0105	0.006
AWS Prices	0.0103	0.001	0.001	0.003	0.0003
Electricity	0.013	0.008	0.003	0.003	0.001

each new model was trained using 200 additional instances of data. It shows a tendency of the RMSE of the models to decrease as more instances of data are included. The RMSE predicted by the model tends to follow this decrease, albeit with a positive bias, as described previously. The predictions are, nonetheless, in line and close to the observed values.

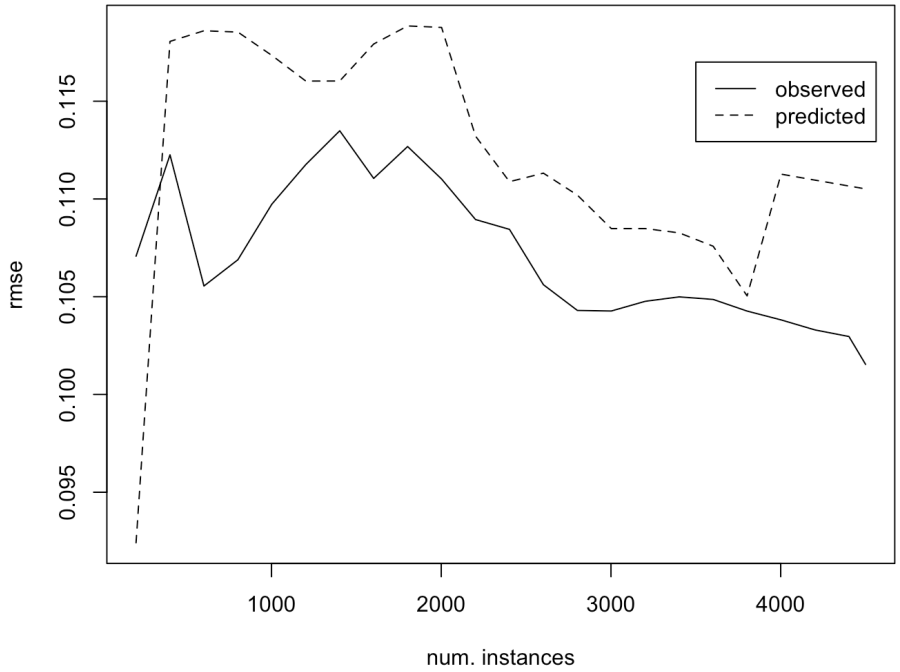

Figure 6.3: Evolution of the predicted and observed RMSE for the wine quality dataset, with models trained at 200-instance intervals.

Figure 6.4, on the other hand, shows the evolution of a selected group of meta-features of each consecutive subset of data, and how it relates to the evolution of the RMSE. While the RMSE data is the same of Figure 6.3, the different scale makes it more difficult to observe its descent over time.

6.2.1 Minimize Model Training

Finally, to see how this approach avoids the training of models over time, a new model is only trained if the meta-model predicts changes by at least a given threshold, otherwise no model is trained. Nonetheless, for validation purposes, all the models are still trained, to ascertain the accuracy of the meta-models.

Specifically, considering two different thresholds (5% and 10%), a new model is only trained if the meta-model predicts a change of 5% or more in RMSE, and in other test case scenario, a new model is trained only if the meta-model predicts a change of 10% or more.

The performance of each experiment is measured through several indicators: the difference between the predicted and observed RMSE (also calculated through the RMSE), the accuracy (measured as the percentage of times in which the meta-model predicted that the RMSE would decrease by at least the

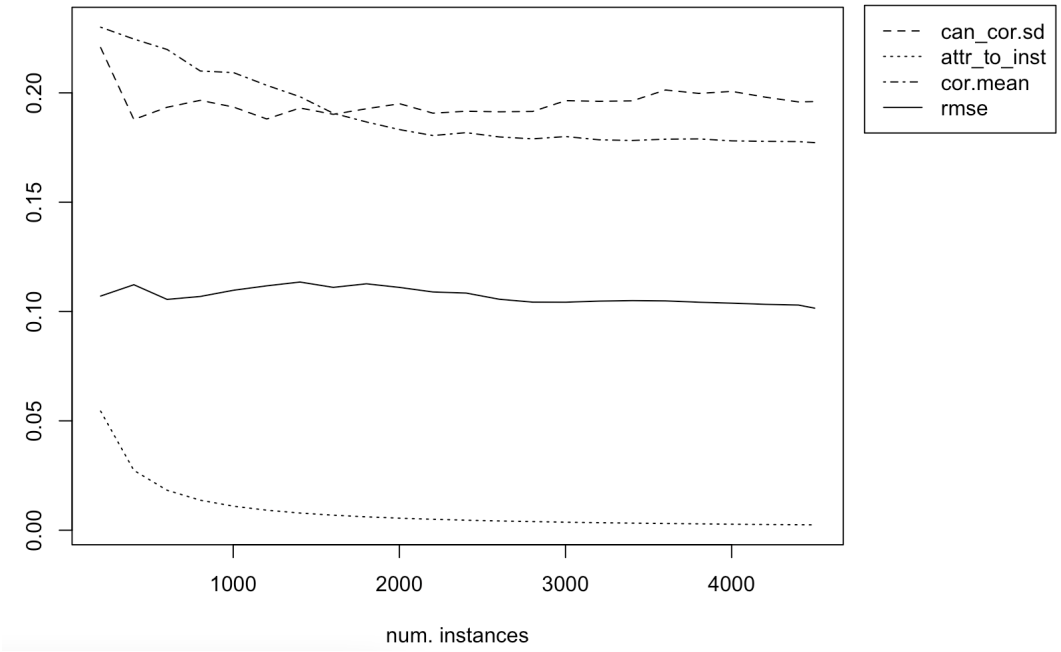

Figure 6.4: Evolution of the values of several meta-features and corresponding RMSE for the wine quality dataset, with models trained at 200-instances intervals.

corresponding threshold and it actually did), and the number of models whose training would have been avoided if the proposed approach was being used. The results are detailed in table 6.4.

One can summarize that with the meta-learning approach described in this dissertation, the percentage of avoided models in a streaming scenario varies between from 63.27% to 95.92%, depending on the selected threshold to re-train the model (see table 6.4).

Also, figure 6.5 shows the observed and predicted RMSE for the case study of Tax Fraud Detection with models trained at 200-instance intervals for validation purposes, along with the model training points suggested by the meta-model. The green points are barely visible because the line of prediction overlaps the observed RMSE. These green points mark the time in which the meta-model predicted an error decrease of 10% and the red points mark moments in which the model was re-trained given the meta-model predictions indicated by the green points.

As figure 6.5 shows, the system only suggests training a new model by approximately 2000 instances. After that point, as the error will never be as good with that number of instances, no other models will be trained.

Nevertheless, a small error does not always mean a better model, as in the case of an overfitted model. In that case, one can simply re-train a model following two different criteria: the time the last model was trained (e.g. ensuring that at least one model is trained every month) or by the number of instances added (e.g. ensuring that at least one model is trained every 5000 instances), even if the error is worse than the previous available model.

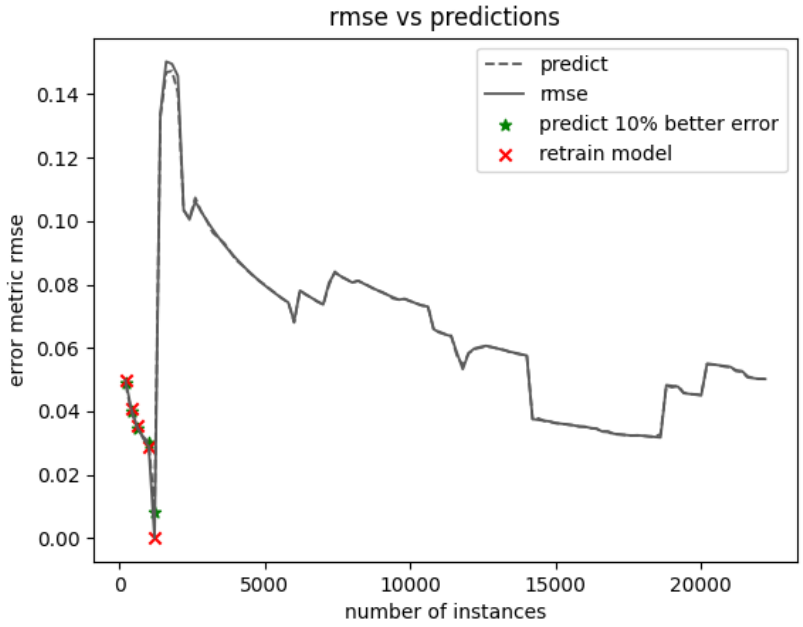

Figure 6.5: Predicted and observed RMSE for the case study of Tax Fraud Detection trained at 200-instance intervals, highlighting the model training points suggested by the meta-model.

Table 6.4: Summary of the performance indicators with the percentage of avoided models for the three streaming test datasets.

Dataset	**Threshold (%)**	**RMSE**	**Accuracy**	**#Avoided Models**
Airlines	5	0.012	100% (49/49)	95.92% (47/49)
Airlines	10	0.012	97.96% (48/49)	97.96% (48/49)
AWS	5	0.004	81.63% (40/49)	63.27% (31/49)
AWS	10	0.004	95.92% (47/49)	87.76% (43/49)
Electricity	5	0.005	100% (49/49)	95.92% (47/49)
Electricity	10	0.005	100% (49/49)	95.92% (47/49)

"Knowing what to measure and how to measure it makes a complicated world much less so. If you learn to look at data in the right way, you can explain riddles that otherwise might have seemed impossible. Because there is nothing like the sheer power of numbers to scrub away layers of confusion and contradiction."

LEVITT AND DUBNER

Chapter 7

Conclusions and Future Work

7.1 Conclusion

Machine Learning (ML) has emerged in the last years as the main solution to many of nowadays' data-based decision problems. However, while new and more powerful algorithms and the increasing availability of computational resources contributed to a widespread use of ML, significant challenges still remain. Two of the most significant nowadays are the need to explain a model's predictions, and the significant costs of training and re-training models, especially with large datasets or in streaming scenarios.

With that in mind, in this work we developed a bundle of services to help build better ML systems, with a greater trust in the models and with reduced costs.

First, to help a decision support agent trust in the models' prediction, an eXplainable Decision Tree (XDT) was developed from scratch that contain state-of-the-art techniques to help understand the value of some prediction and how confident the model is. In addition, it is possible to explore neighbor predictions using what is known as counterfactual-analysis, e.g. exploring what-if scenarios. Also, it is possible to understand whether certain AI models are racist, sexist, or with other types of discrimination. As well as making more sensible decisions, wise and well-informed.

Second, for reducing costs in these systems, a novel approach to predict the error of a ML model was developed, using what is known as meta-learning. The developed solution can be applied in any domain, as the meta-dataset was created based on different datasets, thus covering a wide range of problems. The datasets are from a broad range of domain problems and objectives, to experiment if it was possible to predict with same performance the error of different problems.

Essentially, it is a two-step process. First, a meta-dataset is built that details the relationship between the characteristics of datasets and the performance of the models trained with those datasets. Then, a meta-model is trained with this meta-dataset, which is used to predict the performance of a given model, if it is trained with data containing certain characteristics (i.e., meta-features).

Results show that it is possible to accurately predict the performance of a model in a given ML problem, thus leading to the decrease of resources and time spent in re-training unnecessary models.

7.2 Discussion and Limitations

In the last years, ML has seen a tremendous growth in its capabilities and applications. However, significant challenges still remain. These challenges stem mostly from the new characteristics of data: they are in unprecedented volume, and they are generated faster than ever. The resources to handle these data and to train models with them, for example, must thus also be greater. This is especially so in streaming scenarios, in which new data, with potentially different statistical properties, is added frequently, rendering the models potentially outdated.

At the same time, organizations rely increasingly on data and data-based techniques for decision support. While many decision problems are well structured and can be fully automatized, others require human decision capabilities and accountability. In these cases, ML models play a role of decision-support rather than decision-maker. In these cases, especially, it is important that human decision-makers understand how models work and how and why a given prediction is being made by a model, or what is the risk or certainty associated to that decision. This is especially so in domains that affect people's lives, such as in credit concession or in courtrooms. However, as current problems become more complex, so do ML models. Consequently, models and their predictions are also harder to explain.

In this work we addressed these two main problems. One the one hand we propose an approach for minimizing the necessary resources to manage ML models in data streaming scenarios, by predicting the performance metrics of models before they are trained. Using this information, a model will only be re-trained if the statistical properties of the new data indicate that it will lead to a better model.

On the other hand, we propose an approach for generating explanations for the predictions of a model, in the form of a so-called explanatory interface. The key element is a tree algorithm, that is able to generate elements during training that can be used to create intelligible explanations. These explanations are symbolic and domain dependent, based on features, their relevance, and their statistical properties. They are, therefore, easier to be understood by domain experts.

One of the key aspects of both approaches is that they are independent of the domain of application. That is, while we present a case study in the field of tax fraud detection, which is the domain of the problem that inspired this work, the proposed approaches can be applied in any problem. This is because the approach is based on properties of the dataset (the meta-features) and not the dataset itself. And the meta-features can be obtained from any dataset.

Nonetheless, it must also be stated that performance may vary significantly from one problem to the next, namely if the dataset of a new problem has very different properties. Nonetheless, once the new dataset is incorporated into the meta-dataset, it will contribute to improve the meta-model. Thus, the meta-model becomes increasingly better and more generalized as more datasets are added.

The number of datasets used to create the meta-dataset is indeed the main limitation of this work. This means that there is the risk of the meta-dataset not being representative enough to have a good performance on every problem that it is used on. A relatively small number of test datasets was also used. In order to overcome this limitation, the data in each dataset was divided into multiple subsets of increasing size, to allow for the training of more models and thus increasing the size of the dataset. To some extent this is, however, also a limiting factor as these sub-sets share many of the statistical properties of the entire dataset.

In principle, this would be more problematic in streaming scenarios, in which data and its patterns change over time. Nonetheless, we have shown that when the meta-model is updated at regular intervals

with new data, it can adapt and maintain or even improve its performance.

Parallel and distributed processing was not the subject of this work, nor was the capacity of the scalar system with the amount of data that can increase linearly in a given business or area of activity, and in this way, it can easily become an inexplicable model for a new reality. Therefore, the project is not ready to be applied in the constantly evolving real world, but fulfills its purpose in a stable segment. It is also not ready to be put into production, since the API is not scalable and it would be advisable to use the Python Gunicorn web server as it was used in API deployment, or other, but through a proxy server such as Nginx as recommended by the Gunicorn documentation [82] in case it is used for production [83]. In a business scenario, it is unthinkable for the API not to be able to handle all the requests that receives at the same time, or takes too long to respond.

Another limitation of this work, for the meta-learning approach, is that only regression problems were considered. Indeed, regression and classification problems must be implemented differently. Given that this is an early validation of the proposed approach and a first step towards its full implementation, the decision was to start with regression problems. While some binomial classification problems were included, these were transformed into regression as well.

All in all, we believe that this meta-learning based approach can prove interesting towards a more efficient and intelligent management of the processes associated to model training and scoring, especially in scenarios in which there are large volumes of streaming data.

7.3 Future Work

In order to complement the project, one can try to perceive the similarity of the predictions made between XDT and other algorithms not addressed in this project and maybe find a model even more similar to XDT, although apparently it doesn't look like it.

Regarding the developed API, more services could be developed to deal with more file types and data structures. However, the author did not find it necessary because the datasets in Comma-separated values (CSV) format and the data structure in JavaScript Object Notation (JSON) to communication between the API and the other components is the most usual, especially since the theme "Big Data", eXtensible Markup Language (XML) can already be considered obsolete compared to JSON, due to the speed of processing [84].

As future works, already mentioned in this work, the author hopes to be able to follow the same approach of this project and apply it in the Distributed Random Forests model, as it is an identical to Decision Tree and generally presents better results. With this new objective in mind, it is expected to be able to explain even more models considered to be black-box, to explain and convey the information that they cannot, or that the human being cannot. it does not have the capacity to understand, and to transform the information in a way that it is possible to perceive.

The work in the area of XAI, like XDT and others, such as, Neural-Backed Decision Trees (NBDT) [85], which aim for explainability without giving up the performance of the models show promising results. Allied to studies on the ease of implementation of these systems in real context [59], promises to be an area of future study and increasingly being adopted by companies. Likewise, in a real-case scenario, knowing when to re-train a model for every kind of problem shows to be a good issue to be addressed and dealt by the companies that maintain a ML system in its core to make predictions, as it is expected to have the best ML model in any point of time.

Finally, in future work, we will also use this approach while including multiple algorithms rather than Random Forest alone. This allows the use of the meta-model for predicting the best algorithm to use, asides from the expected training time and performance metrics. Also, we propose to validate this approach for classification problems too, in which we expect a similar degree of success.

7.4 Scientific Results

7.4.1 Journal Publications

2. D. Carneiro, M. Guimaraes, M. Carvalho, and P. Novais, "Using meta-learning to predict performance metrics in machine learning problems," EXPERT SYSTEMS. 2021.
DOI https://doi.org/10.1111/exsy.12900
Abstract Machine learning has been facing significant challenges over the last years, much of which stem from the new characteristics of machine learning problems, such as learning from streaming data or incorporating human feedback into existing datasets and models. In these dynamic scenarios, data change over time and models must adapt. However, new data do not necessarily mean new patterns. The main goal of this paper is to devise a method to predict a model's performance metrics before it is trained, in order to decide whether it is worth it to train it or not. That is, will the model hold significantly better results than the current one? To address this issue, we propose the use of meta-learning. Specifically, we evaluate two different meta-models, one built for a specific machine learning problem, and another built based on many different problems, meant to be a generic meta-model, applicable to virtually any problem. In this paper, we focus only on the prediction of the root mean square error (RMSE). Results show that it is possible to accurately predict the RMSE of future models, event in streaming scenarios. Moreover, results also show that it is possible to reduce the need for re-training models between 60% and 98%, depending on the problem and on the threshold used.
Keywords Error Prediction, Interactive Machine Learning, Meta-Learning.

ISI JCR 2021 Impact Factor: 2.812
Q3 – COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE
Q2 – COMPUTER SCIENCE, THEORY & METHODS

SCImago SJR 2021: 0.599
Q2 – ARTIFICIAL INTELLIGENCE
Q2 – COMPUTATIONAL THEORY AND APPLICATIONS
Q2 – CONTROL AND SYSTEMS ENGINEERING
Q2 - THEORETICAL COMPUTER SCIENCE

1. D. Carneiro, M. Guimarães, F. Silva, and P. Novais, "A predictive and user-centric approach to Machine Learning in data streaming scenarios," Neurocomputing. 2022.
DOI https://doi.org/10.1016/j.neucom.2021.07.100
Abstract Machine Learning has emerged in the last years as the main solution to many of nowadays' data-based decision problems. However, while new and more powerful algorithms and the increasing availability of computational resources contributed to a widespread use of Machine Learning, significant challenges still remain. Two of the most significant nowadays are the need to explain a model's predictions, and the significant costs of training and re-training models, especially with large datasets or in streaming scenarios. In this paper we address both issues by proposing an approach we deem predictive and user-centric. It is predictive in the sense that it estimates the benefit of re-training a model with new data, and it is user-centric in the sense that it implements an explainable interface that produces interpretable explanations that accompany predictions. The

former allows to reduce necessary resources (e.g. time, costs) spent on re-training models when no improvements are expected, while the latter allows for human users to have additional information to support decision-making. We validate the proposed approach with a group of public datasets and present a real application scenario.
Keywords Meta-learning, Explainability, Streaming Data, Big Data.

ISI JCR 2021 Impact Factor: 5.779
Q1 – COMPUTER SCIENCE, ARTIFICIAL INTELLIGENCE

SCImago SJR 2021: 1.66
Q1 – COMPUTER SCIENCE APPLICATIONS
Q1 – ARTIFICIAL INTELLIGENCE
Q1 – COGNITIVE NEUROSCIENCE

7.4.2 Conference Publications

9. D. Carneiro, F. Silva, M. Guimarães, D. Sousa, and P. Novais, "Explainable intelligent environments," Advances in Intelligent Systems and Computing, vol. 1239 AISC. pp. 34–43, 2021.
DOI `https://doi.org/10.1007/978-3-030-58356-9_4`
Abstract The main focus of an Intelligent environment, as with other applications of Artificial Intelligence, is generally on the provision of good decisions towards the management of the environment or the support of human decision-making processes. The quality of the system is often measured in terms of accuracy or other performance metrics, calculated on labeled data. Other equally important aspects are usually disregarded, such as the ability to produce an intelligible explanation for the user of the environment. That is, asides from proposing an action, prediction, or decision, the system should also propose an explanation that would allow the user to understand the rationale behind the output. This is becoming increasingly important in a time in which algorithms gain increasing importance in our lives and start to take decisions that significantly impact them. So much so that the EU recently regulated on the issue of a "right to explanation". In this paper we propose a Human-centric intelligent environment that takes into consideration the domain of the problem and the mental model of the Human expert, to provide intelligible explanations that can improve the efficiency and quality of the decision-making processes.
Keywords Intelligent Environments, Explainable AI, Fraud Detection.

8. D. Carneiro, M. Guimarães, and M. Sousa, "Optimizing Instance Selection Strategies in Interactive Machine Learning: An Application to Fraud Detection," Advances in Intelligent Systems and Computing, vol. 1375 AIST. pp. 124–133, 2021.
DOI `https://doi.org/10.1007/978-3-030-73050-5_13`
Abstract Machine Learning systems are generally thought of as fully automatic. However, in recent years, interactive systems in which Human experts actively contribute towards the learning process have shown improved performance when compared to fully automated ones. This may be so in scenarios of Big Data, scenarios in which the input is a data stream, or when there is concept drift. In this paper we present a system for supporting auditors in the task of financial fraud detection. The system is interactive in the sense that the auditors can provide feedback regarding the instances of the data they use, or even suggest new variables. This feedback is incorporated into newly

trained Machine Learning models which improve over time. In this paper we show that the order by which instances are evaluated by the auditors, and their feedback incorporated, influences the evolution of the performance of the system over time. The goal of this paper is to study of different instance selection strategies for Human evaluation and feedback can improve the learning speed. This information can then be used by the system to determine, at each moment, which instances would improve the system the most, so that these can be suggested to the users for validation.
Keywords Interactive Machine Learning, Concept Drift, Fraud Detection.

7. D. Carneiro, M. Guimarães, M. Carvalho, and P. Novais, "Optimizing Model Training in Interactive Learning Scenarios," Trends and Applications in Information Systems and Technologies - Volume 1, WorldCIST 2021, Terceira Island, Azores, Portugal, 30 March - 2 April, 2021., vol. 1365. pp. 156–165, 2021.
DOI https://doi.org/10.1007/978-3-030-72657-7_15
Abstract In the last years, developments in data collection, storing, processing and analysis technologies resulted in an unprecedented use of data by organizations. The volume and variety of data, combined with the velocity at which decisions must now be taken and the dynamism of business environments, pose new challenges to Machine Learning. Namely, algorithms must now deal with streaming data, concept drift, distributed datasets, among others. One common task nowadays is to update or re-train models when data changes, as opposed to traditional one-shot batch systems, in which the model is trained only once. This paper addresses the issue of when to update or re-train a model, by proposing an approach to predict the performance metrics of the model if it were trained at a given moment, with a specific set of data. We validate the proposed approach in an interactive Machine Learning system in the domain of fraud detection.
Keywords Interactive Machine Learning, Meta-Learning, Error Prediction.

6. M. Guimarães and D. Carneiro, "A Meta-Learning Approach to Error Prediction," Iberian Conference on Information Systems and Technologies, CISTI. 2021.
DOI https://doi.org/10.23919/CISTI52073.2021.9476516
Abstract Machine Learning is one of the most trending topics nowadays. The reason is of course for being more and more present in our everyday life, even if we do not notice it. What goes even more unnoticed is the fact that every Machine Learning model needs computational power. And of course, it also needs data. But how many data are necessary to build the best Machine Learning model possible, and how many times do we need to retrain a model so that it does not become obsolete as data change? That kind of questions are the ones that can reduce unnecessary costs to a company. In this paper we propose a novel approach to predict the performance of a model given some characteristics of the data, that are called meta-features. The goal is, indeed, to only train a new model when some error metric (e.g., RMSE) is expected to decrease substantially compared with a previously trained model. This approach is best applied in scenarios of data streaming or in Big Data, as well on Interactive Machine Learning scenarios. We validate it on a real Fraud Detection case and this scenario is also briefly described.
Keywords Interactive Machine Learning, Meta-Learning, Error Prediction, Fraud Detection, Streaming Data, Big Data.
Award Best Paper ESTG Masters 12ª edition.

5. D. Carneiro, M. Guimaraes, J. Baptista, and M. Sousa. “A Conversational Interface for interacting with Machine Learning models.”, 2022.
   URL http://ceur-ws.org/Vol-3168/XAILA2021ICAIL_paper_1.pdf
   Abstract As Machine Learning and other fields of Artificial Intelligence are increasingly used for automating the most diverse aspects of our day-to-day life, so too increases the scrutiny and accountability that these technologies are subject to. Many issues that were previously only attributed to Human decision-makers, such as prejudice or bias, can now also be seen in these automated means. To add up, these technologies are often harder to scrutinize and understand, and can take eventually wrong) decisions at a much faster rate than Humans, thus having a far more potential impact. Trust, transparency, explainability, interpretability and accountability thus become desirable properties of AI systems. In this paper we start with an analysis of some Legal and Ethical considerations regarding the use of AI and related technologies. We then detail an approach whose main goal is to improve the ability of a ML system to explain its decisions, based on a conversational chatbot. The main goal is that the user can interact with and question the model regarding its predictions and, through this, gain an increased confidence on the model, and a better understanding of how it works.
   Keywords Machine Learning, explainable AI, chatbot.

4. Carneiro, D., Sousa, M., Palumbo, G., Guimarães, M., Carvalho, M., and Novais, P. “Continuously Learning from User Feedback.” World Conference on Information Systems and Technologies. Springer, Cham, 2022.
   URL https://link.springer.com/chapter/10.1007/978-3-031-04826-5_57
   Abstract Machine Learning has been evolving rapidly over the past years, with new algorithms and approaches being devised to solve the challenges that the new properties of data pose. Specifically, algorithms must now learn continuously and in real time, from very large and possibly distributed sets of data. In this paper we describe a learning system that tackles some of these novel challenges. It learns and adapts in real-time by continuously incorporating user feedback, in a fully autonomous way. Moreover, it allows for users to manage features (e.g. add, edit, remove), reflecting these changes on-the-fly in the Machine Learning pipeline. The paper describes some of the main functionalities of the system, which despite being of general-purpose, is being developed in the context of a project in the domain of financial fraud detection.
   Keywords Active machine learning, Online machine learning, Meta-learning, Fraud detection.

3. Rosa, L., Guimaraes, M., Carneiro, D., Silva, F., and Analide, C., “Explainable Decision Tree on Smart Human Mobility”, 2022.
   URL https://scholar.archive.org/work/x7ellcvq6fhrxofcu7rq2vytui/access/wayback/https://ebooks.iospress.nl/pdf/doi/10.3233/AISE220059
   Abstract Artificial Intelligence is a hot topic and Machine Learning is one of the most fluent approaches and practices. The problem with many AI models is that they can be useful for predicting but they are bad at explaining why they behave a certain way. In some contexts, the explanation may even be more important than the prediction itself, mainly in systems in which decisions are made based on their predictions. Therefore, it is increasingly necessary to provide

a forecast accompanied by an explanation, when decisions are made automatically. This paper aims to contribute to the solution of problem based on human mobility research, or at least, to be a starting point for its solution.

Keywords Explainable Artificial Intelligence, Machine Learning, Smart Cities, Smart Human Mobility.

2. A. Borges, M. Carvalho, M. Maia, M. Guimarães and D. Carneiro, "Predicting and Explaining Absenteeism Risk in Hospital Patients Before and During COVID-19", not published yet, 2022.

Abstract In order to address one of the most challenging problems in hospital management - patients absenteeism without prior notice - this study analyses the risk factors associated with this event. To this end, through real data from a hospital located in the North of Portugal, a prediction model previously validated in the literature is used to infer on absenteeism risk factors, and an explainable model is proposed, based on a modified CART algorithm. The latter intends to generate human-interpretable explanation for patient absenteeism, and its implementation is described in detail. Furthermore, given the significant impact the COVID-19 pandemic had on hospitals management, a comparison between patients profile upon absenteeism before and during COVID-19 pandemic situation is performed. Results obtained differ between hospital specialty and time periods meaning that patient profile on absenteeism changes during pandemic periods and within specialty.

Keywords Patients absenteeism, risk factors, logistic model, explainable model, CART algorithm, COVID-19.

1. G. Palumbo, M. Guimarães, D. Carneiro, P. Novais and V. Alves, "Real-time algorithm recommendation using meta-learning", not published yet, 2022.

Abstract As the field of Machine Learning evolves, the number of available learning algorithms and their parameters continues to grow. On the one hand, this is positive as it allows for the finding of potentially more accurate models. On the other hand, however, it also makes the process of finding the right model more complex, given the number of possible configurations. Traditionally, data scientists rely on trial-and-error or brute force procedures, which are costly, or on their own intuition or expertise, which is hard to acquire. In this paper we propose an approach for algorithm recommendation based on meta-learning. The approach can be used in real-time to predict the best n algorithms (based on a selected performance metric) and their configuration, for a given ML problem. We evaluate it through cross-validation, and by comparing it against an Auto ML approach, in terms of accuracy and time. Results show that the proposed approach recommends algorithms that are similar to those of traditional approaches, in terms of performance, in just a fraction of the time.

Keywords Machine Learning, Meta-Learning, Algorithm Recommendation.

Award Electronics Journal MDPI Best Paper.

Acknowledgments

This work was supported by the Northern Regional Operational Program, Portugal 2020 and European Union, through European Regional Development Fund (ERDF) in the scope of project number 39900 - 31/SI/2017, and by FCT - Fundação para a Ciência e a Tecnologia, through project UIDB/04728/2020.

Bibliography

[1] Filip Karlo Dosilovic, Mario Brcic, and Nikica Hlupic. Explainable artificial intelligence: A survey. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2018 - Proceedings*, (May):210–215, 2018.

[2] Dimitrios Ververidis, Constantine Kotropoulos, and Ioannis Pitas. Automatic emotional speech classification. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–593. IEEE, 2004.

[3] Kate Crawford. Artificial intelligence's white guy problem. *The New York Times*, 25, 2016.

[4] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, 2019.

[5] Eva García-Martín, Niklas Lavesson, Håkan Grahn, Emiliano Casalicchio, and Veselka Boeva. How to measure energy consumption in machine learning algorithms. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 243–255. Springer, 2018.

[6] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016.

[7] Randy Goebel, Ajay Chander, Katharina Holzinger, Freddy Lecue, Zeynep Akata, Simone Stumpf, Peter Kieseberg, and Andreas Holzinger. Explainable ai: the new 42? In *International cross-domain conference for machine learning and knowledge extraction*, pages 295–303. Springer, 2018.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[9] Ribana Roscher, Bastian Bohn, Marco F. Duarte, and Jochen Garcke. Explainable Machine Learning for Scientific Insights and Discoveries. *IEEE Access*, 8:42200–42216, 2020.

[10] Kacper Sokol and Peter Flach. Desiderata for interpretability: Explaining decision tree predictions with counterfactuals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10035–10036, 2019.

[11] Berk Ustun, Alexander Spangher, and Yang Liu. Actionable recourse in linear classification. In *FAT* 2019 - Proceedings of the 2019 Conference on Fairness, Accountability, and Transparency*, pages 10–19, 2019.

[12] Fabio Silva and Cesar Analide. Information asset analysis: credit scoring and credit suggestion. *International Journal of Electronic Business*, 9(3):203, 2011.

[13] Sandhya Saisubramanian, Sainyam Galhotra, and Shlomo Zilberstein. Balancing the tradeoff between clustering value and interpretability. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 351–357, 2020.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?" Explaining the predictions of any classifier. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-August-2016:1135–1144, 2016.

[15] Scott M Lundberg and Su In Lee. A unified approach to interpreting model predictions, 2017.

[16] Maonan Wang, Kangfeng Zheng, Yanqing Yang, and Xiujuan Wang. An explainable machine learning framework for intrusion detection systems. *IEEE Access*, 8:73127–73141, 2020.

[17] Liat Antwarg, Bracha Shapira, and Lior Rokach. Explaining Anomalies Detected by Autoencoders Using SHAP. *arXiv*, pages 1–37, 2019.

[18] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *34th International Conference on Machine Learning, ICML 2017*, 7:4844–4866, 2017.

[19] Bach S., Binder A., Montavon G., Klauschen F., Müller K-R., and Samek W. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLoS ONE*, 10(7), 2015.

[20] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M.F. Moura, and Peter Eckersley. Explainable machine learning in deployment. *FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020.

[21] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*, pages 80–89, 2019.

[22] Anand Rajaraman. More data usually beats better algorithms. *Datawocky Blog*, 2008.

[23] Omar Tawakol. More Data Beats Better Algorithms—Or Does It? *All Things D, September*, 7, 2012.

[24] Julian J. Faraway and Nicole H. Augustin. When small data beats big data. *Statistics & Probability Letters*, 136:142–145, 2018.

[25] Full batch, mini-batch, and online learning. `https://kaggle.com/residentmario/full-batch-mini-batch-and-online-learning`. Accessed: 2022-03-28.

[26] Liu Jiang, Shixia Liu, and Changjian Chen. Recent research advances on interactive machine learning. *Journal of Visualization*, 22(2):401–417, 2019. Publisher: Springer.

[27] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4):964–994, 2016. Publisher: Springer.

[28] Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

[29] Burr Settles. From theories to queries: Active learning in practice. In *Active Learning and Experimental Design workshop In conjunction with AISTATS 2010*, pages 1–18, 2011.

[30] David Tripp. Action research: a methodological introduction. *Educação e pesquisa*, 31(3):443–466, 2005.

[31] Stephen Kemmis and Robin McTaggart. Participatory Action Research: Communicative Action and the Public Sphere. In *The Sage handbook of qualitative research, 3rd ed.*, pages 559–603. Sage Publications Ltd, Thousand Oaks, CA, 2005.

[32] Pedro Domingos. *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015.

[33] Susanne Still. Information-theoretic approach to interactive learning. *EPL (Europhysics Letters)*, 85(2):28005, 2009.

[34] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[35] Andreas Holzinger, Markus Plass, Michael Kickmeier-Rust, Katharina Holzinger, Gloria Cerasela Crişan, Camelia-M Pintea, and Vasile Palade. Interactive machine learning: experimental evidence for the human in the algorithmic loop. *Applied Intelligence*, 49(7):2401–2414, 2019.

[36] Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H Witten. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.

[37] Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[38] Stuart Berg, Dominik Kutra, Thorben Kroeger, Christoph N Straehle, Bernhard X Kausler, Carsten Haubold, Martin Schiegg, Janez Ales, Thorsten Beier, Markus Rudy, et al. Ilastik: interactive machine learning for (bio) image analysis. *Nature Methods*, 16(12):1226–1232, 2019.

[39] Andreas Holzinger and Igor Jurisica. Knowledge discovery and data mining in biomedical informatics: The future is in integrative, interactive machine learning solutions. In *Interactive knowledge discovery and data mining in biomedical informatics*, pages 1–18. Springer, 2014.

[40] Bhavya Ghai, Q. Vera Liao, Yunfeng Zhang, Rachel Bellamy, and Klaus Mueller. Explainable Active Learning (XAL): An Empirical Study of How Local Explanations Impact Annotator Experience. *arXiv:2001.09219 [cs]*, September 2020. arXiv: 2001.09219.

[41] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. Power to the people: The role of humans in interactive machine learning. *Ai Magazine*, 35(4):105–120, 2014.

[42] Sik-Ho Tsang. Review: Suggestive Annotation — Deep Active Learning Framework (Biomedical Image Segmentation). `https://towardsdatascience.com/review-suggestive-annotation-deep-active-learning-framework-biomedical-image-segmentation-e08e4b931ea6`, April 2019. Accessed: 2022-03-17.

[43] Qiang Yang, Yu Zhang, Wenyuan Dai, and Sinno Jialin Pan. *Transfer learning*. Cambridge University Press, 2020.

[44] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *arXiv preprint arXiv:2009.00236*, 2020.

[45] Dmitri E Kvasov and Marat S Mukhametzhanov. Metaheuristic vs. deterministic global optimization algorithms: The univariate case. *Applied Mathematics and Computation*, 318:245–259, 2018.

[46] Arjun Chandra and Xin Yao. Evolving hybrid ensembles of learning machines for better generalisation. *Neurocomputing*, 69(7-9):686–700, 2006.

[47] Hassan A Bashir and Richard S Neville. Hybrid evolutionary computation for continuous optimization. *arXiv preprint arXiv:1303.3469*, 2013.

[48] Joaquin Vanschoren. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548*, 2018.

[49] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002. Publisher: Springer.

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

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

[52] Stephen R Gardner. Building the data warehouse. *Communications of the ACM*, 41(9):52–60, 1998.

[53] Thomas Vetterli, Anca Vaduva, and Martin Staudt. Metadata standards for data warehousing: open information model vs. common warehouse metadata. *ACM Sigmod Record*, 29(3):68–75, 2000.

[54] The NSA and Big Data: The Power and Peril of Metadata. `https://solutionsreview.com/business-intelligence/the-nsa-and-big-data-the-power-and-peril-of-metadata/`, June 2013. Accessed: 2022-03-24.

[55] Katie Atkinson, Trevor Bench-Capon, and Danushka Bollegala. Explanation in AI and law: Past, present and future. *Artificial Intelligence*, 289:103387, 2020. Publisher: Elsevier.

[56] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, and Richard Benjamins. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58:82–115, 2020. Publisher: Elsevier.

[57] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using SHAP. *arXiv preprint arXiv:1903.02407*, 2019.

[58] Bryce Goodman and Seth Flaxman. European Union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3):50–57, 2017.

[59] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. Explainable Machine Learning in Deployment. *arXiv:1909.06342 [cs, stat]*, July 2020. arXiv: 1909.06342 version: 3.

[60] Lior Rokach and Oded Maimon. Decision trees. In *Data mining and knowledge discovery handbook*, pages 165–192. Springer, 2005.

[61] John M Chambers. Linear models. In *Statistical models in S*, pages 95–144. Routledge, 2017.

[62] Janet L Kolodner. An introduction to case-based reasoning. *Artificial intelligence review*, 6(1):3–34, 1992.

[63] Alex John London. Artificial intelligence and black-box medical decisions: accuracy versus explainability. *Hastings Center Report*, 49(1):15–21, 2019.

[64] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[65] Ray Wright. Interpreting black-box machine learning models using partial dependence and individual conditional expectation plots. *Exploring SAS® Enterprise Miner Special Collection*, pages 1950–2018.

[66] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on artificial intelligence and statistics*, pages 2907–2916. PMLR, 2020.

[67] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. Generating hierarchical explanations on text classification via feature interaction detection. *arXiv preprint arXiv:2004.02015*, 2020.

[68] David Alvarez-Melis and Tommi S Jaakkola. On the robustness of interpretability methods. *arXiv preprint arXiv:1806.08049*, 2018.

[69] T Nathan Mundhenk, Barry Y Chen, and Gerald Friedland. Efficient saliency maps for explainable ai. *arXiv preprint arXiv:1911.11293*, 2019.

[70] Daniel Z Levin, Rob Cross, and Lisa C Abrams. Why should i trust you? predictors of interpersonal trust in a knowledge transfer context. In *Academy of Management Meeting, Denver, CO*, volume 2002, 2002.

[71] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[72] Neurat project. `https://neurat.pt/`. Accessed: 2022-01-15.

[73] Andreas Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016. ISBN: 2198-4018 Publisher: Springer.

[74] Mariusz Macudziński et al. The standard audit file for tax (saf-t): An it tool for tax system tightening. *Studia Administracji i Bezpieczeństwa*, 1(5):108–120, 2018.

[75] Sonia Singh and Priyanka Gupta. Comparative study id3, cart and c4. 5 decision tree algorithm: a survey. *International Journal of Advanced Information Science and Technology (IJAIST)*, 27(27):97–103, 2014.

[76] Robert I Lerman and Shlomo Yitzhaki. A note on the calculation and interpretation of the gini index. *Economics Letters*, 15(3-4):363–368, 1984.

[77] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2019.

[78] Edesio Alcobaça, Felipe Siqueira, Adriano Rivolli, Luís PF Garcia, Jefferson T Oliva, André CPLF de Carvalho, et al. Mfe: Towards reproducible meta-feature extraction. *Journal of Machine Learning Research*, 21(111):1–5, 2020.

[79] Meta-feature Description Table — pymfe 0.4 documentation. `https://pymfe.readthedocs.io/en/latest/auto_pages/meta_features_description.html`. Accessed: 2022-02-17.

[80] API Documentation — pymfe 0.4.1 documentation. `https://pymfe.readthedocs.io/en/latest/api.html`. Accessed: 2022-03-28.

[81] Spencer Aiello, Cliff Click, Hank Roark, Ludi Rehak, and Pasha Stetsenko. Machine learning with python and H2O. *H2O. ai Inc*, 2016.

[82] Deploying Gunicorn — Gunicorn 20.1.0 documentation. `https://docs.gunicorn.org/en/stable/deploy.html`. Accessed: 2022-03-23.

[83] Tasnuva Zaman. Deploy flask app with Nginx using Gunicorn. `https://faun.pub/deploy-flask-app-with-nginx-using-gunicorn-7fda4f50066a`, September 2020. Accessed: 2022-03-24.

[84] Json (JavaScript Object Notation) - O sucessor do XML. `https://pplware.sapo.pt/tutoriais/json-javascript-objection-notation-o-sucessor-do-xml/`, February 2011. Accessed: 2022-03-24.

[85] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Henry Jin, Suzanne Petryk, Sarah Adel Bargal, and Joseph E. Gonzalez. Nbdt: Neural-backed decision trees, 2020.

Appendix A

Datasets

Table A.1: Characterization of the datasets used for training the meta-model

Dataset	Source	Type	Lines	Cols.
Credit Fraud Detection	https://www.kaggle.com/mlg-ulb/creditcardfraud	B	30000	31
German Credit Card	https://www.kaggle.com/agsam23/german-credit?select=german_credit_data.csv	B	1000	21
Bank Marketing	http://archive.ics.uci.edu/ml/datasets/Bank+Marketing	B	30000	21
House Price	https://kaggle.com/harlfoxem/housesalesprediction	R	4600	19
House prediction	https://www.kaggle.com/abhisheikreddy646/house-prediction-for-zipcode?select=80111.csv	R	67	6
California house prediction	https://www.kaggle.com/camnugent/california-housing-prices	R	546	12
Boston house prediction	https://www.kaggle.com/vikrishnan/boston-house-prices	R	506	14
Real estate	https://www.kaggle.com/quantbruce/real-estate-price-prediction	R	414	8
Concrete Data	https://www.kaggle.com/maajdl/yeh-concret-data	R	1030	9
Suicides per 100k	https://www.kaggle.com/russellyates88/suicide-rates-overview-1985-to-2016	R	10189	10
Heart Disease	https://www.kaggle.com/amanajmera1/framingham-heart-study-dataset	B	4238	16
HR Attrition	https://www.kaggle.com/vjchoudhary7/hr-analytics-case-study?select=general_data.csv	B	4410	22
Fictional HR Attrition	https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset	R	1470	33
HR Salary	https://www.kaggle.com/rhuebner/human-resources-data-set	R	311	30
School grades	https://www.mldata.io/dataset-details/school_grades/	R	649	33
Cardiovascular diseases	https://kaggle.com/aiaiaidavid/cardio-data-dv13032020	B	10000	12
Killed or Seriously Injured	https://kaggle.com/jrmistry/killed-or-seriously-injured-ksi-toronto-clean	B	12557	56
Contains Aditives	https://kaggle.com/jadeblue/openfoodfactsclean	B	774	13
Starbucks proteins	https://kaggle.com/jadeblue/openfoodfactsclean	R	243	7
McDonalds proteins	https://kaggle.com/jadeblue/openfoodfactsclean	R	260	6
Medical Cost	https://kaggle.com/mirichoi0218/insurance	R	1338	7
Car Price Prediction	https://kaggle.com/hellbuoy/car-price-prediction	R	205	24
Social Network Ads	https://kaggle.com/dragonheir/logistic-regression	B	400	4
Abalone	https://www.mldata.io/dataset-details/abalone/	R	4177	9
Auto mpg	https://www.mldata.io/dataset-details/auto_mpg/	R	398	9
Exercise Calories	https://kaggle.com/fmendes/fmendesdat263xdemos	R	9000	8
Fraud Detection	Proprietary	R	22225	13
Computer Hardware	https://www.mldata.io/dataset-details/computer_hardware/	R	209	10
Forbes Billionaires	https://www.mldata.io/dataset-details/forbes_billionaire/	R	2043	6
Meta data error	https://archive.ics.uci.edu/ml/machine-learning-databases/meta-data/	R	528	22
Drug spend	https://datahub.io/core/pharmaceutical-drug-spending	R	1036	6
Servo	https://archive.ics.uci.edu/ml/machine-learning-databases/servo/	R	167	5
TV Shows	https://www.kaggle.com/javagarm/tv-shows-on-ott-platforms	R	5602	8
Predict number of suicides	https://www.kaggle.com/szamil/who-suicide-statistics	R	43776	6
World Happiness 2015	https://www.kaggle.com/mathurinache/world-happiness-report	R	158	10
World Happiness 2016	https://www.kaggle.com/mathurinache/world-happiness-report	R	157	11
World Happiness 2017	https://www.kaggle.com/mathurinache/world-happiness-report	R	155	10
World Happiness 2018	https://www.kaggle.com/mathurinache/world-happiness-report	R	156	8
World Happiness 2019	https://www.kaggle.com/mathurinache/world-happiness-report	R	156	7
Happiness up to 2017	https://worldhappiness.report/ed/2017/	R	1420	16
Happiness up to 2020	https://worldhappiness.report/ed/2020/	R	1848	19
Diagnostic breast cancer	https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/	B	569	32
Prognostic breast cancer	https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/	B	198	34
Wine quality (red)	https://kaggle.com/uciml/red-wine-quality-cortez-et-al-2009	R	1599	12

Table A.2: Characterization of the datasets used for the validation of the proposed approach. The first 3 are for batch scenarios, while the last 3 are for streaming data scenarios). (R: Regression, B: Binary Classification)

Dataset	Source	Type	Lines	Cols.
Wine quality (white)	https://data.world/uci/wine-quality/workspace/data-dictionary	R	4500	12
Diabetes	https://kaggle.com/kandij/diabetes-dataset	B	768	9
Breast Cancer	https://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/	B	699	10
Airlines	https://moa.cms.waikato.ac.nz/datasets/	B	10000	8
AWS Prices	https://moa.cms.waikato.ac.nz/datasets/	R	10000	8
Electricity	https://moa.cms.waikato.ac.nz/datasets/	B	10000	9

Appendix B

XDT API Endpoints

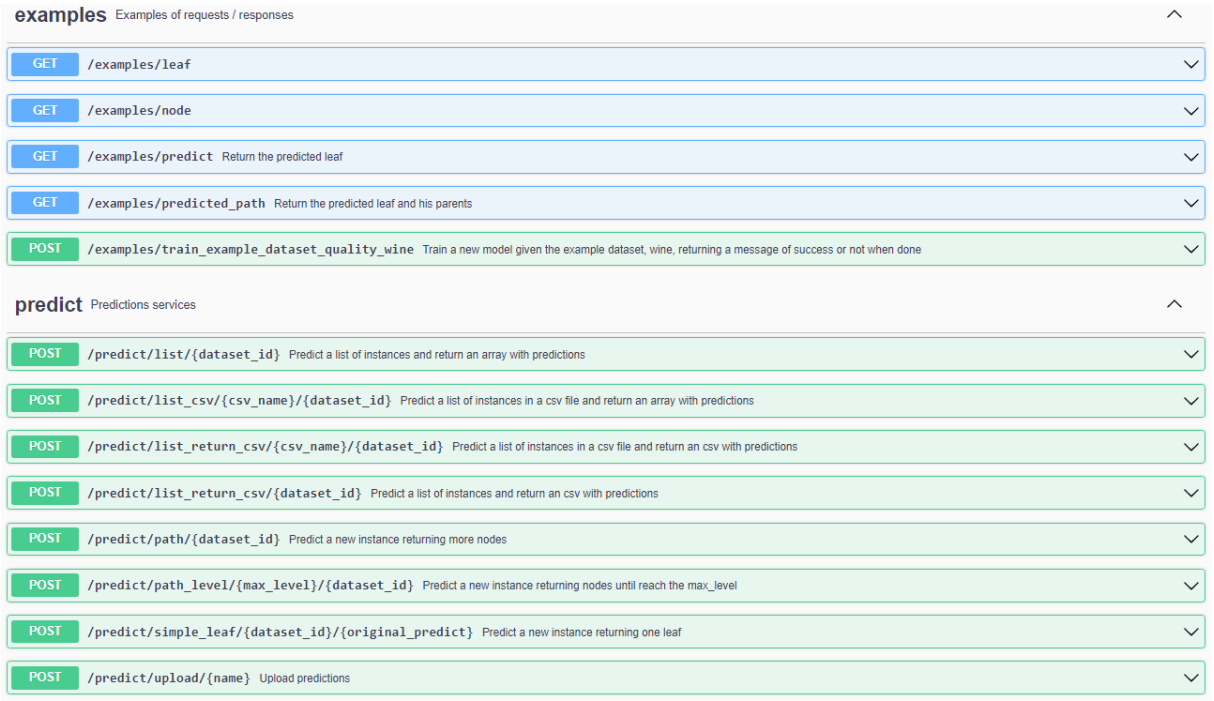

Figure B.1: XDT API endpoints of examples and predict services

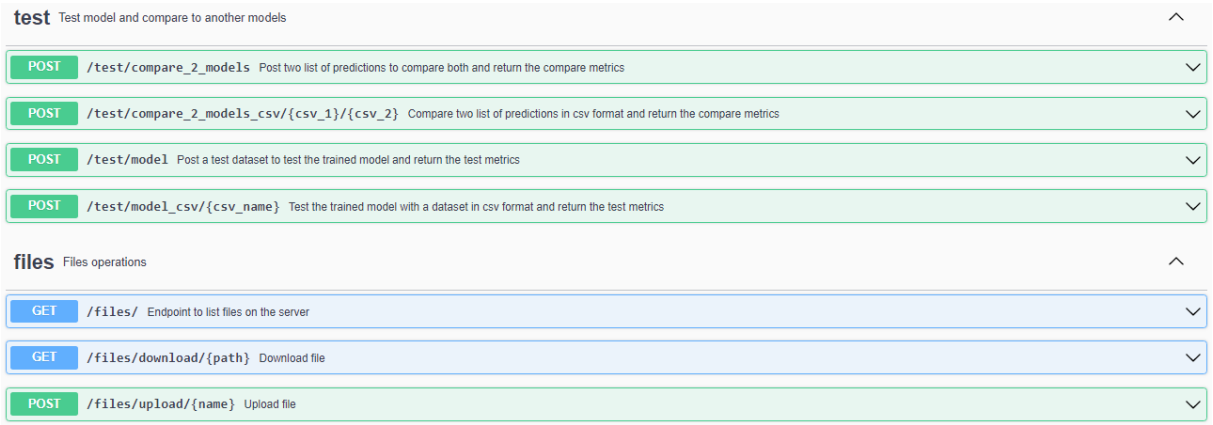

Figure B.2: XDT API endpoints of test and files services

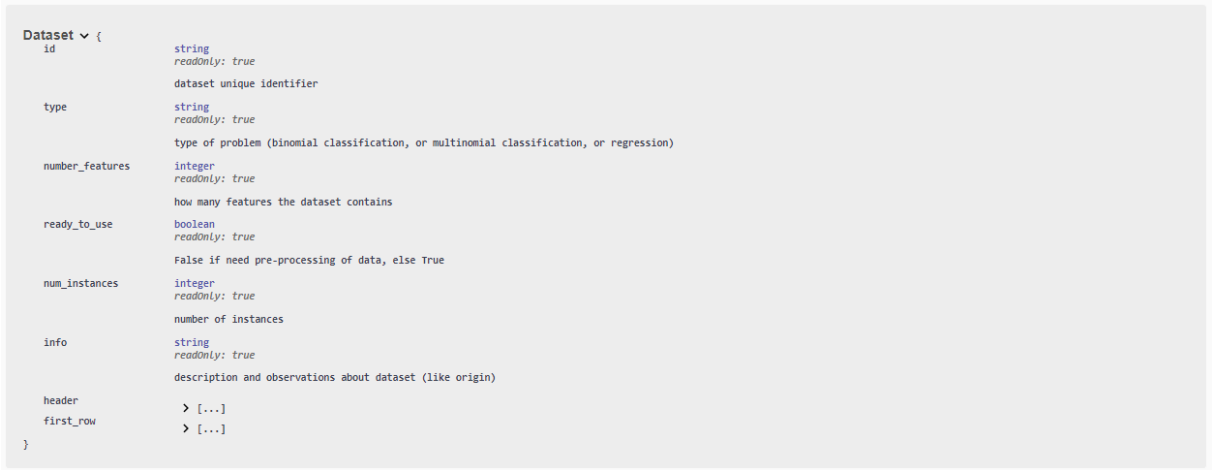

Figure B.3: XDT API dataset model

Figure B.4: XDT API configurations model

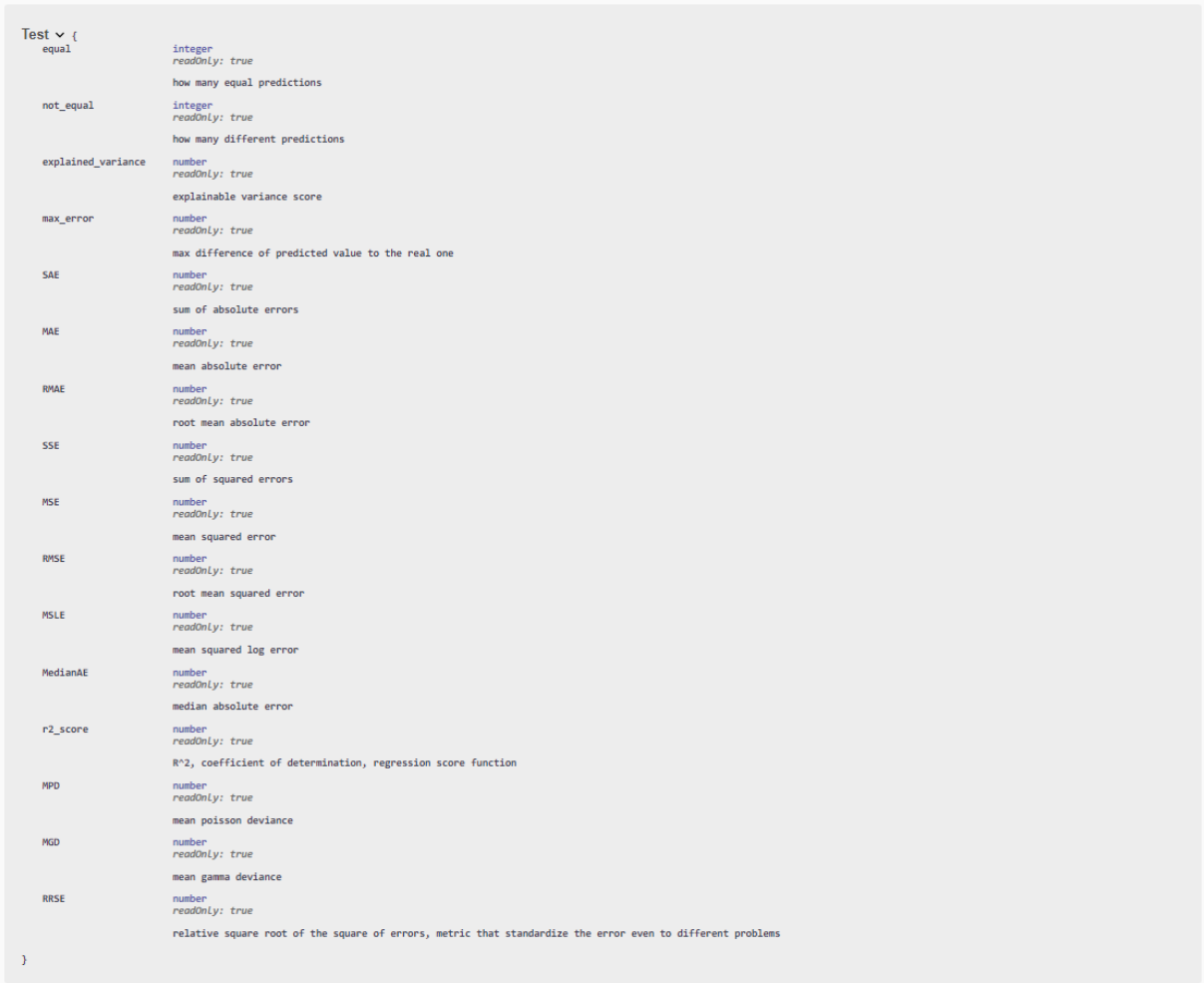

Figure B.5: XDT API test model

Appendix C

Example of a Decision Tree


```
{
  "id": 1,
  "type": "node",
  "index": 1,
  "var_of_division": "outlook",
  "type_of_var": "string",
  "value_string": "overcast",
  "value_number": 0,
  "min_value": 0,
  "max_value": 0,
  "mean_value": 0,
  "IQR": 0,
  "outlier_limit_low": 0,
  "outlier_limit_high": 0,
  "outliers": 0,
  "outliers_ids_low": 0,
  "outliers_ids_high": 0,
  "dispersion": 0.17857142857142858,
  "predict": 0.6428571428571429,
  "predict_message": "Not Fraud with 84% of probability.",
  "var": 0.22959183673469388,
  "std": 0.47915742374995496,
  "num_instances": 14,
  "instances": [0, 4, 7, 11, 13],
  "left": {
    "id": 2,
    "type": "leaf",
    "min_value": 1,
    "max_value": 1,
    "mean_value": 1,
```

```
    "IQR": 0,
    "outlier_limit_low": 1,
    "outlier_limit_high": 1,
    "outliers": 0,
    "outliers_ids_low": [],
    "outliers_ids_high": [],
    "predict": 1,
    "predict_message": "Not Fraud with 80% of probability.",
    "var": 0,
    "std": 0,
    "num_instances": 4,
    "instances": [0, 1, 2, 3]
  },
  "right": {
    "id": 3,
    "type": "node",
    "index": 4,
    "var_of_division": "humidity",
    "type_of_var": "number",
    "value_string": "",
    "value_number": 85,
    "min_value": 70,
    "max_value": 96,
    "mean_value": 82.7,
    "IQR": 15.5,
    "outlier_limit_low": 51.75,
    "outlier_limit_high": 113.75,
    "outliers": 0,
    "outliers_ids_low": [],
    "outliers_ids_high": [],
    "dispersion": 0.16000000000000003,
    "predict": 0.5,
    "predict_message": "Not Fraud with 85% of probability.",
    "var": 0.25,
    "std": 0.5,
    "num_instances": 10,
    "instances": [6, 12, 5, 8, 4],
    "left": {
      "id": 4,
      "type": "leaf",
      "min_value": 0,
      "max_value": 1,
```

```
      "mean_value": 0.8,
      "IQR": 0,
      "outlier_limit_low": 1,
      "outlier_limit_high": 1,
      "outliers": 1,
      "outliers_ids_low": [6],
      "outliers_ids_high": [],
      "predict": 0.8,
      "predict_message": "Not Fraud with 82% of probability.",
      "var": 0.16,
      "std": 0.4,
      "num_instances": 5,
      "instances": [6, 5, 7, 12, 13]
    },
    "right": {
      "id": 5,
      "type": "leaf",
      "min_value": 0,
      "max_value": 1,
      "mean_value": 0.2,
      "IQR": 0,
      "outlier_limit_low": 0,
      "outlier_limit_high": 0,
      "outliers": 1,
      "outliers_ids_low": [],
      "outliers_ids_high": [4],
      "predict": 0.2,
      "predict_message": "Not Fraud with 88% of probability.",
      "var": 0.16000000000000003,
      "std": 0.4,
      "num_instances": 5,
      "instances": [8, 9, 10, 11, 4]
    }
  }
}
```

Appendix D

Example of an Explanation


```
{
  "leaf": {
    "id": 24,
    "type": "leaf",
    "min_value": 6,
    "max_value": 7,
    "mean_value": 6.76,
    "IQR": 0,
    "outlier_limit_low": 7,
    "outlier_limit_high": 7,
    "outliers": 4,
    "outliers_ids_low": [
      0,
      69,
      134,
      307
    ],
    "outliers_ids_high": [],
    "predict": 6.76,
    "predict_message": "6.76 more or less 0.42",
    "var": 0.17993079584775087,
    "std": 0.42418250299576343,
    "num_instances": 17,
    "instances": [
      0,
      131,
      146,
      154,
      155
    ]
```

```
  },
  "explanation": [
    {
      "variable": "alcohol",
      "direction": "below",
      "value": 11.6
    },
    {
      "variable": "volatile acidity",
      "direction": "below",
      "value": 0.3
    },
    {
      "variable": "free sulfur dioxide",
      "direction": "below",
      "value": 75
    },
    {
      "variable": "free sulfur dioxide",
      "direction": "above",
      "value": 17
    },
    {
      "variable": "pH",
      "direction": "below",
      "value": 3.66
    },
    {
      "variable": "sulphates",
      "direction": "equal",
      "value": 0.55
    },
    {
      "variable": "sulphates",
      "direction": "below",
      "value": 0.62
    },
    {
      "variable": "citric acid",
      "direction": "above",
      "value": 0.28
    }
```

```
],
"counterfactual_analysis": [
  {
    "variable": "alcohol",
    "direction": "above",
    "value": 0.6,
    "new_prediction": "5.86",
    "prediction_message": "5.85 more or less 0.34",
    "var": 0.12244897959183673,
    "std": 0.3499271061118826,
    "num_instances": 7,
    "id": 85
  },
  {
    "variable": "volatile acidity",
    "direction": "above",
    "value": 0.03,
    "new_prediction": "5.83",
    "prediction_message": "5.83 more or less 0.37",
    "var": 0.1388888888888889,
    "std": 0.37267799624996495,
    "num_instances": 6,
    "id": 62
  },
  {
    "variable": "pH",
    "direction": "above",
    "value": 0.46,
    "new_prediction": "5.00",
    "prediction_message": "5.0 more or less 0.0",
    "var": 0,
    "std": 0,
    "num_instances": 3,
    "id": 28
  },
  {
    "variable": "sulphates",
    "direction": "below",
    "value": 0.1,
    "new_prediction": "5.90",
    "prediction_message": "5.90 more or less 0.42",
    "var": 0.18140589569160998,
```

```
      "std": 0.4259177099999599,
      "num_instances": 84,
      "id": 16
    },
    {
      "variable": "sulphates",
      "direction": "above",
      "value": 0.07,
      "new_prediction": "6.00",
      "prediction_message": "6.0 more or less 0.0",
      "var": 0,
      "std": 0,
      "num_instances": 10,
      "id": 27
    }
  ],
  "human_conterfactual_explanation": "'alcohol' is below 11.6 units.
  'volatile acidity' is below 0.3 units. 'free sulfur dioxide' is
  below 75.0 units. 'free sulfur dioxide' is above 17.0 units. 'pH'
  is below 3.66 units. 'sulphates' is 0.55. 'sulphates' is below
  0.62 units. 'citric acid' is above 0.28 units.  However: If
  'alcohol' was above 0.6 units, the prediction would be 5.86. If
  'volatile acidity' was above 0.03 units, the prediction would be
  5.83. If 'pH' was above 0.46 units, the prediction would be 5.00.
  If 'sulphates' was below 0.1 units, the prediction would be 5.90.
  If 'sulphates' was above 0.07 units, the prediction would be 6.00."
}
```

Appendix E

XDT Frontend

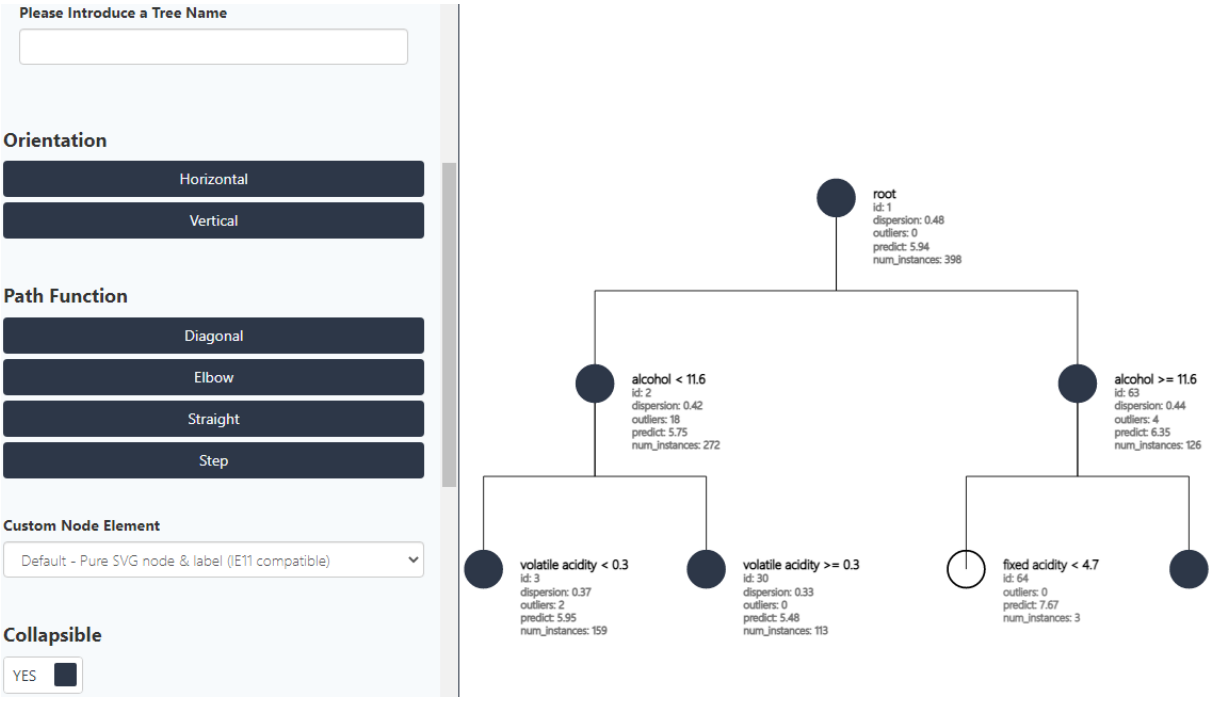

Figure E.1: XDT Frontend with some visualization options in left side panel

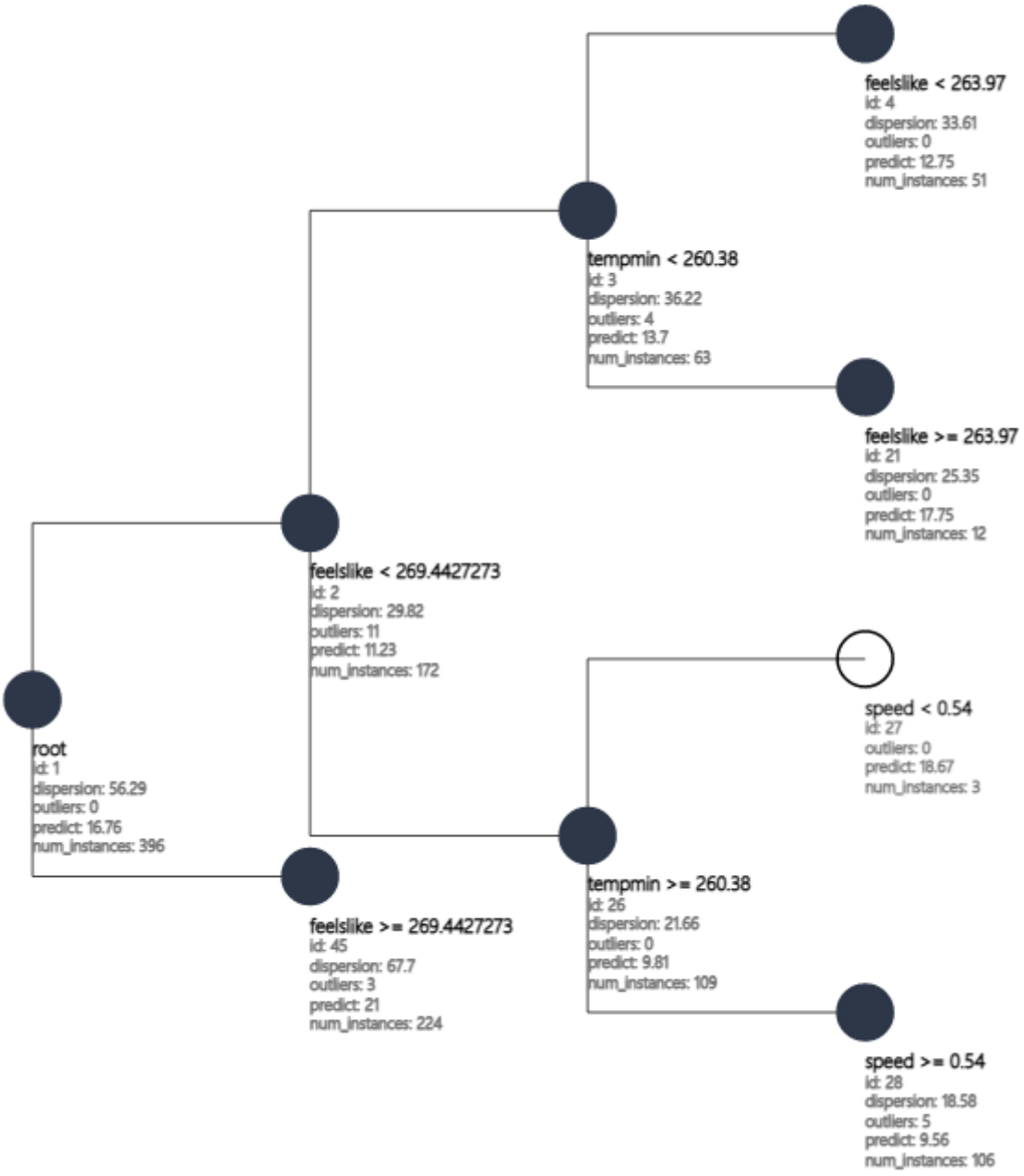

Figure E.2: XDT frontend with horizontal orientation

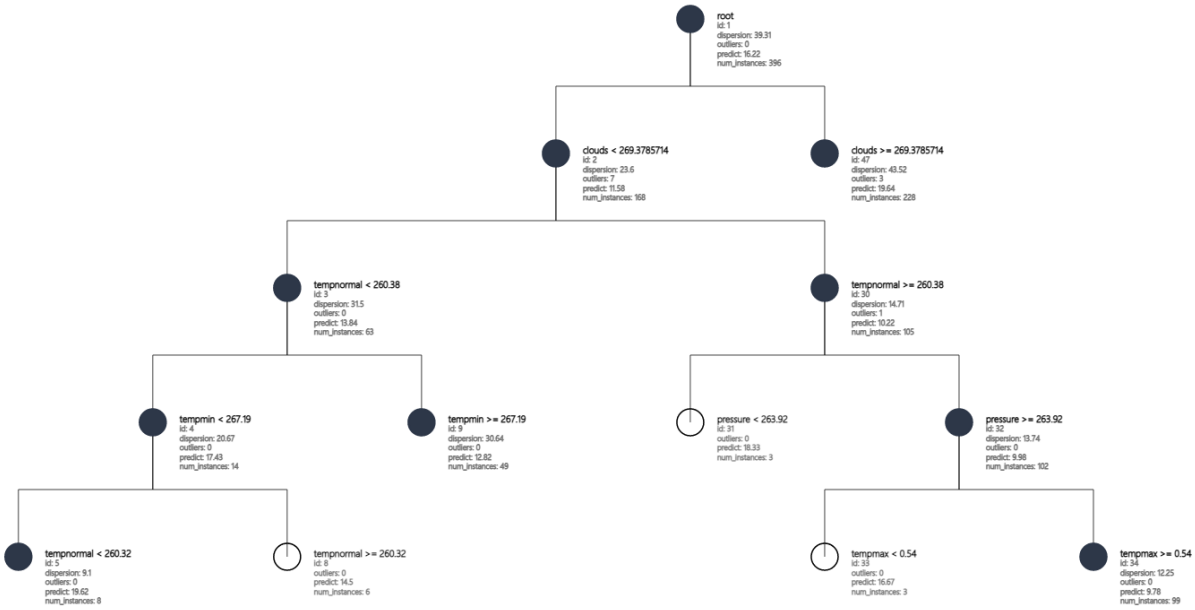

Figure E.3: XDT frontend with vertical orientation

Appendix F

Requirements

```
aniso8601==9.0.1
attrs==21.2.0
click==8.0.3
colorama==0.4.4
csvalidate==1.1.1
Flask==2.0.2
Flask-Cors==3.0.10
Flask-CSV==1.2.0
flask-restx==0.5.1
greenlet==1.1.2
itsdangerous==2.0.1
Jinja2==3.0.3
joblib==1.1.0
jsonschema==4.3.2
MarkupSafe==2.0.1
marshmallow==3.14.1
numpy==1.21.5
pandas==1.3.5
prettytable==2.5.0
pyrsistent==0.18.0
python-dateutil==2.8.2
pytz==2021.3
scikit-learn==1.0.1
scipy==1.7.3
six==1.16.0
sklearn==0.0
SQLAlchemy==1.4.28
threadpoolctl==3.0.0
wcwidth==0.2.5
Werkzeug==2.0.2
```