SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

M MASTER
COMPUTER ENGINEERING

# Interactive Learning in Decision Support

Miguel Ângelo da Silva e Sousa

10/2022

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

Miguel Ângelo da Silva e Sousa. Interactive Learning in Decision Support.
10/2022

SCHOOL OF
MANAGEMENT
AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

Interactive Learning in Decision Support
Miguel Ângelo da Silva e Sousa

M MASTER
COMPUTER ENGINEERING

SCHOOL OF
MANAGEMENT AND
TECHNOLOGY
POLYTECHNIC OF
PORTO

M

MASTER
COMPUTER ENGINEERING

# Interactive Learning in Decision Support

Miguel Ângelo da Silva e Sousa

Davide Rua Carneiro

## Agradecimentos

Ao professor Davide Carneiro, por me guiar durante todo este projeto, sempre com total disponibilidade e companheirismo.

A todos os meus amigos, pelo apoio e motivação transmitidos.

À minha família, em especial ao meu pai, mãe e irmã, sem os quais nada disto seria possível.

## Resumo

De acordo com o dicionário *priberam* da língua portuguesa, o conceito de Fraude pode ser definido como uma "ação ilícita, punível por lei, que procura enganar alguém ou alguma entidade ou escapar a obrigações legais". Este tópico tem vindo a ganhar cada vez mais relevância em tempos recentes, com novos casos a se tornarem públicos de uma forma frequente. Desta forma, existe uma procura contínua por soluções que permitam, numa primeira fase, prevenir a ocorrência de fraude, ou, caso a mesma já tenha ocorrido, a detetar o mais rapidamente possível. Isto representa um grande desafio: em primeiro lugar, a evolução tecnológica permite que se elaborem esquemas fraudulentos cada vez mais complexos e eficazes e, portanto, mais difíceis de detetar e parar. Para além disto, os dados e a informação que deles se pode retirar são vistos como algo cada vez mais importante no contexto social. Consequentemente, indivíduos e empresas começaram a recolher e armazenar grandes quantidades de todo o tipo de dados. Isto representa o conceito de *Big Data* – grandes quantidades de dados de diferentes tipos, com diferentes graus de complexidade, produzidos a ritmos diferentes e provenientes de diferentes fontes. Isto veio, por sua vez, tornar inviável a utilização de tecnologias e algoritmos tradicionais de deteção de fraude, uma vez que estes não possuem capacidade para processar um tão grande conjunto de dados, tão diversos. É neste contexto que a área de *Machine Learning* tem vindo a ser cada vez mais explorada, na busca por soluções que permitam dar resposta a este problema.

Normalmente, os sistemas de *Machine Learning* são vistos como algo completamente autónomo. Nos últimos anos, no entanto, sistemas interativos nos quais especialistas humanos contribuem ativamente no processo de aprendizagem têm vindo a apresentar um desempenho superior quando comparados com sistemas completamente automatizados. Isto pode verificar-se em cenários em que existe um grande conjunto de dados de diversos tipos e de diferentes origens (*Big Data*), cenários em que o *input* é um fluxo de dados ou quando existe uma alteração do contexto no qual os dados estão inseridos, num fenómeno conhecido por *concept drift*.

Tendo isto em conta, neste documento é descrito um projeto cujo tema se insere no contexto da utilização de aprendizagem interativa no suporte à decisão, abordando a temática das auditorias digitais e, mais concretamente, o caso da deteção de fraude fiscal. Desta forma, a solução proposta passa pelo desenvolvimento de um sistema de *Machine Learning* interativo e dinâmico, na medida em que um dos principais objetivos passa por permitir a um humano especialista no domínio não só contribuir com o seu conhecimento no processo de aprendizagem do sistema, mas também que este possa contribuir com novo conhecimento, através da sugestão de uma nova variável ou um novo valor para uma variável já existente, em qualquer altura. O sistema deve então ser capaz de integrar o novo conhecimento de uma forma autónoma e continuar com o seu normal funcionamento. Esta é, na verdade, a principal característica inovadora da solução proposta, uma vez que em sistemas de *Machine Learning* tradicionais isto não é possível, visto que estes implicam uma estrutura do *dataset* rígida, e em que qualquer alteração neste sentido implicaria um reinício de todo o processo de treino de modelos, desta vez com o novo *dataset*.

**Palavras-chave:** Fraude, Deteção de fraude, *Big Data*, *Concept Drift*, *Machine Learning* interativo

## 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 datasets.

Usually, Machine Learning systems are seen as something fully automatic. Recently, however, interactive systems in which the human experts actively contribute towards the learning process have shown improved performance when compared to fully automated ones. This may be so on 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 that learns and adapts in real-time by continuously incorporating user feedback, in a fully autonomous way. Moreover, it allows for users to manage variables (e.g. add, edit, remove), reflecting these changes on-the-fly in the Machine Learning pipeline. This paper describes 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:** Fraud, Fraud Detection, Big Data, Concept Drift, Interactive Machine Learning

## Contents

# List of Figures

# List of Tables

## List of Acronyms

**AL** – Active Learning

**AI** – Artificial Intelligence

**DAG** – Directed Acyclic Graph

**IML** – Interactive Machine Learning

**MAE** – Mean Absolute Error

**ML** – Machine Learning

**MSE** – Mean Squared Error

**OL** – Online Learning

$\boldsymbol{R^2}$ – R-Squared

**RMSE** – Root Mean Squared Error

# 1. Introduction

Fraud is an intentionally deceptive action designed to provide the perpetrator with an unlawful gain or to deny a right to a victim. There are several types of fraud (tax fraud, financial fraud, wire fraud, securities fraud, etc.) and it can be carried out by one individual, multiple individuals or a business firm as a whole. In recent times, this subject has become more and more relevant, mainly due to the leak of large groups of documents whose contents show how intricate networks of people and organizations have been created to conduct unethical or even criminal activities, including financial fraud, drug trafficking or tax evasion. Some of the notorious leaks include Panama Papers [1], Paradise Papers [2] and Luanda Leaks [3]. According to Infosecurity Magazine, fraud cost the global economy £3.2 trillion in 2018, and for some businesses, losses to fraud reach more than 10% of their total spending [4]. Furthermore, a 2020 study conducted by the Portuguese technology company Feedzai indicates that, due to the emergence of the Covid-19 pandemic and the consequent sudden increase in online purchases and transactions, the financial fraud rate has skyrocketed by 60.5% in a matter of weeks [5].

With this in mind, and considering the increasing complexity of fraudulent schemes, supported by constant technological evolution, it becomes necessary to find effective solutions to face this problem. In this regard, Machine Learning (ML) has been playing an increasingly important role. This branch of Artificial Intelligence (AI) has been around for some decades now, but, despite its relative age, it is now at one of its most exciting moments. This happens because the relationship between humans as a society and data has changed significantly. Nowadays, it is generally agreed that, when used correctly, data are one of, if not the most important asset of an organization, helping it to better understand their costumers, make better business decisions, improve business processes and monitor competitors, among others. With technological developments in processing power, storage, networks and other related technologies leading to a general availability and relatively low cost of processes for acquiring, storing, processing and managing data, everyone from individuals to large organizations started to collect them on a large scale. This led to organizations collecting increasing volumes of all types of data over time, with varying degrees of complexity (structured, semi-structured and unstructured), generated at different speeds, and varying degrees of ambiguity (due to the collected data being generated from different sources which can be uncertain and incomplete). This represents the concept of Big Data [6] – large volumes of diverse data that reach such proportions that make them impossible to process and extract some kind of value when using traditional technologies, processing methods or algorithms. So now, although organizations had the data they needed, they did not have the ability to extract value from them and consequently use them in decision support. This is where ML is quite useful – it can categorize the incoming data, recognize patterns and translate the data into insights helpful for business operations. This has made it an increasingly exploited solution, for which more and more applications are being found. One such application, which will be the one discussed in this paper, is in fraud detection.

## 1.1. Problem Description

The problem of fraud detection is not new, albeit it has been gradually changing over time alongside technological development. In essence, fraud detection refers to the analysis of large sets of transactions in order to find those that do not conform to a given standard or audit guideline, or simply those transactions that can be seen as outliers in the sense that they are statistically very diverse from the norm.

Traditionally, the solution found by most organizations to face this problem is to resort to rule-based systems [7]. These use pre-programmed behavior scripts to identify suspicious transactions, helping to automate procedures and checks that a human expert would typically handle. The system checks if a transaction meets any of the risky patterns expressed in the rules and if it does, it blocks it or sends it to be manually reviewed by humans. The downside of these types of systems is that they are static, in the sense that they can only identify situations for which there is a defined rule. This can be solved by gradually implementing new rules as new fraudulent behaviors become known, but in the meantime undetected fraudulent situations and consequent losses may have already taken place. One top of that, rule-based systems become more and more expensive to maintain as they grow larger and more complex. This complexity, in turn, increases the frequency of false positives, i.e., situations marked as potentially fraudulent when in fact they are not, leading to increased costs and work to try to reverse the situation, which is challenging, as it often results in an increase in false negatives, which have a much higher cost. In the age of Big Data, these methods are becoming impractical, so more and more organizations are starting to look for more effective solutions.

This makes the topic of fraud detection continue to be one of the most interesting case studies and areas of application of ML. Even though ML systems offer a much more proactive and flexible approach to identifying fraudulent instances, enabling organizations to detect irregularities and identify subtle changes in large datasets in a much more precise and granular way than was previously possible, there are still some challenges that need to be considered:

1. Fraud detection datasets are usually very biased, with fraud instances constituting only a small minority. This makes the dataset unbalanced and any ML model will most likely predict every instance it is given as not fraudulent.
2. Fraud patterns may change over time, deeming existing ML models irrelevant. Something considered to be fraudulent today may not be so in the future, so the patterns previously learned by the models become outdated, consequently affecting the predictions made by them.

3. Fraud detection datasets are often very large, calling for the use of distributed and parallel technologies and approaches. This is because in addition to being large, these datasets share a number of characteristics of the so-called Big Data that make them impossible for traditional algorithms to process.
4. Given the sensitivity of the task, fraud detection increasingly needs explainable and transparent approaches, that provide the human decision-maker with the necessary elements for taking an informed and properly grounded decision.

These challenges will be discussed in detail in the Fraud Detection section.

## 1.2. Objectives

The main goal of this project is to develop and implement an innovative solution that answers the problems described above regarding the issue of fraud detection, especially with regard to false positives.

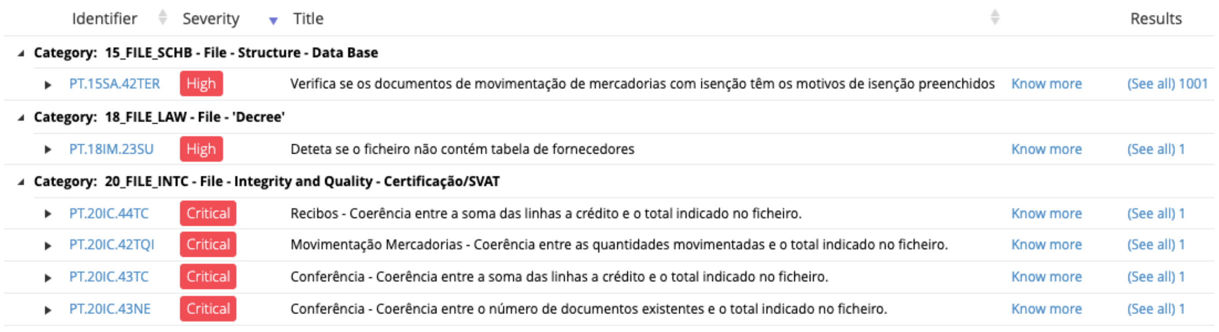
That said, the goals include:

- Development of an Interactive Machine Learning (IML) system capable of integrating user-provided feedback through instance validation, in an autonomous way.
- Allow new knowledge (variables or values of variables) to be suggested by a user at any time when using the system. The system must be able to integrate this new knowledge autonomously.
- Ensure the system evolves in identifying true positive instances, thus mitigating the identification of false positive ones.

## 1.3. Contextualization

All the work described throughout this document was developed in the context of the Neurat project (reference 3990 – 31/SI/2017), which arises from a collaboration between the Polytechnic Institute of Porto and the Petapilot company. For this reason, the proposed and implemented solution was based on the practical case of the Col.bi digital audit software, one of the main products of Petapilot. This software performs a set of digital audit tests that are based on a set of predefined audit rules, and consequently flagging potentially fraudulent cases. Each of these cases is defined as a finding. In the context of an organization, the number of findings resulting from these tests can run into thousands, and it is common for the same finding to be repeated hundreds of times (Figure 1).

Considering the fact that this is a rule-based system and therefore quite susceptible to the occurrence of false positives, and that there is no way to distinguish them from true positives, the auditors' work is made more difficult and their workload increased, as they have to analyze every single case.



| Identifier | Severity | Title | | Results |
|---|---|---|---|---|
| **Category: 15_FILE_SCHB - File - Structure - Data Base** | | | | |
| PT.15SA.42TER | High | Verifica se os documentos de movimentação de mercadorias com isenção têm os motivos de isenção preenchidos | Know more | (See all) 1001 |
| **Category: 18_FILE_LAW - File - 'Decree'** | | | | |
| PT.18IM.23SU | High | Deteta se o ficheiro não contém tabela de fornecedores | Know more | (See all) 1 |
| **Category: 20_FILE_INTC - File - Integrity and Quality - Certificação/SVAT** | | | | |
| PT.20IC.44TC | Critical | Recibos - Coerência entre a soma das linhas a crédito e o total indicado no ficheiro. | Know more | (See all) 1 |
| PT.20IC.42TQI | Critical | Movimentação Mercadorias - Coerência entre as quantidades movimentadas e o total indicado no ficheiro. | Know more | (See all) 1 |
| PT.20IC.43TC | Critical | Conferência - Coerência entre a soma das linhas a crédito e o total indicado no ficheiro. | Know more | (See all) 1 |
| PT.20IC.43NE | Critical | Conferência - Coerência entre o número de documentos existentes e o total indicado no ficheiro. | Know more | (See all) 1 |

Figure 1 - List of findings resulting from Col.bi's digital audit tests

This software then represents a case of the previously described rule-based systems.

In this way, the goal is to develop a solution that can learn from the auditor’s work, becoming more and more effective in identifying the truly positive cases, in order to facilitate the auditors’ task.

## 1.4. Proposed Solution

Given the context presented above, it was proposed the development of an intelligent, knowledge-oriented decision support system, capable of identifying useful patterns hidden in the data subject to analysis and inferring corresponding logical propositions in order to improve, incrementally and autonomously, the system’s precision and accuracy in identifying truly positive occurrences.

Taking into account the various ML approaches that already exist for fraud detection, in order to make the proposed solution innovative, it was defined as one of the most important requirements the possibility for a human expert to suggest new knowledge, either in the form of a completely new variable, or a new value for an already existing one, with the system being able to adapt and integrate this knowledge in an autonomous way. This made it impractical to develop and implement a ML mechanism through a traditional approach, since this would imply a rigid dataset structure, with a well-defined number of variables and values, and where any change in this direction would make the use of the system unfeasible.. This is because in supervised ML, a model is trained using a labeled dataset (where all values are known), which is then used to make predictions. During the training phase, the model “learns” the relationship between the independent variables (inputs) and the corresponding dependent variable (output). Once this phase is completed, the model is able to make predictions for instances where the value of the dependent variable is unknown. Adding a new variable to the dataset would make the model outdated, since the variable was not present during the training phase and consequently the model would not know it or take it into account when making predictions, which could affect its results. In this way, the whole process would have to be repeated whenever new knowledge was suggested.

With this in mind, along with the challenges described in Section 1.1, the proposed solution was to develop an interactive and dynamic ML mechanism, following a set of approaches known as human-in-the-loop, such as Online Learning (OL) and Active Learning (AL), which rely on human actors to improve learning performance.

In traditional approaches, the human involvement is generally at the early stages of the ML process, such as data collection and curation, or feature engineering. In the approach followed, the human is placed right in the center of the learning process and assumes the role of guiding and facilitating the process. The approach is thus interactive (in the sense that there is human interaction during the learning stages) and iterative (in the sense that learning takes place over multiple iterations). In general, the functioning of the suggested system can be described as follows (Figure 2): models are trained on a given input and then used to make predictions. These predictions are then presented to a human expert, who must analyze them and provide feedback, which is incorporated to be considered in the next iteration of model training. It is expected that with the feedback provided by the human the system will evolve in an iterative manner, becoming more and more effective, over time, and increasingly releasing the human expert from the more repetitive tasks.

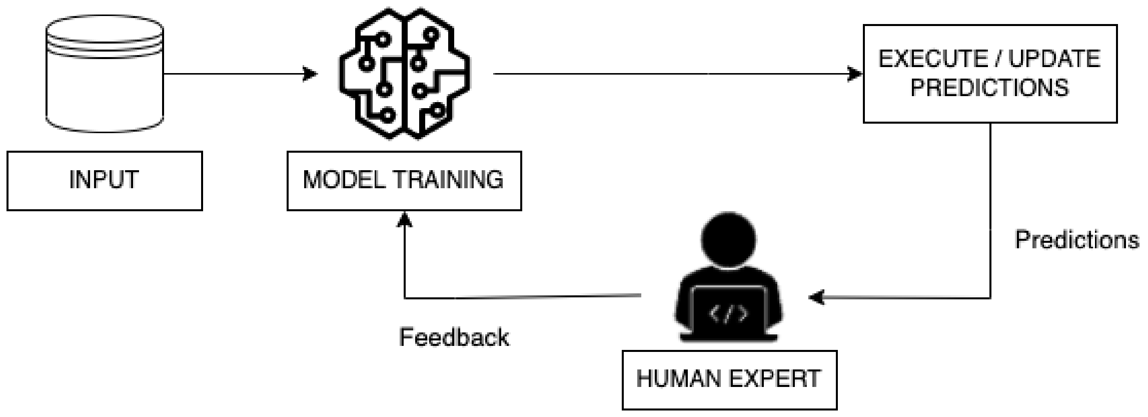


Figure 2 - Basic operation of an interactive Machine Learning system

## 1.5. Research Methodology

In this project it was followed the Design Science Research (DSR) Methodology [8], a form of science knowledge production that involves the development of innovative constructions, intended to solve real world problems. The main outcome of this type of research methodology is an artifact that solves a particular domain problem, also known as solution concept, which must be assessed against criteria of value or utility. In the context of this project, this means presenting an innovative solution to a problem that covers several domains. This methodology has been pointed out as a suitable research approach when researchers work in close collaboration with organizations (as is the case in this project), as it allows new ideas to be tested in a real-life context.

A DSR process commonly includes six steps or activities:

1. **Identification of the problem,** defining the research problem and justifying the value of a solution.
2. **Definition of objectives** for a solution.
3. **Design and development** of artefacts (constructs, models, methods, etc.).
4. **Demonstration** by using the artefact to solve the problem.
5. **Evaluation** of the solution, comparing the objectives and the actual observed results from the artefact.
6. **Communication** of the problem, the artefact, its utility and effectiveness to other researches.

Throughout this document, which represents the sixth and final step in this process, the remaining steps will also be presented: The first and second steps have already been described earlier in this section, while the third, fourth and fifth steps, which essentially describe the implemented solution and its results, will be described in a later section.

## 1.6. Document Structure

The structure of this document is described below.

In section number 2 there is a theoretical background concerning the two main domains of this work: ML and Fraud Detection.

In section number 3 the proposed solution and its implementation is described in detail.

In section number 4 all the results resulting from the realization of this project are presented. Here 3 types of results are addressed: those related to the full version of the solution, those related to the demo version and those related to the use of AL.

Section number 5 presents all the conclusions of the work done, a list of all the scientific publications published and an analysis of the work that can be done in the future in the context of the project.

# 2. Theoretical Background

## 2.1. Machine Learning

ML is a subset of Artificial Intelligence which has been studied since the late 1950s. In fact, it was in 1959 that IBM published a paper in the *IBM Journal of Research and Development* [9] whose title aroused a great deal of curiosity, much due to the general ignorance about the subject at the time. Authored by Arthur Samuel, an American pioneer in the field of computer gaming and AI, the paper titled "*Some Studies in Machine Learning Using the Game of Checkers"* explored the use of ML in the game of checkers [10] "to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program" [10]. From this paper, the ML concept, as it is known today, was born.

Since then, this concept has become increasingly relevant, and several definitions have been presented by different authors in an attempt to answer the question "*what is Machine Learning?"*. IBM, for instance, defines ML as "a branch of artificial intelligence and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy" [11].

In short, what a "learning machine" does is find a mathematical formula that, when applied to a set of inputs, the so-called "training data", produces the desired outputs. Once trained, the machine should be able to predict and generate the correct outputs for other inputs (completely new, not used during the training process), as long as those inputs come from the same or a similar statistical distribution as the one the training data was drawn from.

There are many different types of ML systems, which makes the concept extremely complex. However, it is possible to classify them into broader categories based on:

1) Whether there is or not human supervision during the learning process (supervised, unsupervised, semi-supervised and reinforcement learning).
2) Whether or not they can learn incrementally in real time (online vs offline learning).

### 2.1.1. Learning Types

ML systems can be classified according to the amount and type of supervision they receive during the training phase of the models. There are four main categories: supervised learning, unsupervised learning, semi-supervised learning and reinforcement learning.

#### 2.1.1.1. Supervised Learning

Supervised learning is the most popular and frequently used type of ML.

In supervised learning [12], the algorithm is provided with a labeled dataset, i.e., a dataset in which the values of both the independent variables (inputs) and the dependent variable (output) are known from the start. Then, during the training phase, the algorithm deciphers patterns that exist in the data and creates a model with the mapping between the inputs and their respective output, capable of reproducing the same underlying logic with new data, that is, instances of which the values of their inputs are known, but the value of the output is unknown. The term "supervised" comes from the fact that there is human intervention in this type of learning, namely in the dataset labelling process. The fact that the dataset is labeled and therefore the output associated to each instance is known in advance, allows the performance of the model to be evaluated.

As far as supervised learning problems are concerned, they can be further grouped into Regression and Classification problems. Both follow the general logic and functioning of supervised learning, being only distinct with respect to the type of output generated.

- **Classification** is used when the output variable is categorical i.e., with 2 or more classes. One of the best known and studied examples of this type of problem is the detection of spam emails [13]. Here, the goal is for the algorithm to be able to, based on the variables of a given email (inputs), classify it as spam or not spam (categorical output). Some of the common types of classification algorithms are the linear classifiers, support vector machine, decision tree and random forest.
- **Regression** is used when the output variable is a real or continuous value. An example of this type of problem is the prediction of the price of a house [14]. The algorithm "learns" the relationship between the variables of the houses (inputs) and their price (numerical output) and is able to predict the price of houses for which the variables are known, but not the price. The most common regression algorithm is the linear regression.

Although there are algorithms exclusive to one type of problem, such as the support vector machine in the case of classification or the linear regression in the case of regression, there are also algorithms that can be used in both, such as the decision tree or the kNN (k – Nearest Neighbors).

In the book "Machine Learning for dummies" [15], the author ends up comparing this type of learning to the way humans learn under the supervision of a teacher: "*The teacher provides good examples for the student to memorize, and the student then derives general rules from these specific examples.*".

### 2.1.1.2. Unsupervised Learning

In the case of unsupervised learning [16], contrary to supervised, not all variables and data patterns are classified. Algorithms therefore work on datasets where it is not possible to establish a mapping between inputs and outputs, since the latter are not known. Thus, these focus on discovering hidden patterns or groups of similar data. The most challenging aspect of this type of learning is that without human intervention and consequently without a labeling of the datasets, there is no way to evaluate the performance of the algorithm, since it is not possible to compare the results obtained by it with outputs (which are not known).

The k-means clustering algorithm is a popular example of this type of learning. In fact, clustering can be considered the most important type of unsupervised learning problem (Figure 3) . Here, the goal of the algorithm is to group data in which it finds similarities or differences. Thus, a cluster can be defined as a collection of data that are similar to each other and different from data belonging to other clusters.

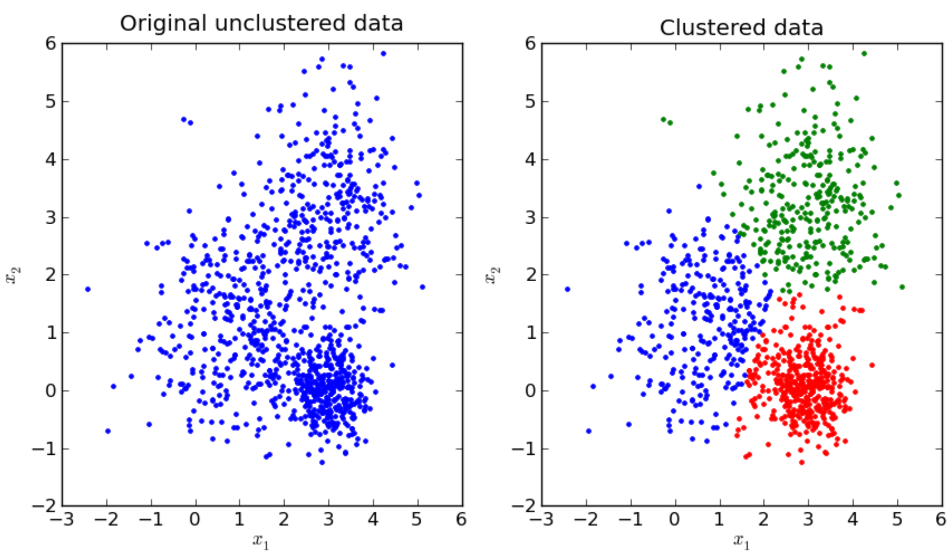


Figure 3 - Example of how a clustering algorithm works (Source: [17])

The main advantage of unsupervised learning is that it allows the discovery of previously unknown patterns in the data, which can then be considered.

The clustering technique and its algorithms have been extensively studied and more and more applications are being found in fields such as medicine, marketing, social science, among others [18]. One of the most popular examples of the application of this type of learning, and one that most people are in contact with, are recommender systems [19].

About unsupervised learning, the author of the book “Machine Learning for dummies” [15], once again drawing a parallel with the way humans learn, states that “*as a kind of learning, it resembles the methods humans use to figure out that certain objects or events are from the same class, such as by observing the degree of similarity between objects”*

#### 2.1.1.3. Semi-Supervised Learning

Semi-supervised learning [20] is a combination of supervised and unsupervised learning which aims to address the main drawbacks presented by each. In supervised learning, a considerable amount of labeled data is required, which involves the need for a labeling process performed by a human, which is cost-effective and time consuming. On the other hand, although the dataset labeling process is not required, the base of potential applications for purely unsupervised learning is rather limited. That said, semi-supervised learning algorithms are trained on a dataset that contains a mix of labeled and unlabeled data, although unlabeled data usually predominates. The goal of these algorithms is to combine supervised and unsupervised learning by improving the performance of one by using data generally associated with the other. For instance, when dealing with a classification problem (supervised learning), additional data points for which the label is unknown might be used to help in the classification process. For clustering methods (unsupervised learning), the learning procedure might benefit from the knowledge that certain points belong to the same class.

There are many applications of this type of learning in fields such as medicine [21] or image categorization [22].

#### 2.1.1.4. Reinforcement Learning

Reinforcement learning is the most advanced algorithm category in ML [23]. This type of learning occurs when the algorithm is provided with unlabeled examples, as in unsupervised learning. However, these examples are usually accompanied with positive or negative feedback according to the solution the algorithm proposes. The algorithm must make decisions, and the decisions bear consequences.

The best way to explain this kind of learning is to make an analogy with a video game. As a player progresses through the virtual space of a game, they learn the value of various actions under different conditions and become more familiar with the field of play. Those learned values then inform and influence a player’s subsequent behavior and their performance immediately improves based on their learning and past experience [24]. This analogy has been widely studied and even put into practice, as in this example [25] where one of the most popular semi-supervised algorithms, Q-learning, was used to learn how to play the famous Super Mario game.

As could not be otherwise, also for this type of learning the author of “Machine Learning for dummies” [15] draws an analogy with human learning: *“In the human world, it is just like learning by trial and error. Errors help you learn because they have a penalty added (cost, lost time, regret, pain, and so on), teaching you that a certain course of action is less likely to succeed than others.”*.

All these analogies to human learning for each type of learning meet IBM’s previously presented definition of ML: *“…focuses on the use of data and algorithms to imitate the way that humans learn”*.

### 2.1.2. Offline Learning vs Online Learning

A ML system can be classified not only by the type of learning applied, but also by the way it learns, i.e., whether or not the system can learn incrementally from a stream of input data [26]. In offline learning (or batch learning), the algorithm is trained on a set of resting data, available from the beginning of the learning process. In OL, on the other hand, the algorithm is trained incrementally as new data becomes available.

#### 2.1.2.1. Offline Learning

Offline Learning, also known as Batch Learning, is the traditional way algorithms learn in ML. In this case, the algorithm is fed with an initial dataset containing all the available information at that moment, which is then used to create and train a model (Figure 4) [26]. These models are static in nature which means that once they get trained, their performance will not improve until a new model gets retrained.

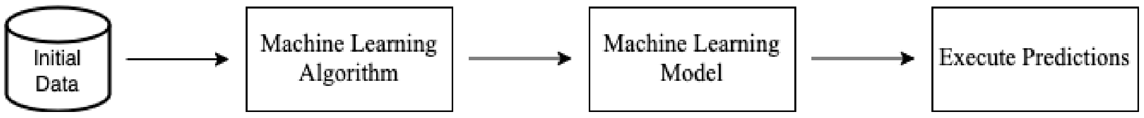


Figure 4 – First iteration of batch learning

Since the model is unable to learn incrementally from a stream of live data, as new data becomes available, it accumulates, leading to a new batch. The model retraining is then done with the complete dataset available, composed of old data (already used in the previous training) and new data (Figure 5). Usually this happens at regular intervals such as weekly, monthly, quarterly, etc.

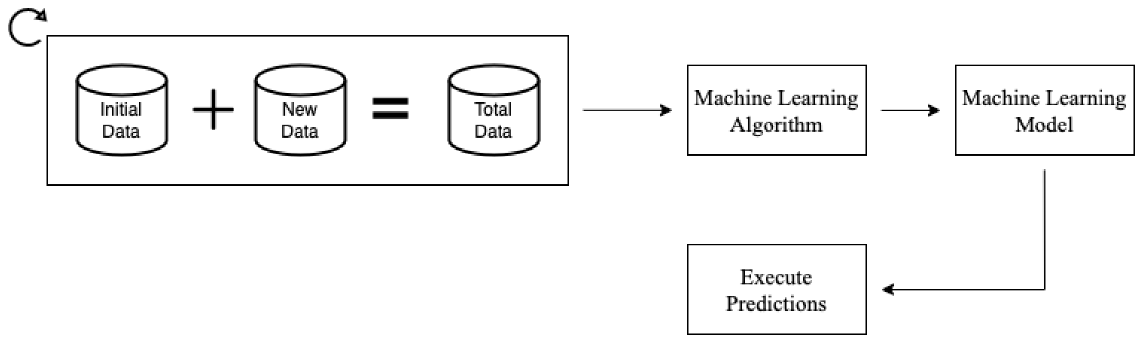


Figure 5 – Retraining of a model in batch learning

The fact that the model must be invariably trained with large amounts of data represents the main and major disadvantage of this way of learning, since it means that more time and resources such as CPU and memory space are needed. Previously, data was referred to as showing a few changes over time, or even stay unchanged, and offline learning was used. This was perfectly acceptable, even when considering both the time and computational resources. More recently, however, when dealing with streaming data, using a batch approach to create models is no longer feasible. ML algorithms needed to evolve over the years to bridge the differences that have naturally arisen with the technological evolution and the increasing amount of generated and produced data. OL emerged in this context: when there is a need for users to be able to train models in real-time and simultaneously improve accuracy performance.

#### 2.1.2.2. Online Learning

OL [27] is a subfield of ML that has emerged to overcome the drawbacks of batch learning and it has received a lot of attention in recent years given the need to respond to scalability problems in Big Data. In OL the model is trained in an incremental manner as new labeled data arrives, in a sequential way, which can be helpful in data streaming environments. The main goal of OL is that the model will be able to improve its performance (accuracy) with respect to the predictions it makes, based on previous predictions that were made correctly. Taking the case of spam mail detection as an example, a model is trained with an initial set of labeled data. After performing a set of predictions for new data, these are compared with the correct answers. The quality of the model is assessed by a loss function that measures the discrepancy between the predicted answer and the correct one. The goal is for the model to be able to minimize this cumulative loss over time, becoming more and more accurate [27].

This way of learning is especially suited for ML systems that receive data as a continuous flow and need to adapt to possible changes rapidly or autonomously, since OL algorithms are the most scalable and efficient approach to resort to in large-scale ML tasks and in real-world data analytics problems [27].

The main disadvantage of OL has to do with the quality of the information: a model can give predictions only as good as the quality of the training data fed to it. If it is fed with bad data, the performance will be very poor, and the user will see the impact instantly. This can, however, be overcome by involving humans in the ML process. In this sense, IML is increasingly relevant.

#### 2.1.2.3. Interactive Learning

In what regards the relationship between human experts and AI 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 data. The human may be involved in the process of data labelling 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 human action 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 human, it is deemed IML [28]. IML 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 [29]. Or it may happen that the problem that's being solved is computationally hard, and the involvement of human experts may help speed up or simplify the learning process [30]. IML 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 ML 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 IML can thus be depicted as in Figure 6: the human expert analyses the prediction of a model for a given input and provides his feedback, which is then incorporated into the model somehow to iteratively improve it over time.

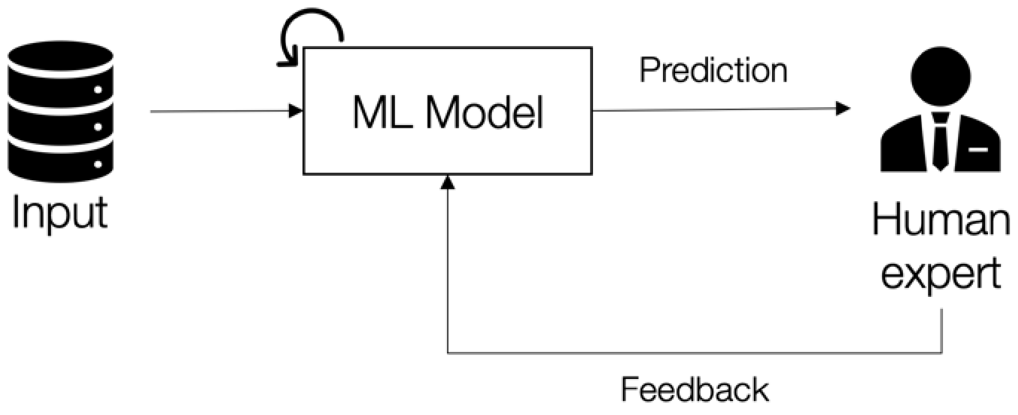


Figure 6 - Interactive Machine Learning

There are many different approaches to integrate human knowledge and skills into ML models. For instance, in [30] 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 [31]. 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 ML techniques.

Healthcare in particular, in which datasets are often small, can benefit especially from a 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 ML algorithms [32]. Specific applications exist, including in the annotation of medical imaging [33] or in knowledge discovery in bioinformatics [34].

That said, the major goal of IML is to allow feedback provided by human experts to be incorporated into the model training process, allowing for faster learning and consequent performance improvement (accuracy) over iterations. Despite this, it is possible to make the learning process even faster and precise by integrating AL into the system.

#### 2.1.2.4. Active Learning

AL, also known as "query learning", is a subfield of ML and the main idea is that the learning algorithm can choose the data it wants to learn from, by querying an "oracle" (or the human) to label some of the unlabeled instances [35] [36]. The process is exactly the same as that followed in IML, but instead of the human expert randomly choosing unlabeled data to label, it is the system itself that asks the human to label certain data that will contribute to an improvement in its performance. This allows the system to require fewer instances to gradually become more accurate.
Several advantages result from AL [36] [37] [38] :

1. Easing the data labelling process, as the AL system incorporates human experts in the learning process. It often accomplishes better performance results since the process of data labeling could be complicated and difficult to perform. When dealing with large datasets, this process can become expensive, time-consuming, and complex. The AL systems help to improve the process of labeling new data and minimize the cost of acquiring labeled data.
2. Maximizing prediction accuracy using fewer labeled instances.
3. Improving the learning rate, as the AL system is allowed to select which unlabeled instances to query by the human expert, the learning algorithm will achieve better results using a lower number of labeled instances, with fewer iterations and consequently less training time. There are some approaches to unlabeled instances selection in AL systems, namely:
    a) Random Sampling, in which the unlabeled instances are randomly chosen [39].
    b) Uncertainty Sampling, in which the chosen unlabeled instances are close to the decision boundary and their class is unclear [40] .
    c) Diversity Sampling, in which the chosen unlabeled instances are unknown and new to the model.

### 2.1.3. Machine Learning Algorithms

The choice of which ML algorithm to use is very important and depends on several factors such as the size and structure of the available dataset. This section presents some of the main algorithms used in the various types of ML – supervised (classification and regression), unsupervised, semi-supervised and reinforcement learning.

#### 2.1.3.1. Linear Regression

Linear regression is one of the most well-known and easy to understand algorithms in ML. This algorithm can be used in supervised learning cases and performs a regression task, which is the prediction of the value of one variable (dependent) based on the value of another variable (independent). So, this regression technique finds out a linear relationship between an input (independent variable) and an output (dependent variable).

Given its characteristics and popularity, this algorithm is used in everything from biological, behavioral, environmental and social sciences to business, and it is in the latter that it can have a significant impact, helping leaders of organizations make better decisions by transforming large amounts of data into relevant information.

In this simple, fictitious example [41]**,** this algorithm was used to help an investor decide whether or not to buy a few shares of a particular start-up that in its first three years of existence had considerable value growth, as can be seen in Figure 7A**.**

Now, the problem is that when it comes to investments, what matters in not the value of the stock when it is purchased but rather the expected values it may reach in the future. At this stage, linear regression was used to predict the value of the dependent variable (stock value) based on the value of the independent variable (the year). The algorithm does this by finding the line that best fits the data points on the plot, making it possible to predict the future value of the stock more or less accurately (Figure 7B**)**.

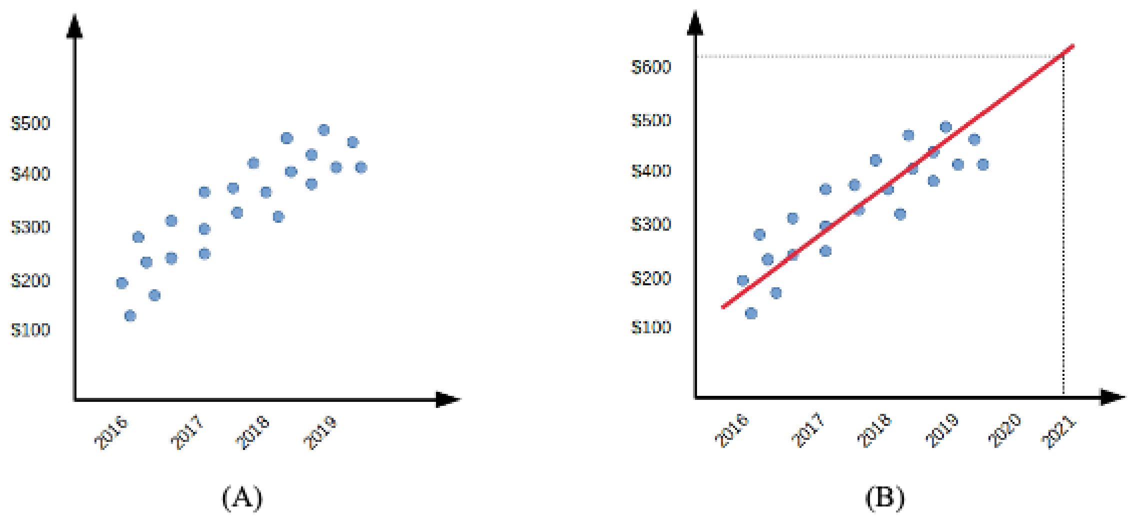


Figure 7 - Linear Regression example (Source: [41])

### 2.1.3.2. Decision Tree

Decision Tree is a supervised learning algorithm that can be used to solve both classification and regression problems [42]. In general, the operation of this algorithm boils down to a continuous splitting of data based on a given parameter. To do this, it uses a representation of a tree (Figure 8), where each internal node denotes a test on an attribute, i.e., a point in which a decision has to be made, each branch represents an outcome of the test, and each leaf node holds a class label. In the simplest and most frequent case, each test considers a single attribute, such that the instance space is partitioned according to the attribute's value. In the case of numeric attributes, the condition refers to a range.

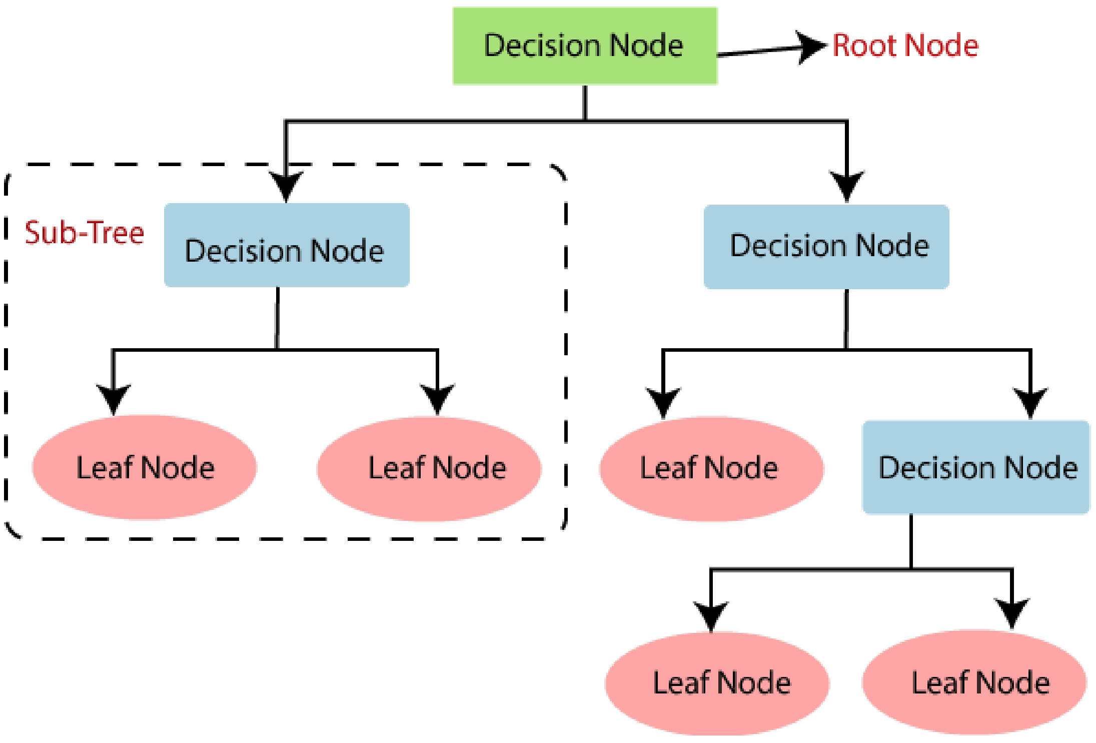


Figure 8 - Decision Tree structure (Source: [43])

Decision trees classify the examples by sorting them down the tree from the root node, that represents the entire sample or dataset, to some leaf node, which can also be called terminal node since it represents a node that does not split.

The popularity of this algorithm can be explained by the fact that it is simple to interpret and understand when compared to other algorithms, which may be due to the fact that it shows a tree-like structure and usually mimics human thinking ability while deciding. On the other hand, this algorithm can be unstable, and it can be difficult to control the size of the resulting tree.

This is a very popular algorithm in the field of anomaly detection as can be seen from its many applications in this area [44] [45].

### 2.1.3.3. K-Means

K-Means is an unsupervised learning algorithm, often used in solving clustering problems [46]. This algorithm groups the unlabeled dataset into a pre-defined number of clusters that need to be created in the process, represented by the *k*. If *k*=3, there will be three clusters, and for *k*=4, there will be four clusters, and so on. The process is repeated until the algorithm does not find better clusters.

The first step is to define the value of *k*. The fact that this value must be defined *a priori* represents one of the main disadvantages of this algorithm, but for which there are already several approaches that allow the automatic determination of its value [47]. Once the value is defined, the algorithm selects *k* random points, called centroids. These points may or may not belong to the dataset. In the example shown in Figure 9 three centroids are represented by the blue, green and red crosses.

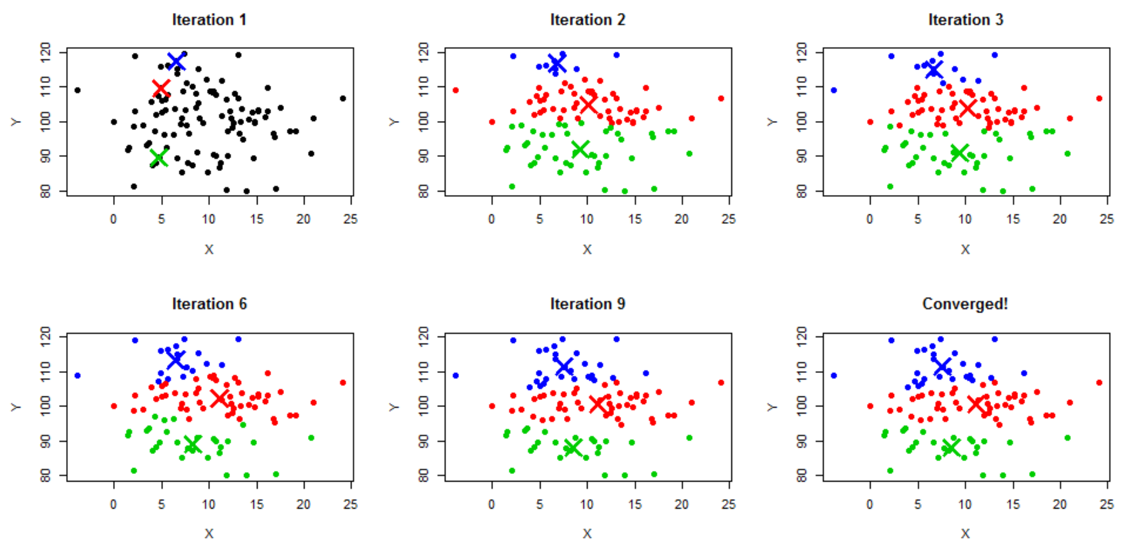


Figure 9 - K-Means algorithm example (Source: [48])

Each data point is then assigned to their closest centroid, which will form the predefined *k* clusters. At this stage, the means value is recalculated – the values of the centroids. The new value of a centroid is the sum of all the points belonging to that centroid divided by the number of points in the cluster. This process is repeated until no centroid changes its value in re-calculation, meaning that each centroid has centered itself in the middle of its cluster.

This algorithm has several advantages, among which the fact that it is relatively simple to implement, scales to large datasets and easily adapts to new examples. This makes it very popular and applied in different scenarios such as customer segmentation [49] and recommender systems [50]**.** In fact, applications based on costumer characteristics and identification of buying patterns are among the ones that can most take advantage of this type of algorithm.

### 2.1.3.4. Random Forest

Random Forest [51] is a popular supervised ML algorithm that can be used in both regression and classification problems, although it is in the latter where the algorithm performs better. In fact, this algorithm has gained a substantial interest in ML because of its efficient discriminative classification. As the name implies, this algorithm consists of a number of individual decision trees (Figure 10). Each of these trees is made by randomly selecting data from the dataset and returns a prediction. This prediction can be a numerical value, in regression problems, or a categorical value, in classification problems. As for the final prediction – the one returned by the algorithm – it is also dependent on the type of problem. In regression problems, the average of the values predicted by each of the trees used in the algorithm is returned. In classification problems, on the other hand, the final prediction consists of the class most predicted by the trees.

Each resulting tree can be totally different from the other, and individually each has low accuracy, however when combined, they usually achieve better accuracy than the uniqueness of a single decision tree.

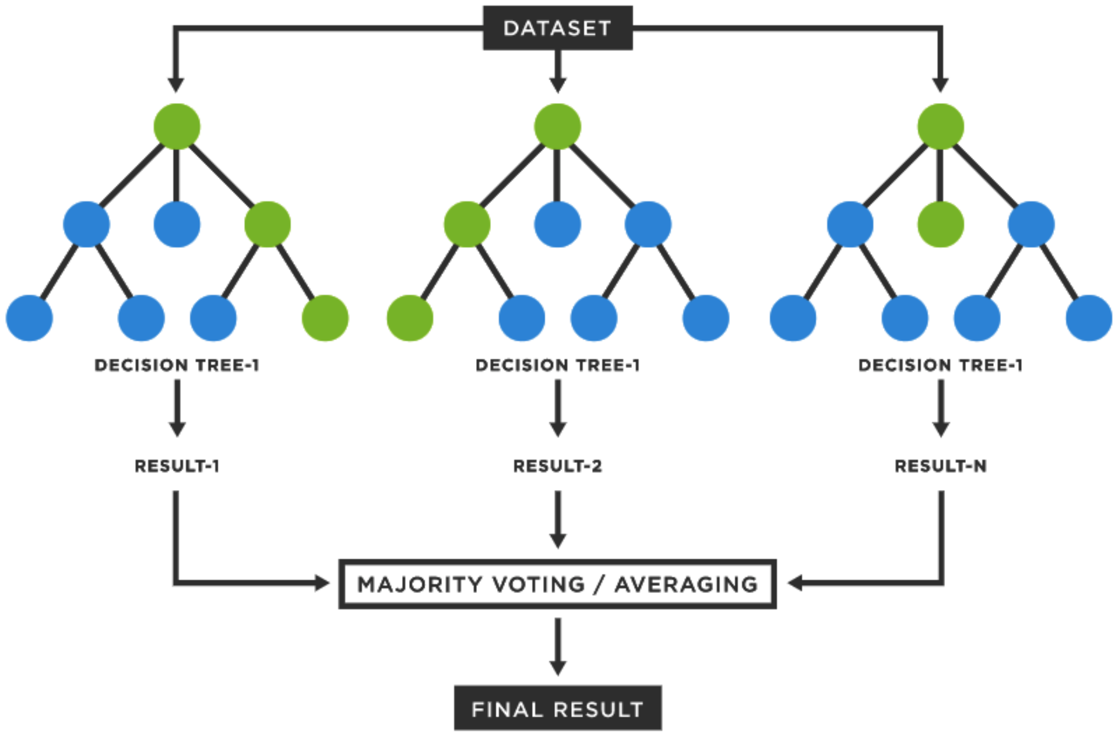


Figure 10 - Random Forest algorithm example (Source: [52])

The popularity of this algorithm owes much to the fact that it can handle large datasets with high dimensionality, predicting the outputs with great accuracy. In addition, it helps to prevent the overfitting issue. There are applications of this algorithm in several areas, including banking, marketing, medicine and bioinformatics [53].

### 2.1.4. Performance Evaluation of Machine Learning Models

Evaluating the performance of a ML model is a crucial task, but one that presents many challenges [54]. Indeed, the ability to correctly evaluate a ML model is imperative in order to ensure that the predictions returned by it make sense with respect to their context. However, there are so many different performance metrics that can be used that it is often overwhelming to choose which one to use. Different ML tasks have different evaluation metrics. Generally speaking, these can be divided into regression and classification.

#### 2.1.4.1. Classification Performance Metrics

Classification problems can be divided into binary classification problems and multiclass classification problems. In binary classification, the output is restricted to two classes, for example, positive or negative. In multiclass classification, the output can assume more than two classes. This is something that needs to be taken into account when it comes to analyzing the performance of a model in a classification problem, since there are metrics whose calculation is only possible in binary classification problems.

In general, the metrics for performance evaluation of models responsible for classification tasks are based on two key concepts: the real outcome and the predicted outcome. This outcome predicted by the model, in turn, is of one of four types:

- **True Positive (TP)** – when a certain instance is predicted to be positive, and its label is actually positive.
- **False Positive (FP)** – when a certain instance is predicted to be positive, but its label is actually negative.
- **True Negative (TN)** – when a certain instance is predicted to be negative, and its label is actually negative.
- **False Negative (FN)** – when a certain instance is predicted to be negative, but its label is actually positive.

Although not a metric for evaluating the performance of a model, the Confusion Matrix (Figure 11) is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes. It consists of a tabular summary of the number of correct and incorrect predictions made by the model in question. In case of a binary classification task, a confusion matrix is a 2x2 matrix. If there are three different classes, it is a 3x3 matrix and so on.

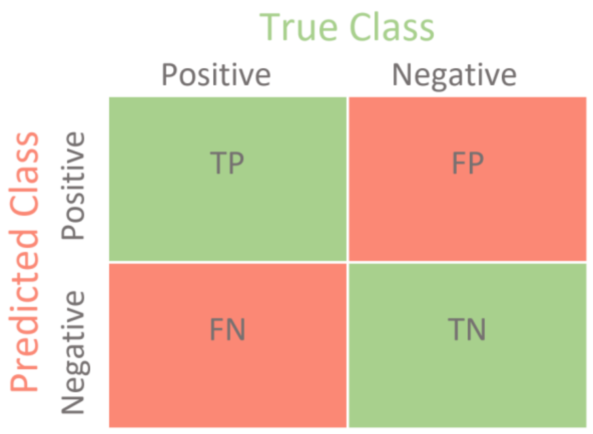


Figure 11 - Confusion Matrix (Source: [55])

Moreover, this matrix can also be used as an aid in calculating performance metrics such as those mentioned below.

**Accuracy** is perhaps the simplest metric to use and implement. It represents the fraction of predictions the model got right out of all the predictions. The accuracy of a model is obtained by dividing the number of correct predictions by the total number of predictions returned by the model.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + False\ Positives + True\ Negatives + False\ Negatives}$$

This metric ranges between 0 and 1. If the model fails all predictions and consequently there are no true positives or true negatives, its accuracy is 0. In turn, if the model gets all the predictions right there are no false positives or false negatives, so the numerator equals the denominator, bringing the accuracy value to 1.

Accuracy, however, may not always assume a reliable value relative to model performance, especially in cases where the data is unbalanced. For instance, considering an example where there are 100 instances, 95 of which labeled as belonging to class ‘yes’ and 5 labeled as class ‘no’, a model that predicts the class ‘yes’ for all instances turns out to have an accuracy of 0.95, or 95%.

**Precision** can be defined as the proportion of positive predictions that were actually correct. This is one of the metrics usually used to overcome the limitations of accuracy. The value of this metric is obtained by dividing the number of instances correctly predicted as positive by the total number of positive correct or incorrect predictions.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

**Recall**, also known as sensitivity or hit rate, aims at measuring what proportion of actual positives was identified correctly. It is calculated as the ratio between the number of positive instances correctly classified as positive (true positives) to the total number of positive instances. In general, this metric measures the model's ability to detect positive instances. The higher the recall, the more positive instances detected.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

**F-Score**, also known as F1-score, is a less known performance metric that combines the precision and recall of a model into a single metric by taking their harmonic mean. This metric is primarily used to compare the performance of two classifiers: if a classifier A has a higher recall and a classifier B has higher precision, the f1-scores for both the classifiers can be used to determine which one produces better results.

$$F1 - Score = \frac{2 * \ True\ Positives}{2 * True\ Positives + False\ Positives + False\ Negatives}$$

The highest value of an F1 Score is 1, indicating perfect Precision and Recall, and the lowest possible value is 0 if either the Precision or the Recall is zero.

Of the metrics presented here, only *Accuracy* is common to binary and multiclass classification problems. The calculation of the remaining ones is only possible in binary classification problems.

#### 2.1.4.2. Regression Performance Metrics

The performance metrics of models responsible for regression tasks are based on the concept of error. In this context, the error represents the difference, or delta, between the actual outcome and the predicted outcome. The higher the difference between these two, the more one can assume that it is a weak model, whose predictions do not correctly represent the context in which they are set.

**Mean Absolute Error (MAE)** calculates the average difference between the actual outcomes and the predicted outcomes. Since this is a loss function, the objective is for its value to be the minimum.

$$MAE = \frac{\sum_{i=1}^{N}(Actual\ outcomes - Predicted\ Outcomes)}{N}$$

**Mean Squared Error (MSE)** can be considered the simplest and most common error metric for regression problems. It basically calculates the difference between the actual and the predicted outcomes, squares these results and then computes their average.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(Actual\ outcomes - Predicted\ Outcomes)^2$$

Just like MAE, a perfect mean squared error value is 0.0, meaning that all predictions (predicted outcomes) matched the expected values (actual outcomes) exactly.

**Root Mean Squared Error (RMSE)** is an extension of the MSE. Just like MSE, it also calculates the average of the squared errors across all instances but, in addition, takes the square root of the result, effectively taking the square root of MSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Actual\ outcomes - Predicted\ Outcomes)^2}{N}}$$

**R-Squared ($\boldsymbol{R^2}$)** it's a metric that's calculated using other metrics. This metric gives an indication of how good a model fits a given dataset by defining the degree to which the variance in the dependent variable can be explained by the independent variables. Ideally, it would be desirable if the independent variables were able to explain all the variations of the dependent variable, representing a model that fits the dataset perfectly. In this case, the R-Squared value would be 1. The higher the r-squared value, the better the model.

## 2.2. Fraud Detection

The topic of fraud detection is of such importance and dimension that there are books, guides and organizations devoted exclusively to it. Still, it is common for there to be some confusion between this concept and that of fraud prevention, so it is necessary to first make a clear distinction between the two. Fraud prevention consists of a set of measures, such as the personal identification numbers for bankcards or passwords on computer systems, implemented with the objective of preventing fraud from occurring in the first place. Fraud detection, on the other hand, comes into play once fraud prevention fails, with the goal of identifying fraud as quickly as possible once it has been carried out. This topic is constantly evolving, much by necessity, since with technological evolution the fraudulent schemes supported by it also evolve, becoming increasingly complex and difficult to stop. Sometimes the evolution of fraud detection techniques can be complicated, since there is no point in discussing existing solutions in detail in the public domain, so as not to make them known to perpetrators, and the sharing of datasets and experiment results is very limited [56].

From a technical standpoint, fraud detection traditionally derives from the broader problem of anomaly detection [57]. When approached in this way, the goal is to find data points or groups of data points that deviates from what is considered the normal pattern. In this regard, different types of anomalies can be searched for:

1. **Point anomalies** – when a particular data point deviates from the remaining or from the normal. These are the simplest type of anomaly and are the focus of most research applying anomaly detection techniques [57]. A practical example of this type of anomaly is, for example, when in transactions associated with a credit card, there are cases in which the amount spent is higher than the normal spending range (Figure 12).

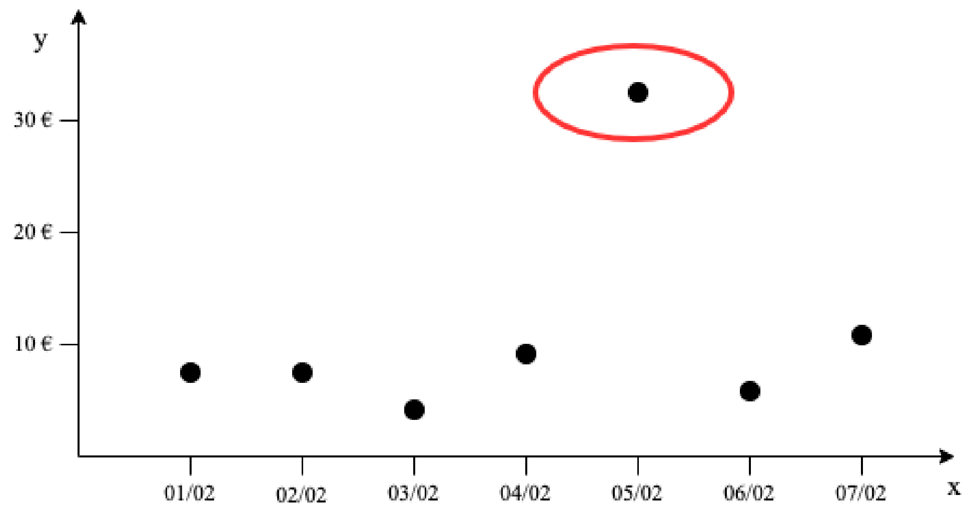


Figure 12 - Example of a point anomaly

2. **Contextual anomalies** – when data points are considered anomalous in a given context, but eventually not in others [57]. In this type of anomaly, it is necessary to consider the contextual attribute of the instance where it occurs. For instance, a contextual attribute in credit card fraud would be the time of purchase. When analyzing a person's monthly expenses throughout the year, it can be considered normal for this figure to be higher in the month of December, since it's during this month that festivities such as Christmas and New Year's Eve are celebrated. If, on the other hand, this were to occur, for example, in the month of March, this would already be considered a contextual anomaly because it does not conform to the typical behavior in the context of time (Figure 13).

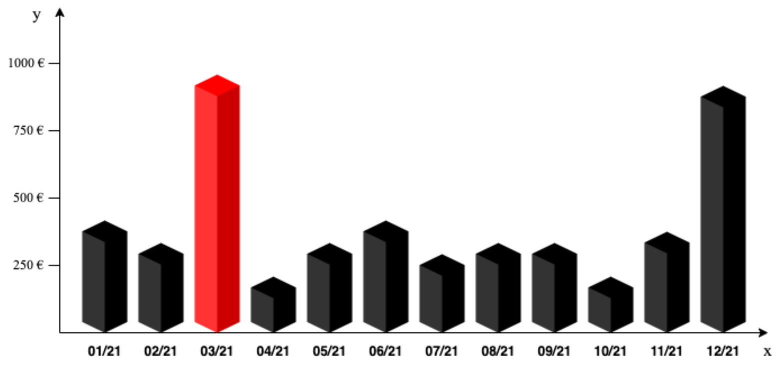


Figure 13 - Example of a contextual anomaly

3. **Collective anomalies** – when whole groups of data points deviate. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous. This type of anomaly can only be found in data sets where there is a relation between data instances [57]. One of the main cases in which this type of anomaly can be verified is by analyzing the results of a human electrocardiogram output. The red region in Figure 14 represents a set of low values. Now, these values, by themselves, do not represent an anomaly. The problem is that the same low value exists for an abnormally long time, resulting in an anomaly.

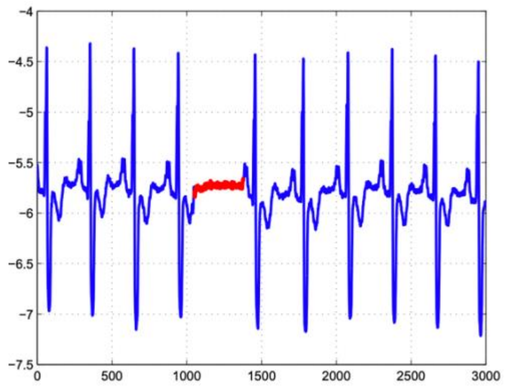


Figure 14 - Example of a collective anomaly (Source: [57])

Nonetheless, the use of anomaly-based approaches for fraud detection has some known challenges. First and foremost, an anomaly or an outlier is not necessarily representative of fraud, so there's the risk of identifying false positives, depending on the sensitivity of the method used. Moreover, noise might also be difficult to distinguish from actual fraud instances. Finally, fraud patterns change over time so what was once fraud may no longer be, and vice-versa. For these reasons, the search for anomalies has become an increasingly important aspect of data mining [58], especially considering the era of Big Data and the resulting huge volumes of data from which valuable information needs to be extracted. Thus, several anomaly detection techniques have been studied in different application scenarios. These can be divided into three types – supervised anomaly detection, semi-supervised anomaly detection and unsupervised anomaly detection – whose use is dependent on whether or not there is labeled data. The label associated with a given data instance classifies it as normal or anomalous. Obtaining a labeled dataset implies a great effort in terms of both resources and money since the labelling process is usually performed by a human. Furthermore, obtaining a dataset that contains both normal and anomalous data instances that are representative of all types of fraudulent behaviors can be quite difficult, especially when considering the fact that anomalous behavior if often dynamic in nature, with new types of anomalies potentially appearing at any time, for which there is no labeled data. Unsupervised anomaly detection techniques are thus most widely used, with numerous applications [59].

Unsupervised anomaly detection methods are used when there are no prior data representative of legitimate and fraudulent behaviors. The goal is to find hidden structures and patterns among the unlabeled data. Among these techniques, clustering-based techniques stand out. The concept behind these techniques is quite simple – data instances with similar attributes are grouped into clusters. New instances are then tested against the resulting model to check if they fit into any of the previously defined clusters. These clustering-based techniques can be divided into three different categories [59], which differ in their operation and interpretation of results:

- Category 1: This category of techniques assumes that all normal instances belong to some cluster while anomalous instances belong to none.
- Category 2: The second category assumes that all normal instances are close to the center of their closest cluster, while anomalies are far from the center of their closest cluster. These techniques are the most applied in the areas of fraud detection, intrusion detection and fault detection [59].
- Category 3: Finally, the third category of techniques assumes that all instances belong to clusters – normal instances are grouped into large, dense clusters, while anomalous instances are grouped into smaller, more dispersed clusters.

There are several anomaly detection related applications of clustering-based techniques. In [60] , the authors review different clustering techniques that can be used in financial fraud detection. Another example is the application of clustering techniques such as SAS EM and CLUTO for health insurance fraud detection [61] . Despite their wide applicability, much due to the fact that they are unsupervised anomaly detection techniques and therefore do not require a labeled dataset, clustering-based techniques also have some drawbacks. Some algorithms force the association of each instance in the training set to a cluster, which may lead to the incorrect classification of an anomalous instance, as it could be assigned to a large cluster, and consequently treated as normal. Moreover, these techniques are not quite efficient when the anomalous instances form large clusters, as it becomes hard to distinguish them from normal behavior [62].

When a labeled dataset is available, supervised anomaly detection techniques can be used. These techniques use the labeled data to train a model, in order to setup a normal class and an anomalous one. New data instances are then tested with the resulting predictive model and are assigned to one of those classes. Classification-based techniques are the most popular of supervised anomaly detection. These can be either multi-class or single class categories. In the former case, the model contains multiple normal classes, and a classifier is used to differentiate between each of those classes against the other. If the new instance to be tested is not found to belong to any of these normal classes, it is considered anomalous. On the other hand, single-class technique classifies all the normal instances into one normal class, and therefore any test instance that does not fall into that region is considered anomalous [59]. Some of the popular classification techniques used in anomaly detection are the Neural Networks and Bayesian Networks, in this example [63] used in the context of credit card fraud detection. In fact, the area of credit card fraud detection is one for which there are more applications related to anomaly detection [64] – suspicious transactions are compared to that customer’s profile and if they don’t follow the pattern, they are considered as anomalous. Other techniques used in this area are the Decision Trees and Support Vector Machine [65]. Furthermore, Rule-based techniques can also be used, in areas such as the detection of anomalous phone calls [66] or the detection of phishing attacks [67]. As mentioned before, the main disadvantage of this type of technique is that it is extremely difficult to obtain not only a labeled dataset, but also one that is accurately labeled.

### 2.2.1. Challenges

Regardless of the problem area and the technique used, there are some general challenges associated with the topic of fraud detection.

#### 2.2.1.1. Concept Drift

Up until recently, ML problems were seen as static: a model would be trained for a given dataset, it would be deployed, and it would stay in production for relatively long periods of time. Nowadays the paradigm changed, and ML is seen as highly dynamic: data arrive in streaming, changing over time, and models need to adapt continuously. Asides from the evident challenges of adapting models, there’s also the risk that the properties of the data change over time. This includes not only the distribution of its variables but also the relationship between the dependent and independent variables. Different types of concept drift have been identified [68], depending on characteristics like drift subject, frequency, duration, transition, recurrence or magnitude. The challenges of learning under concept drift are multiple and include learning continuously and fast enough for models not to be outdated and being able to distinguish noise from signal [29]. Concept drift is frequently observed in fraud detection problems since fraud patterns change over time, as perpetrators continuously adapt avoid being caught.

#### 2.2.1.2. Big Data

Fraud detection datasets generally share a characteristic with other classes of ML problems: they tend to be very large datasets. They often also share other characteristics of the so-called Big Data [69]: data are generated in real-time, data may be very diverse, there may be missing data and the value of each instance may be low. These characteristics call for specific algorithms and methods that are able to deal with the complexity and specificity of these datasets [70]. One of the first solutions for this problem was to parallelize and distribute learning tasks, just like storage has already been distributed. In distributed learning [71], datasets are usually scattered across the nodes of a cluster, and learning tasks are also distributed and take place in parallel, taking into consideration the load of the cluster or principles such as data locality. Other authors focused on reducing the computational complexity of the task, by avoiding the training of models altogether, and either adapting existing models over time or predicting whether or not training a new model will result in better predictive performance [72].

Another common approach is to learn meaningful and useful representations of variables, allowing to reduce the complexity of the problem in terms of number of variables. This is achieved by the so-called Representation Learning [73], and other variable reduction techniques. Better learning performance in Big Data problems can also be achieved through Transfer Learning [74]. Here, the key idea is to transfer previous knowledge between a source and a destination domain, in order to avoid training models from scratch. Other approaches, such as AL and OL have also been used to deal with the challenge of learning from Big Data.

#### 2.2.1.3. Bias

Bias in data may have very different sources (e.g., selection bias, lack of meaningful variables), but it can generally be defined as data that is not representative of the phenomenon being studied. As a result, data will often encode a prejudice, evident or not, against a part of the phenomenon (e.g., a particular group of people). The main consequence is that ML models will also be biased and, thus, discriminatory. Contrary to other domains, bias in fraud detection does not result from any form of prejudice: data are naturally biased since fraudulent instances are, fortunately, far less frequent than non-fraudulent ones (the ratio of imbalance can be as severe as 0.062 [75]). If bias in the dataset is not properly dealt with, any ML model will most likely predict every instance it is given as not fraudulent. Most approaches to deal with class imbalance revolve around variations or combinations of undersampling (removing instances of the over-represented classes) or oversampling (replicating the under-represented instances), in which the instances to over or under-represent are selected randomly. Other more complex techniques exist, such as SMOTE [76]  or ADASYN [77].

#### 2.2.1.4. Prediction Errors

When the problem of fraud detection is treated as a binomial classification one (fraud or not fraud), there are four possible outcomes: two are correct (true positive and true negative), and the other two are prediction errors (false positive and false negative, also known as Type I and Type II errors, respectively). A false positive happens when an instance, flagged by the model as fraud, is actually not fraud. Conversely, a false negative happens when an actual instance of fraud is missed by the model. In this domain, as in many others, errors have a different cost. The cost of a false positive will be given by the impact of a bank denying a transaction or an insurance to a prospective (innocent) client, or by the cost of triggering an audit to a company that did nothing wrong after all. The cost of a false negative may be estimated by the social and economic impact of letting a perpetrator get away with tax evasion, for instance. In fraud detection, false positives are a major challenge. Several approaches can be adopted to deal with it. In [78] the authors combine a deductive approach (red flag) with process mining to counteract the deficits inherent to purely inductive fraud detection approaches. Other authors have looked into variables engineering approaches to address this issue. In [79], the authors rely on a Deep Feature Synthesis algorithm to automatically derive behavioral variables based on the historical data of the card associated with a transaction. Specifically, they generate 237 variables for each transaction, and use a random forest to learn a classifier. The authors evidence a decrease of 54% in the number of false positives.

# 3. Proposed Solution

As already mentioned, the solution implemented was based on the practical case of the Col.bi software, developed by Petapilot. Thus, it becomes necessary to briefly describe the general operation of this software.

First of all, the source of information of this software are SAF-T files. SAF-T, which stands for *Standard Audit File* for *Tax* purposes is a standard maintained by the OECD that is used to exchange electronic accounting data between organizations and tax authorities or external auditors for audit and compliance purposes. This standard was first adopted in Portugal in 2008 and has now been spreading to other EU countries.

As soon as a SAF-T file is loaded into the software, it creates a new project associated with that file where, besides having all the information contained in it in a simple and intuitive way for analysis and exploration, a set of digital audit tests are performed, with the purpose of checking the compliance of the file with the SAF-T and with OECD's audit guidelines. From these tests result the so-called findings, which represent non-conformities or potential non-conformities that are found either in the structure or in the content of the file. These can range from syntax errors in the SAF-T to potential fraud positives. At the end of this process, an auditor is presented with a list of all the resulting findings, so that he can analyze them.

The main disadvantage of this software is that the audit tests performed are based on predefined rules, many of them consisting of relational algebra operations. For instance, a given rule may fire if the sums of the values of two columns, which should be equal, do not match. However, upon further review, the auditor concludes that the difference is minimal and due to a known rounding error in a given ERP. Considering that the contents of a file usually consist of billions of transactions, it is to be expected that this rule will, most likely, fire hundreds or thousands of times throughout the SAF-T file. In any case, the auditor will have to spend time analyzing these instances and determine that they are false positives. Assuming the finite capacity of the auditors and considering the fact that it is humanly impossible to analyze and validate every one of the thousands of findings resulting from the tests, these analyses are currently done using sampling techniques, which in itself limits their extent and allows potential cases of fraud not to be identified by the auditor during this process. With this in mind, it can be said that there is a lack of adequate support tools that are able to identify and infer the quality of truly positive instances, especially when considering the increasing complexity of financial systems, technology-supported fraud schemes and the standards imposed by regulators to combat them. The implemented solution, which will be described in detail later, seeks to address this problem by complementing the existing software in order to facilitate the auditors' work, through a system capable of learning from the auditors' actions and knowledge, in order to become increasingly accurate in identifying the truly positive instances. This will allow auditors to focus on the really relevant cases, since the repetitive ones will be dealt with by the system.

An overview of the project can be seen in the figure below (Figure 15).

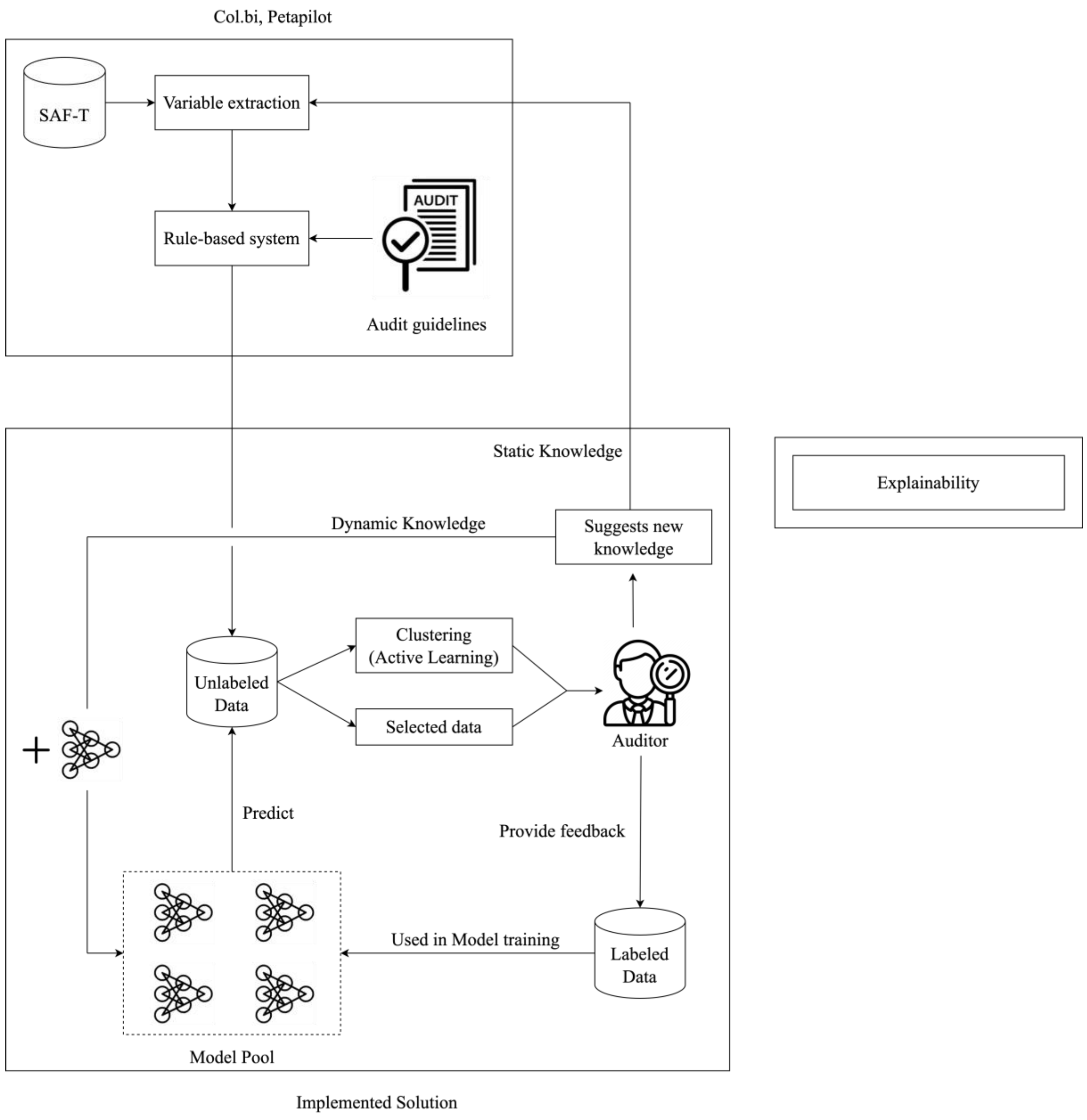


Figure 15 - General scheme of the implemented solution

As mentioned before, the implemented solution complements the existing software (Col.bi), as it works on its results (findings). That is, the general operation of the Petapilot software remains unchanged, but instead of the list of findings resulting from the audit tests being presented directly to the auditors, they now go first to the implemented ML system, where they are stored in the "Unlabeled Data" database. This contains the instances that have not yet been reviewed by an auditor. As soon as instances arrive in this database, the implemented ML system starts working on them. This is because in the model pool there is always one or more models simultaneously. The most important one, and the only one that always exists by default, is responsible for predicting and assigning a potential fraud level to each of the instances present in the "Unlabeled Data" database, and that therefore have not yet been analyzed by an auditor. This is a numeric variable that can take values between 0 and 9, where 0 represents a high confidence on the part of the ML system that the instance is a false positive and therefore shows no evidence of fraud, while the value 9 represents a high confidence on the part of the system that the instance is a true positive and therefore could potentially be fraudulent. At any time, an auditor can work on this data, using a search and query mechanism, selecting a set of instances (findings) to analyze, being able to focus on those with a higher level of potential fraud, and providing feedback. This instance selection by the auditor may be influenced with the goal of leading the auditor into providing feedback on instances that would provide the system with more knowledge (e.g. instances that are "more different" from the already labeled ones). This process is described in a later section.

The analyzed and consequently newly labeled instances are then moved to the "Labeled Data" database. This contains all the instances that have already been analyzed by an auditor and are used for training the models.

The auditor can also contribute with knowledge by suggesting new variables to be included in the pipeline. This can happen in two ways, depending on the type of variable suggested:

- The auditor can suggest a variable that is present in the SAF-T file but is not being used and therefore not being extracted during the variable extraction process. In these cases, the solution is quite simple: the variable in question starts to be extracted and consequently to be considered. These variables, which are possible to extract directly from the dataset, are called **static variables**.
- However, most of the time, the new knowledge suggested by the auditor consists of annotations or appreciations that are meaningful to an auditor, and make sense in the domain, but whose subjectivity may make it impossible to extract from the original dataset. It is a fact that experienced auditors have their own mental shortcuts: values, value transformation, or combinations of values that are seen as good indicators of potential fraud. Thus, in these cases a new model is added to the model pool. This model analyzes existing patterns and, based on the rest of the information associated with a given instance, makes predictions for the value of the new variable proposed by the auditors. In this way, the auditors then only have to validate these predictions. These variables, which are not possible to extract directly from the dataset, are called **dynamic variables**.

This means that multiple models are maintained in the system: the main one, that predicts the likeliness of fraud of an instance, and one model for each dynamic variable in the pipeline. Since the labeled data changes throughout time, as the auditors do their work, these models must be updated regularly.

Finally, the system also includes an explainability module, developed by another colleague. The main goal of this module is to provide auditors with an intelligible explanation for each prediction done by the system. That is, for every variable whose value has been predicted by a model and not yet validated by an auditor, there is an accompanying explanation that is symbolic in the sense that it uses the domain´s concept and relies on simple and familiar statistics concepts. The explanation is designed to provide the auditor with a degree of confidence on the prediction provided by the system.

The core of the project was developed and implemented using Apache Airflow, an open-source tool that allows the scheduling and orchestration of data pipelines or workflows. In a simpler way, it essentially allows the automation of processes, without the need for human intervention. In addition, MySQL relational databases are used for data storage and all model management is done through H2O, a fully open source, distributed in-memory ML platform with linear scalability. Finally, the entire developed project was containerized through the use of Docker, an open-source platform that allows to build, deploy, run, update and manage containers – a standard unit of software that packages up code and all its dependencies, so the application runs swiftly and reliably from one computing environment to another. This resulted in 5 containers that communicate with each other through a virtual network (Figure 16).

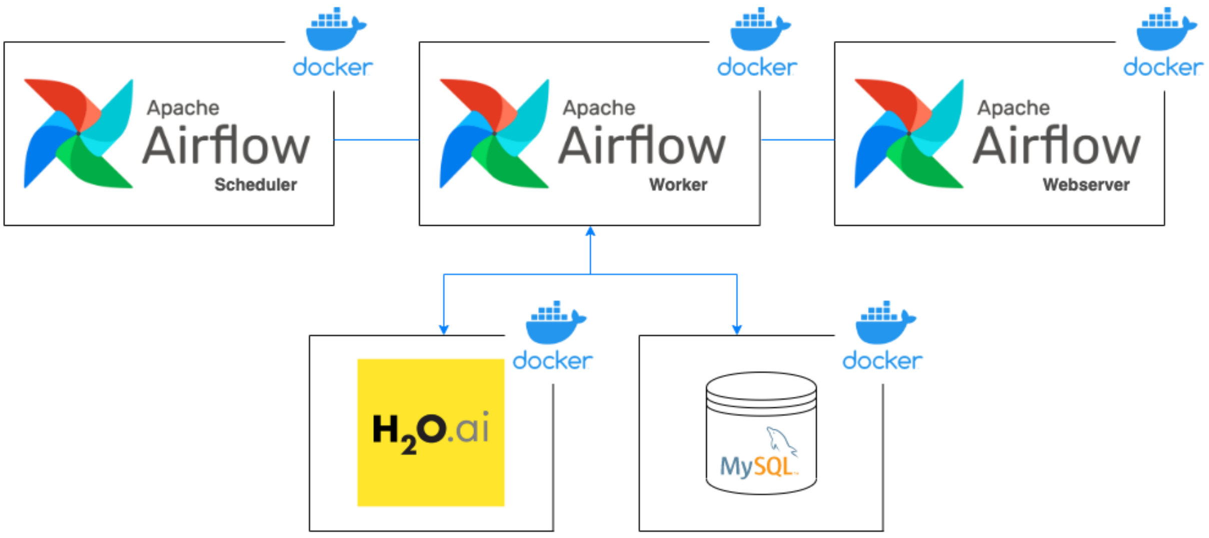


Figure 16 - Existing technologies and docker containers

## 3.1. Variable Management

There are two ways in which the auditor can provide feedback. The first is by proposing new variables to be added to the data extraction pipeline.

Auditors can create new variables, or change or delete existing ones, and these changes are incorporated into the system in real-time. This means that other auditors can also start using the proposed variables in their own audits if they find them useful.

As detailed before, the system considers two types of ML variables, deemed **static** and **dynamic**. Static variables are closer to traditional ML variables in the sense that they can be added to the variable extraction pipeline, and their values calculated or extracted from the SAF-T raw data. Dynamic variables, on the other hand, are used to represent higher-level or abstract concepts, which have meaning for an auditor, but that cannot be directly computed from the raw data. For instance, an auditor may suggest a new variable “fraud risk”, with the values “high”, “medium” or “low”. To provide a value for a given instance, the auditor may look simultaneously at different indicators or may use his intuition or past knowledge regarding the specific company being audited, or the ERP being used.

Auditors have a specific UI for managing variables, which allows them to create a new variable and provide its general information (e.g. name, description, type) as well as its specifics: its different possible values in the case of an enumeration, or its range of values in the case of a numeric variable.

Due to these variables representing rather abstract concepts, they cannot be directly computed from the input raw data. Nonetheless, we still want the system to make predictions for these variables so that they can be automatically filled, otherwise these variables would be mostly empty in the dataset, especially at the beginning. To this end, the system maintains a model in the pipeline for each dynamic variable. The first model for a variable is trained when a minimum number of validated instances exist for which that variable has data. When training this model, the variable it is associated with is the dependent variable, and the remaining variables of the dataset are the independent ones.

The implementation of this functionality requires an unconventional representation of the data. Indeed, in ML projects, data are usually represented in a tabular fashion, in a single table. However, allowing for variables to be added or removed in real-time would imply that the columns of this table would have to change, which is impossible. To address this issue, the data is stored in a normalized set of tables.

A simplified version of the database schema that supports this is provided in Figure 17. The table *Variable* stores the general information for each dynamic variable, including name, description, owner, last edit data, among others. The table *VariableValue* stores the different values that each variable may hold (in the case of an enumeration) or the boundaries of the variable on the case of numerical variables (as multiple ranges).
There is a 1:N relationship between *Variable* and *VariableValue*.

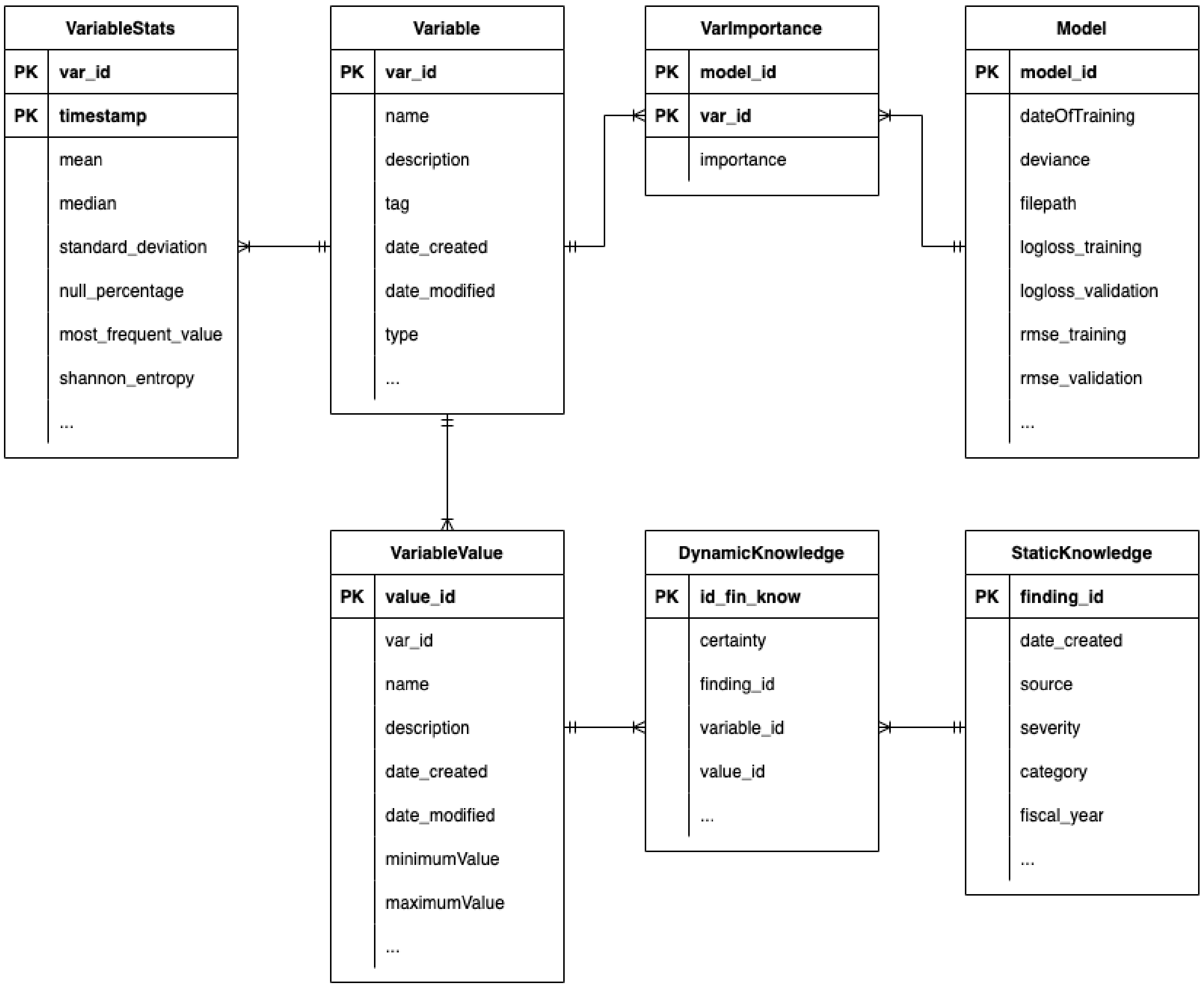
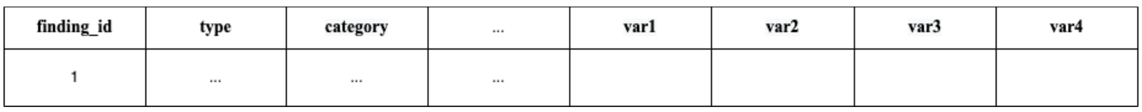


Figure 17 - Simplified version of the database schema

The table *Model* is where the performance metrics of all trained models are stored. This allows for the performance of the system to be monitored over time. The table *VarImportance* connects the former two tables by storing information regarding the relative importance of each variable in each model. Among other aspects, this allows to understand which of the dynamic variables proposed by the auditors are statistically relevant, and which are not.

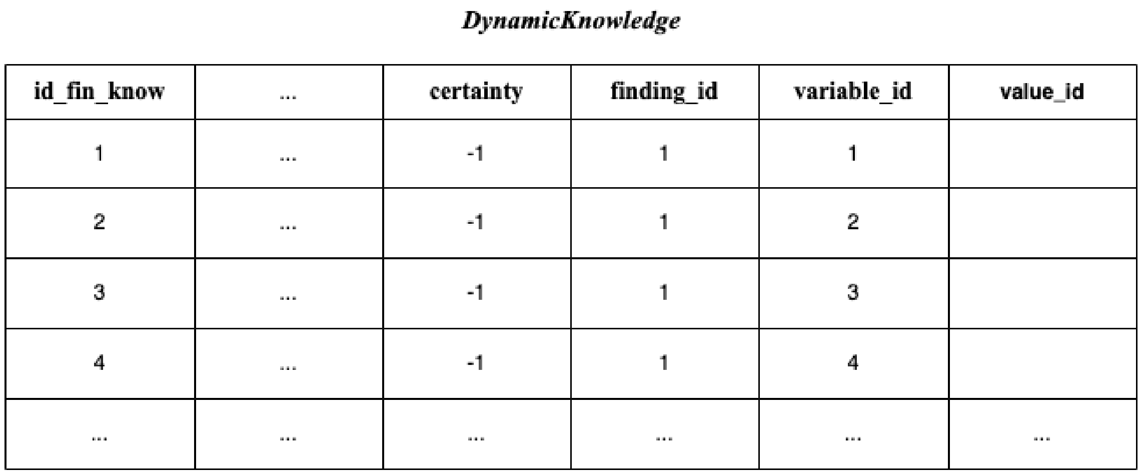
The table *StaticKnowledge* can be seen as the traditional "main" dataset: it holds the data describing each instance under audit, including all the static variables (those extracted from the SAF-T). There is a 1:N relationship between this table and the *DynamicKnowledge* table, which holds information about the dynamic variables. Traditionally, these two tables would be a single one, adding the dynamic variables to the rest of the information of each instance (Figure 18).



| finding_id | type | category | ... | var1 | var2 | var3 | var4 |
|---|---|---|---|---|---|---|---|
| 1 | ... | ... | ... | | | | |

*Figure 18* – representation of an instance if the information regarding dynamic variables was not stored in a separate table

This, however, would imply that the four dynamic variables represented in the example above (var1, var2, var3 and var4) already existed at the beginning of the system's use, and it would also not be possible to add new variables or remove existing ones. Thus, the information regarding the dynamic variables is then stored in the *DynamicKnowledge* table, which has, for each instance, a row for each dynamic variable that exists in the system (Figure 19).



***DynamicKnowledge***

| id_fin_know | ... | certainty | finding_id | variable_id | value_id |
|---|---|---|---|---|---|
| 1 | ... | -1 | 1 | 1 | |
| 2 | ... | -1 | 1 | 2 | |
| 3 | ... | -1 | 1 | 3 | |
| 4 | ... | -1 | 1 | 4 | |
| ... | ... | ... | ... | ... | ... |

Figure 19 - *DynamicKnowledge* table structure

This ultimately allows for the number of variables to change over time, by adding/removing them in this table.

Another relevant aspect is that each dynamic variable of each instance may be in one of several states. The state of a given dynamic variable can be identified by the value present in the corresponding 'certainty' column:

- -1 – the default value. Value assigned when the variable is created and does not yet have an assigned value, either manually or automatically.
- Predicted value – when the system predicts the value of a variable, its certainty value becomes the predicted value. In the case of non-numeric variables, a default value of 1.5 is assigned.
- 2 – when a human expert analyzes and validates the prediction made by the system, the value of the certainty of the corresponding variable is changed to 2.

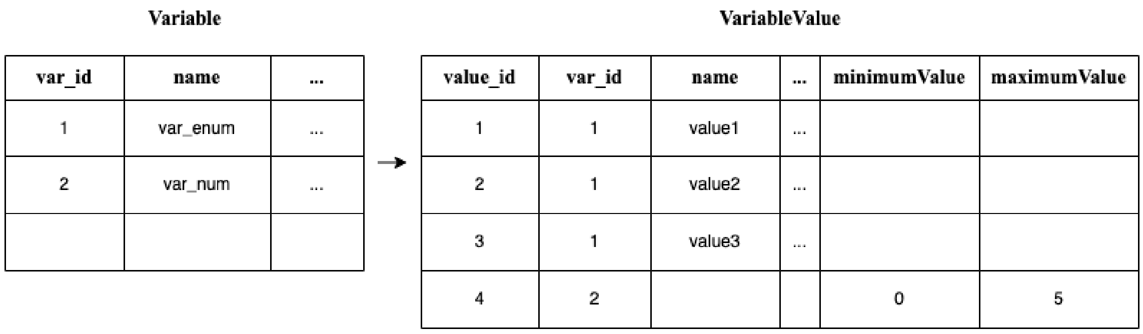
It is normal for a variable to pass through all states during its existence in the system.

When the variable is in one of the first two states, it is considered to be part of the unlabeled dataset. When it is in the validated state, it is considered to be part of the labeled dataset. Only this last group of data is used by the system to train models. The remaining data are filled in by the ML models as soon as they are available, to be later validated by the auditors.

From this it also results that a given instance may be in three different states: not validated (when the auditor has not audited it yet), validated (when all variables of the instance are validated) , or partially validated (when an auditor has validated this instance in the past, but in the meantime a new variable has been added).

It is important to note that in the *VariableValue* table, the values of its primary key, *value_id*, are all negative, in order to avoid conflicts when generating the views that constitute the datasets used for both model training and predictions. This is because while in a non-numeric variable each value has a unique identifier, which facilitates access to it, the same is not true for a numeric variable, where there is a range of values all associated with the same identifier. Thus, in these cases what matters to store in the *value_id* field of the *DynamicKnowledge* table is the actual value of the variable, whether it was predicted by the system or determined by the auditor, rather than the unique identifier of the value, as happens with other variables. The problem is if the actual value of a numeric variable is the same as the unique identifier of some other value that already exists.

Consider, for example, the existence of two dynamic variables in the system: a non-numeric variable, “var_enum”, for which there are 3 values whose identifiers range from 1 to 3, and a numeric variable, “var_num”, that has only one value, representing a range of values (Figure 20).



**Variable**

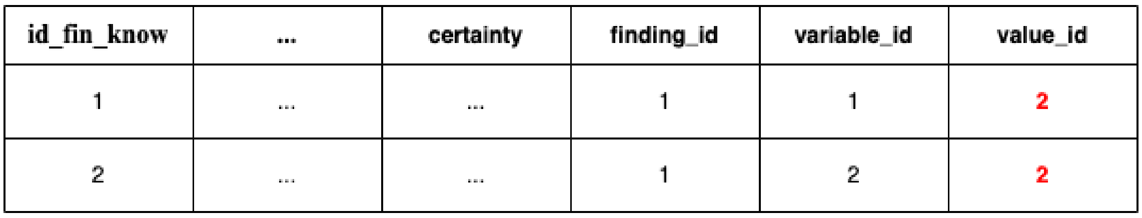
| var_id | name | ... |
|---|---|---|
| 1 | var_enum | ... |
| 2 | var_num | ... |
| | | |

**VariableValue**

| value_id | var_id | name | ... | minimumValue | maximumValue |
|---|---|---|---|---|---|
| 1 | 1 | value1 | ... | | |
| 2 | 1 | value2 | ... | | |
| 3 | 1 | value3 | ... | | |
| 4 | 2 | | | 0 | 5 |

Figure 20 - Example of existing variables and their values

Being dynamic variables, their value will be, at some point, predicted by the system and later validated by an auditor, being updated in the *DynamicKnowledge* table in the context of each instance. In this case, consider the instance (finding) with the identifier number 1. In the case of the non-numeric variable, “var_enum”, if the value predicted by the system or validated by the auditor is “value2”, the respective row in the *DynamicKnowledge* table is updated with the predicted variable identifier, i.e. 2. However, in the case of numeric variables, there is not an identifier for each value but a range of values, so if the prediction made by the system or the value validated by the auditor is 2, this is the value that is stored in the *DynamicKnowledge* table (Figure 21).



| id_fin_know | ... | certainty | finding_id | variable_id | value_id |
|---|---|---|---|---|---|
| 1 | ... | ... | 1 | 1 | 2 |
| 2 | ... | ... | 1 | 2 | 2 |

Figure 21 - Possible conflict regarding the identifier of the values inserted into the DynamicKnowledge table

This would create a conflict since, for the system, two different variables would be assuming the same value. When running the *views* responsible for generating the datasets, which are mainly based on the information contained in the *DynamicKnowledge* table, these two values would be considered as the same unique identifier of a certain value, and in the resulting dataset the numeric value would take the value whose unique identifier is 2, instead of literally taking the value 2. Thus, by defining the unique identifiers of the variable values as negative, it becomes possible to distinguish them from the actual numeric variable values.

Summarizing, the adopted data model allows for auditors to define and propose their own intended variables, which are used side-by-side with the "traditional" variables extracted from the SAF-T files. To fill-in the values for the proposed variables, a ML model is automatically added to the pipeline for each dynamic variable, which learns from the auditor's feedback to become increasingly better at labeling the instances with the user-proposed variables.

## 3.2. Guided Instance Labelling and User Feedback

This work is built around the idea that it is valuable to incorporate human agency (e.g. experience, sense of meaning, goal-oriented behavior, domain-knowledge) into the AI life-cycle. However, in order for the benefits of human action to be maximized, the actions of human experts should be supervised by the system as much as humans supervise the system. In other words, there must be a close cooperation between humans and AI.

In this project, this is realized by providing the system with some degree of control over which instances of data the human expert is going to validate/label next. Indeed, and as shown in previous work [80], the order by which instances are labeled does influence the rate at which the system learns. It may so happen that an auditor is more familiarized with a given type of cases, or for some reason they are easier to audit (e.g. past experience, domain), and may focus more on such cases. In these cases, the time of the auditor may be under-utilized from a ML perspective, as the instances being labeled are very similar to already labeled ones.

The proposed system is able to, given the current pool of labeled data, point out instances that would contribute with more knowledge, if labeled. The goal is to ensure that the learning speed is maximized. In other words, we want to minimize the gap between the model's performance at any point in time, and the performance if the model were trained with the final set of data. At the same time, it is also a goal to minimize uncertainty and fluctuations in the model’s performance, which are frequent in scenarios of streaming data such as this. Indeed, in such cases, early models are often regarded as good (often due to an unrepresentative dataset), only to later see their performance decrease as larger datasets are used.

To achieve this, an unsupervised learning step was added to the typical AL pipeline (Figure 22), as follows. The underlying assumption behind the use of a clustering algorithm is that labeling instances that belong to a cluster with a significant number of already validated instances will contribute with fewer knowledge than labeling instances of clusters that have fewer validated instances. Thus, clustering is used to determine how many groups or types of instances exist, and then we look at the percentage of validated instances in each group to guide the process.

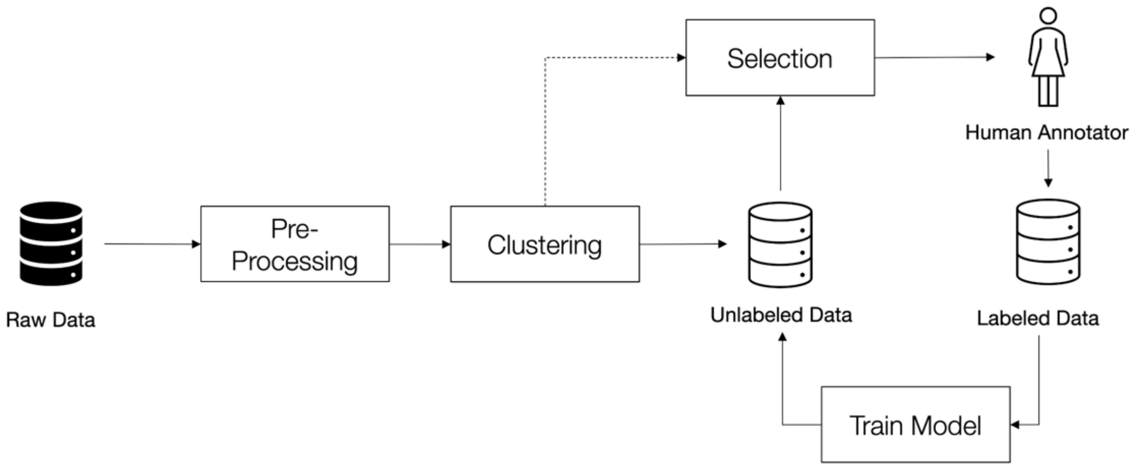


Figure 22 - Active Learning implementation, using clustering

Thus, while the auditor can always decide autonomously which instances to label, there is an “assistant” that suggests the next instances to audit. The underlying goal of this assistant is to minimize the difference between the percentage of labeled instances in each cluster.

To implement it, a batch approach was followed: groups of instances of a given size (the batch size) are selected and suggested to the auditor, who may label some, all, or none of the selected ones. This is done to provide the auditor with a sense of control, in the sense that he can still choose from among a group of instances which to audit. However, the instances presented to him are selected with the previously mentioned goal. Other selection schemes exist, such as streaming (in which one instance is selected at a time) or synthetic (in which made up instances are used). The former was not used as it could be rejected by the human expert due to a sense of lack of control, and the latter does not apply in this case since real data is being used.

The process is thus cyclic and iterative: a batch of instances is selected from the unlabeled data pool, the human expert labels them, and a new model is trained with all the labeled data. As data is moved from the unlabeled to the labeled dataset, the percentage of labeled instances in each cluster is updated, to be used in the next run of the selection function.

Thus, in each iteration, the auditor will be presented with a batch of *x* instances to choose from. However, the instances will not be equally distributed (according to the clusters identified): they will be selected, randomly, according to the proportion of instances in each cluster that have already been labeled. Thus, independently of the starting state of the system (e.g. some data may have already been labeled), it will quickly tend to have a balanced representation of instances according to their clusters.

There are currently several approaches to AL in the literature [81]. In the so-called uncertainty sampling, arguably the most popular approach to AL, the model is queried to find the instances it is more uncertain of, where uncertainty is defined by a measure of entropy or confidence. The way this is actually implemented depends on the type of ML problem: regression, multiclass classification or binary classification. Another more expensive approach, deemed "by committee", takes the predictions of several models and will select for labelling those instances in which the models disagree the most. Finally, another alternative is to select instances that are going to change the model the most if their labels are known, in which the measure of change is error reduction. This alternative is called "expected model change". When compared with these approaches, this proposal constitutes a relatively simpler and less computationally intensive alternative.

## 3.3. Model Management

As mentioned before, in the proposed system there is a model pool where there's always at least one model, responsible for predicting the potential fraud level of a given instance. To this, an undetermined number of new models may be added during the use of the system, one for each new variable suggested by the users. With this in mind, the way these models are managed is of high importance, even more so considering that IML systems have particular characteristics of their own when compared to traditional, one-shot ones. Specifically, there are challenges that stem from the dynamic nature of the data: when the data change, so must, in principle, the models. Thus arises the question as to **when** and **how** to update models.

There are also trade-offs to take into consideration. For instance, a more dynamic approach in which models are updated frequently (e.g. with minor changes in the data) will react more rapidly to change, but it will also be more resource-intensive and more sensitive to noise. Doing the opposite, i.e., having a longer time between updates, will require less computational resources but result in a system that might be operating with under-performing models in the sense that they may be partly outdated.

As already mentioned, the entire process of model training and predictions is managed using Apache Airflow. In this tool, workflows are represented through Directed Acyclic Graphs (DAGs), the core concept of Airflow. A DAG consists of a python file where a set of tasks are declared, organized with dependencies and relationships to define how they should run. The execution of a DAG can be triggered manually or automatically by using the airflow scheduler. In this case, cron expressions can be used in order to define when the DAG should run (on a certain day and at a certain time, every hour, daily, weekly, etc.).

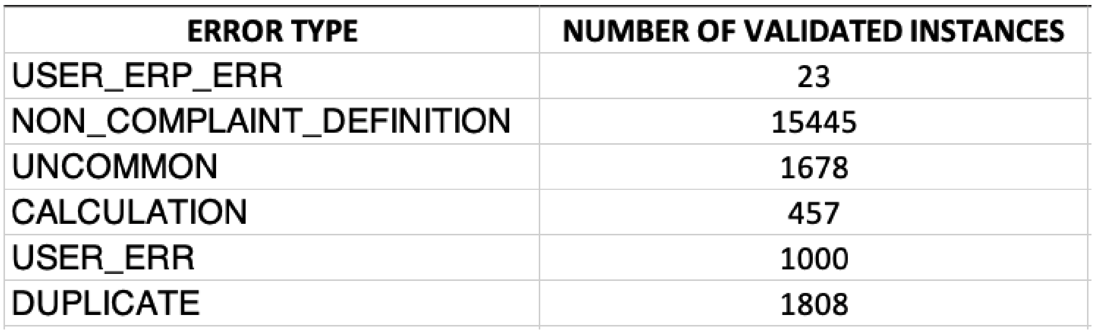
Thus, in the context of this project two DAGs were developed, one responsible for model training and the other for prediction making.

In the training DAG three tasks are defined:

- The 1st one is responsible for checking which variables exist in the system, and of those, for which a model can be trained. This is because to be able to train a model for a variable, there must be a minimum number of validated instances containing that variable. Once this task is finished, the DAG enters a loop in which the 2nd and 3rd tasks are executed for each of the variables found in this 1st task.
- The 2nd task starts by obtaining the training dataset. In the case of non-numerical variables, this dataset goes through a balancing process. This is because there may be certain classes with a higher number of validated instances than others, making the dataset imbalanced. Training a model with an imbalanced dataset would make it biased towards the majority class.

  The approach used to address this problem was to randomly resample the dataset. The two main approaches to randomly resampling an imbalanced dataset are to delete examples from the majority class, called **undersampling**, and to duplicate examples from the minority class, called **oversampling**.

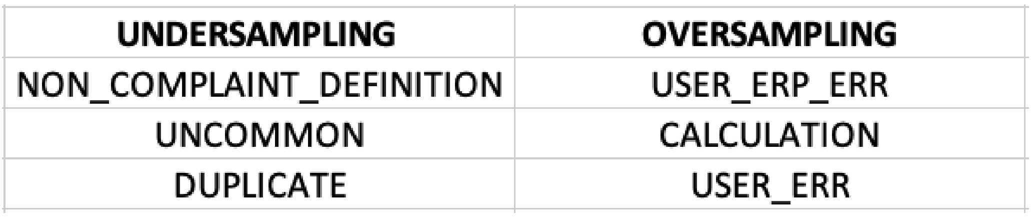
  So, the process starts with a count of validated instances for each class or value of the variable. In order to facilitate the understanding of the process, the balancing of the training dataset of the variable *Error Type* is presented as an example (Figure 23).



| ERROR TYPE | NUMBER OF VALIDATED INSTANCES |
|---|---|
| USER_ERP_ERR | 23 |
| NON_COMPLAINT_DEFINITION | 15445 |
| UNCOMMON | 1678 |
| CALCULATION | 457 |
| USER_ERR | 1000 |
| DUPLICATE | 1808 |

Figure 23 - Number of instances of each value of the *Error Type* variable

As can be seen in the figure above, there is a clear discrepancy between the number of validated instances of each of the variable values, making its dataset quite unbalanced. In these cases, the median of the number of validated instances of each variable value is calculated in order to find and set a target value of instances (in this case, 1339). Once this target value is set, the classes or values of the variable are then split into two groups (Figure 24): values whose number of validated instances is less than the target number of instances go into the group where the oversampling approach will be followed. In turn, the values whose number of validated instances is higher than the target number of instances go to the group where the undersampling approach will be followed.



| UNDERSAMPLING | OVERSAMPLING |
|---|---|
| NON_COMPLAINT_DEFINITION | USER_ERP_ERR |
| UNCOMMON | CALCULATION |
| DUPLICATE | USER_ERR |

Figure 24 - Division of the variable values between *undersampling* and oversampling

At the end of this process, the expected is to obtain a completely balanced dataset, in which each value of the variable in question has the same exact number of validated instances, which will be used to train the model.

Once the dataset is balanced, it is then ready to be used to train a model. Model training is managed through a fully open source, distributed in-memory machine learning platform with linear scalability, H2O. In this way, the balanced dataset is first transformed into an H2OFrame, the primary data store for H2O. In other words, the dataset is loaded into an H2O cluster.

At this stage, the dataset goes through a factorization process, in which all columns whose contents are strings are converted to categorical. This step is necessary since H2O would ignore all variables whose value are strings when training a model and would not attribute any importance to them.

The dataset is then ready to be used for training the model. The ML algorithm used for this is the Random Forest algorithm, with 20 trees. The choice of this algorithm was due to the fact that it is a distributed algorithm (capable of training simultaneously on multiple nodes in a cluster) and it is robust in overfitting scenarios. Furthermore, it also facilitates the process of explainability, which, as already mentioned, is one of the modules implemented in this project.

The 2nd task ends with storing the performance metrics of the trained model in the database.

- The 3rd and last task of this DAG is to define the variable's active model, that is, the one that will be used to make predictions. All the models of a variable are stored in the database, so it becomes necessary to define which one to use to make predictions, marking it as the active model. Thus, this task goes through all the existing models of the variable and checks which one is the most recent and sets it as the active model. Although this was criterion chosen for selecting the active model, others could also have been chosen, such as analyzing a particular error metric and choosing the model which minimizes it.

The prediction DAG is composed of two tasks:

- The 1st task is responsible for identifying all the variables for which the conditions exist to make predictions. To this end, it begins by identifying all variables for which there is at least one active model associated. Then, among these, it is necessary to verify for which there are non-validated data. It only makes sense to make predictions on data whose value is not known or not yet validated. If all existing instances of the variable are already validated, there is no data on which it is necessary to perform predictions.
  This DAG also enters a loop at the end of the 1st task, in which the 2nd task is executed for each of the identified variables.
- The 2nd task is where the predictions are made for the variable in question. To do so, the variable's active model is first identified. Once the model is identified, it is loaded into the H2O platform which, as with model training, is also responsible for the prediction process. Along with the model, the dataset to be used to make the predictions is also uploaded to the H20 platform. This dataset is composed only of the instances that have not yet been validated. As the predictions are being made, the *DynamicKnowledge* table is updated.

## 3.4. Monitoring and self-evaluation

One last relevant functionality of an autonomous learning system is that of monitoring and self-evaluation. Indeed, since everything that goes on happens without the supervision of a human expert, there is the need to monitor how the system is evolving, namely in terms of the performance metrics of the models in the pool. This becomes especially important when taking into account the fact that the performance of a model drops in a silent way, that is, the system continues to function normally and the models continue to make predictions, but their quality may drop dramatically.

With this in mind, and in order to address this problem, there is a whole section in the database consisting of a set of tables where different information is stored regarding both the performance of the models and the quality of the data (Figure 25) .

The table *Model* stores various information about each existing model, including the variable for which the model was trained and the date it was trained. In addition, and more importantly, a set of model performance metrics are also stored. These metrics, when properly analyzed, allow one to evaluate how well the model does in the presence of unseen data, that is, data not used in training and therefore unknown to the model. They are collected at the end of the training of each model, and it is also taken into account that there are metrics common to regression and classification problems and others unique to each of these types of problems.

However, the performance metrics of a model alone can be misleading since they are also dependent on the quality and diversity of the data with which the model is trained. In fact, a model's accuracy value being too high should be a cause for suspicion, since it often indicates that something may be wrong: does the dataset have enough instances? Are the instances diverse and different from each other? Was there overfitting during training? That said, in order to be able to analyze and control the quality of the data with which the models are trained, a set of characteristics relating to each variable are stored in the *VariableStats* table, such as the most frequent value, the percentage of null values or, in the case of numerical values, calculations such as the mean and median of the values.

Finally, at the end of the training of each model, the importance that each of the existing variables had in that same training is saved. This allows one to understand whether or not the new knowledge added by a human expert contributes significantly to the evolution of the system.

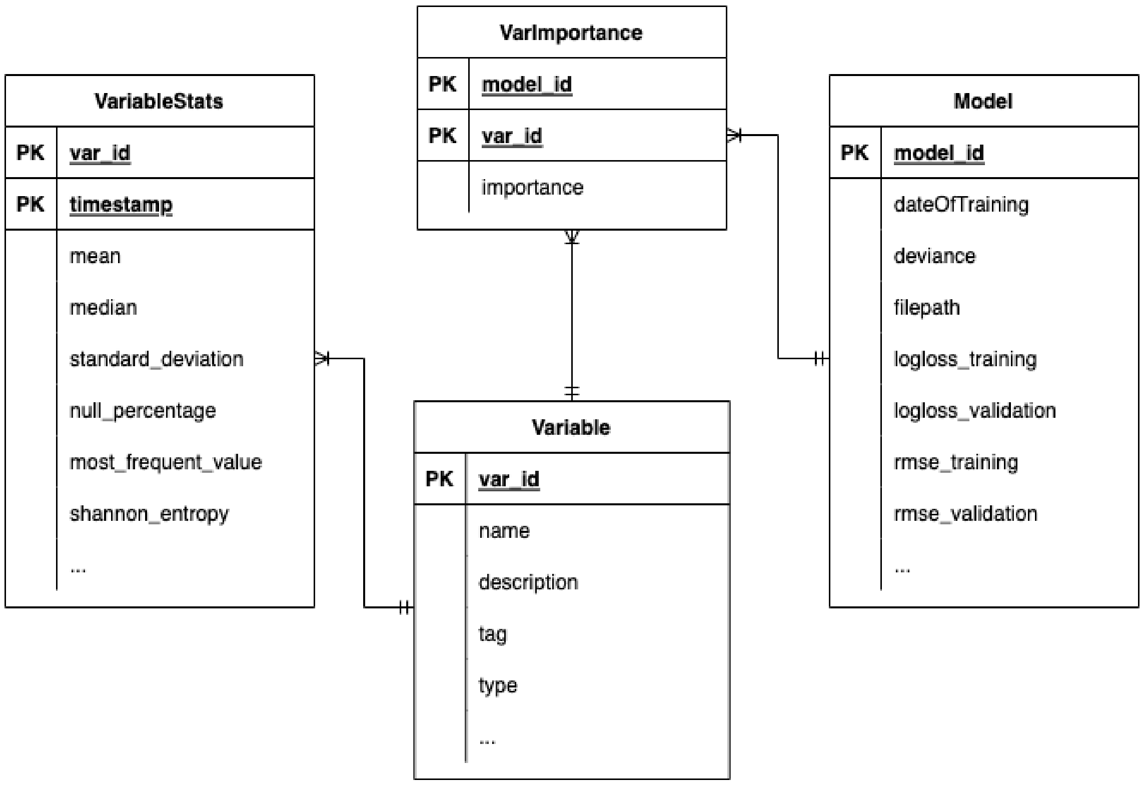


Figure 25 - Scheme of the database section where the information used for system monitoring is stored

## 4. Results

This section will describe the results obtained during the course of the project. Since this is a project with a very extensive practical component, the objective of this section is to demonstrate that the proposed solution can achieve the expected results.

These results can be divided into three sections: full version, demo version and AL.

### 4.1. Full version

As far as the full version is concerned, it is not possible to show many results, partly because the solution is not integrated, and it is not possible to observe it in a real-life context.

So there are two DAGs: one responsible for training a model, and the other responsible for predictions. The idea is that these run sequentially, with the prediction DAG being executed right after the model training DAG.

The training DAG is set to run once a week, using the *@weekly* tag (Figure 26). However, this is not happening, as this would require an instance of Airflow running continuously on a server, something that would only be done when integrating the implemented solution.

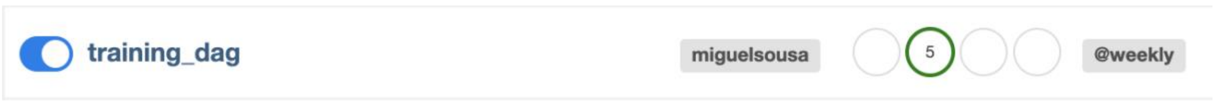


Figure 26 - Training DAG, as shown in the Airflow interface

Furthermore, it can be proven that this DAG does the job that is expected of it: for each existing dynamic variable, a task is created that trains a ML model associated to it, and another responsible for defining the variable's active model, that is, the model that the system must use to make the predictions (Figure 27).

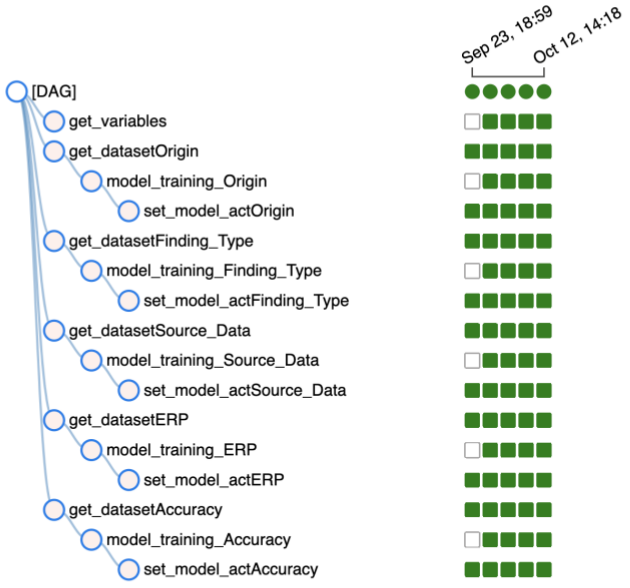


Figure 27 - Tree representation of the different successful executions of the training DAG tasks

As for the predictions DAG (Figure 28), it also has the *@weekly* tag, although it is in the same situation as the training DAG, that is, at this stage it only runs when manually triggered.

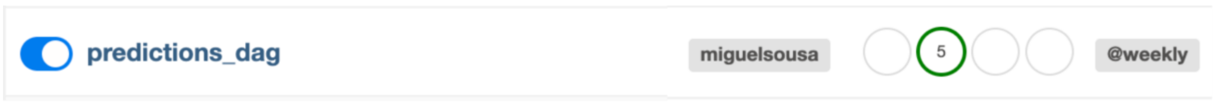


Figure 28 - Predictions DAG, as shown in the Airflow interface

Also in this case it is possible to verify that the work done by the DAG is as expected: for each dynamic variable that has the minimum number of validated instances, a task that is responsible for making/updating the predictions is created (Figure 29).

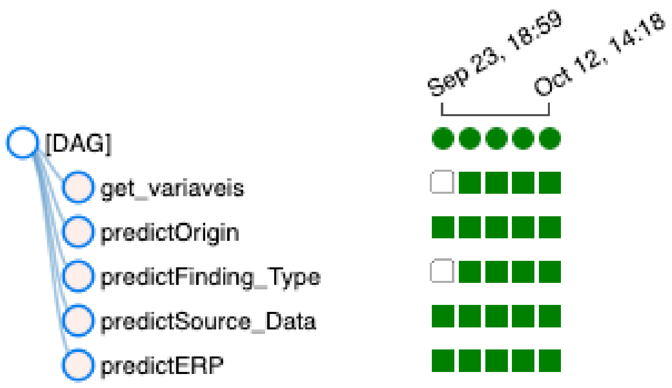


Figure 29 - Tree representation of the different successful executions of the predictions DAG tasks

## 4.2. Demo version

In order to be able to test and see the implemented solution fully operational, it would be necessary to integrate it with the already existing software, Col.bi. As this didn't happen, it became necessary to develop a demonstration in order to simulate the use of the solution in a "real life" scenario.

This demonstration was developed based on a monolithic architecture consisting of three layers that communicate with each other:

- The data layer, containing a MySQL relational database.
- The middle layer, containing a Node.js API. It is also through this API that a set of scripts written in Python are executed. The purpose of each of these scripts will be described later.
- The frontend, developed using the *javascript* library React.

This demo version has also been containerized using Docker. This resulted in 4 containers that communicate with each other through a virtual network (Figure 30).

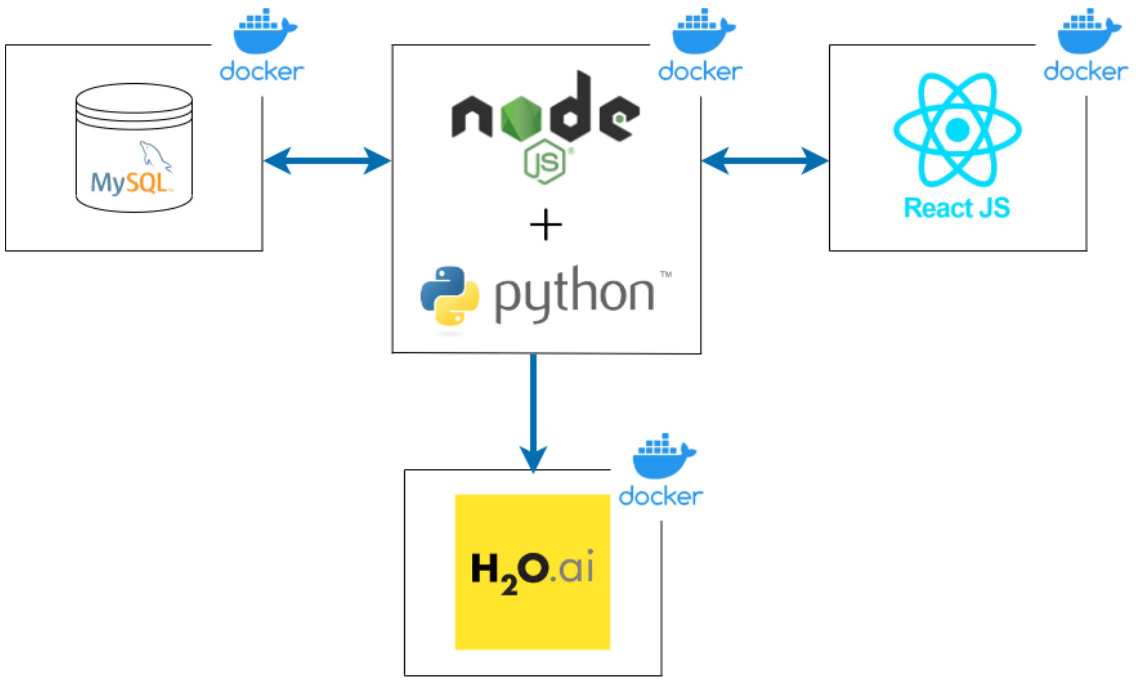


Figure 30 - Demo version structure and existing containers

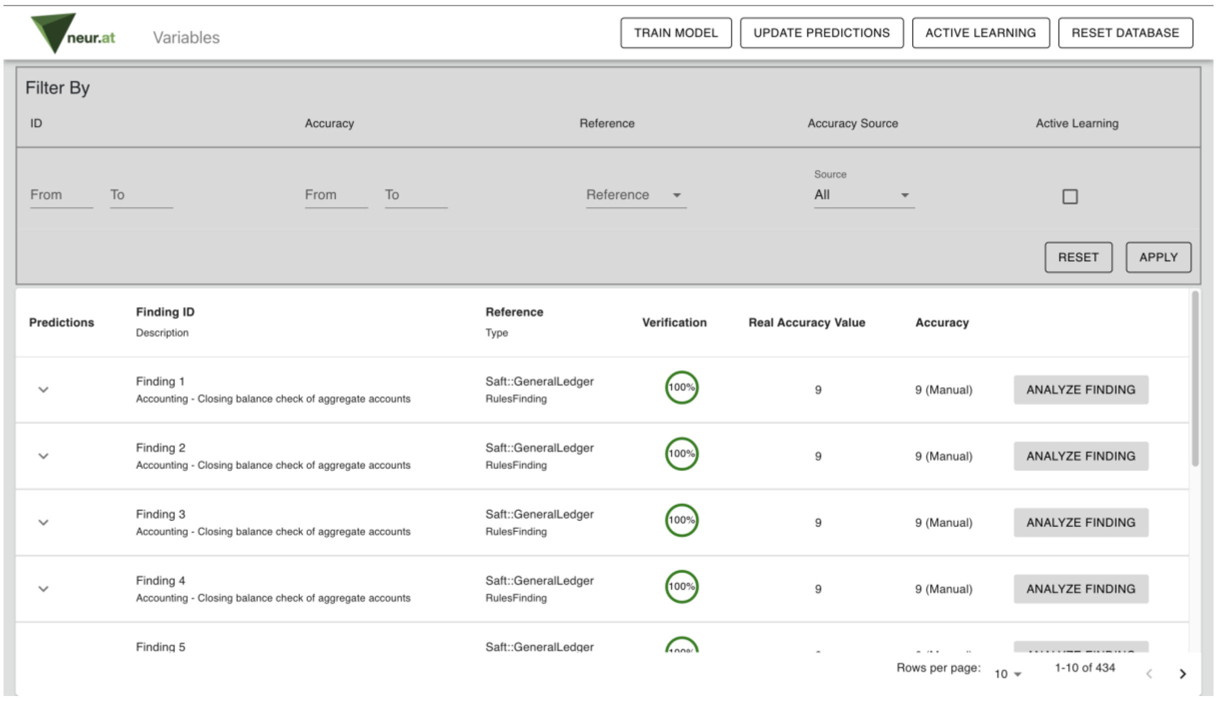
The main objective of this demo version was to make it possible to observe the evolution of the system as it interacted with the human expert and assimilated the knowledge provided by him. In order to make the experience faster and lighter, instead of models being trained and predictions being made for all variables, this version focuses on the one that was previously presented as the most important variable in the system, *Accuracy*, since it represents the potential level of fraud for a given instance.

The first step was to define the dataset to be used. To do so, a total of 434 instances previously validated by a human expert were selected, and subsequently 200 of them were manually modified in the database so that the system would consider them as non-validated instances.

The human expert interacts with the demo through an interface, represented in Figure 31, which, being part of a demo version, has additional debugging information and actions. For example, in this version, the training of models, the making/update of predictions and the use of AL must be triggered manually through a set of buttons in the navbar. In the final version, all these processes are automated. Furthermore, the human expert is presented with a list of instances to audit, which he can filter or even reorder by activating the AL module. Each instance, in turn, besides containing some basic information such as a description and its type, also has two very important fields that allow the evaluation of the system's evolution:

- Real Accuracy Value – value that the prediction made by the system is expected to approach or even equal. The display of this value in this version was due to the fact that it allows a quick and easy analysis regarding the general quality of the predictions made by the system.
- Accuracy – value assigned to the *Accuracy* variable in the context of the demonstration. It can assume "no prediction", when no prediction has yet been made for that instance, a value

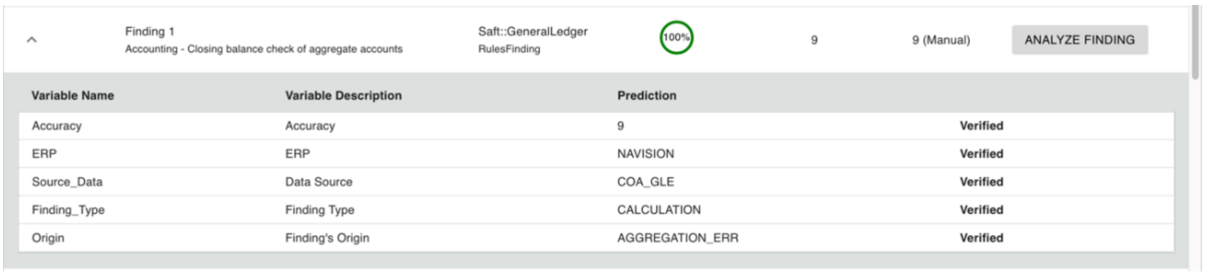
followed by “(Prediction)”, representing a value predicted by the system, or a value followed by “(Manual)”, representing a value assigned by the human expert.



| Predictions | Finding ID / Description | Reference / Type | Verification | Real Accuracy Value | Accuracy | |
|---|---|---|---|---|---|---|
| | Finding 1 Accounting - Closing balance check of aggregate accounts | Saft::GeneralLedger RulesFinding | 100% | 9 | 9 (Manual) | ANALYZE FINDING |
| | Finding 2 Accounting - Closing balance check of aggregate accounts | Saft::GeneralLedger RulesFinding | 100% | 9 | 9 (Manual) | ANALYZE FINDING |
| | Finding 3 Accounting - Closing balance check of aggregate accounts | Saft::GeneralLedger RulesFinding | 100% | 9 | 9 (Manual) | ANALYZE FINDING |
| | Finding 4 Accounting - Closing balance check of aggregate accounts | Saft::GeneralLedger RulesFinding | 100% | 9 | 9 (Manual) | ANALYZE FINDING |
| | Finding 5 | Saft::GeneralLedger | | | | |

Figure 31 - Demo version main page

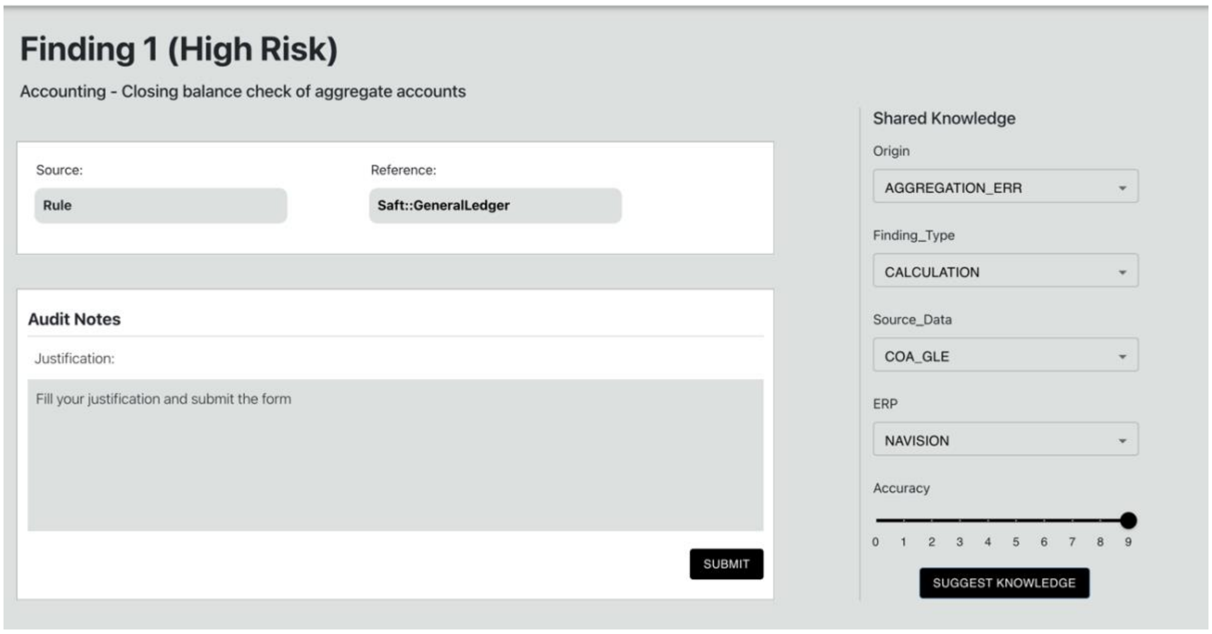
Each instance is associated with the dynamic variables that exist in the system and is only considered to be fully validated when all these variables are validated. The degree of validation of an instance presented to a user is measured based on the number of dynamic variables validated on the context of that instance. It is possible to check the status of each dynamic variable in the context of a given instance (Figure 32).



| Variable Name | Variable Description | Prediction | |
|---|---|---|---|
| Accuracy | Accuracy | 9 | Verified |
| ERP | ERP | NAVISION | Verified |
| Source_Data | Data Source | COA_GLE | Verified |
| Finding_Type | Finding Type | CALCULATION | Verified |
| Origin | Finding's Origin | AGGREGATION_ERR | Verified |

Figure 32 - Validation status of the dynamic variables associated with an instance

It is also possible, by selecting one of the instances, to analyze it in more detail (Figure 33). The system provides suggestions regarding the risk of fraud and the values for the dynamic knowledge, based on the ML models. The human expert does any changes deemed necessary and then saves them, marking the instance as validated. When this is done, all its data is moved to the labeled dataset and can be used as additional training data from that moment on.


**Finding 1 (High Risk)**

Accounting - Closing balance check of aggregate accounts

Source: Rule

Reference: Saft::GeneralLedger

**Audit Notes**

Justification:

Fill your justification and submit the form

Shared Knowledge

Origin: AGGREGATION_ERR

Finding_Type: CALCULATION

Source_Data: COA_GLE

ERP: NAVISION

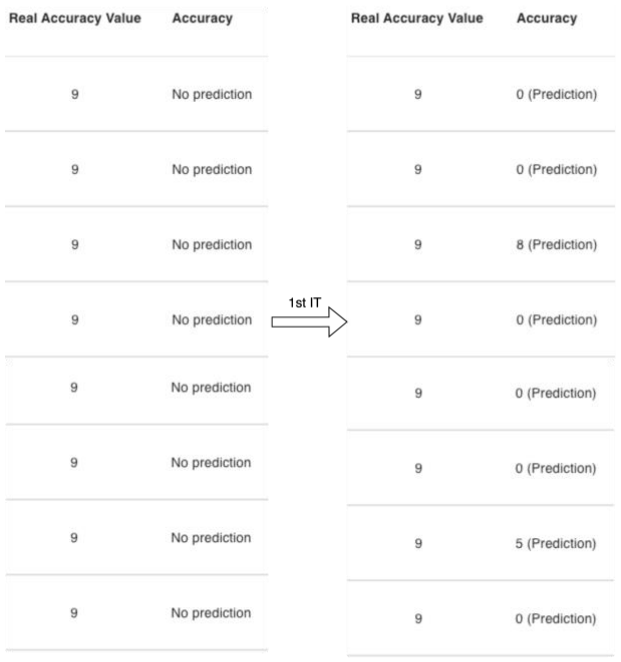
Accuracy: 9

Figure 33 - Interface through which the human expert provides feedback to the system

This is therefore the interface through which the human expert provides feedback to the system.

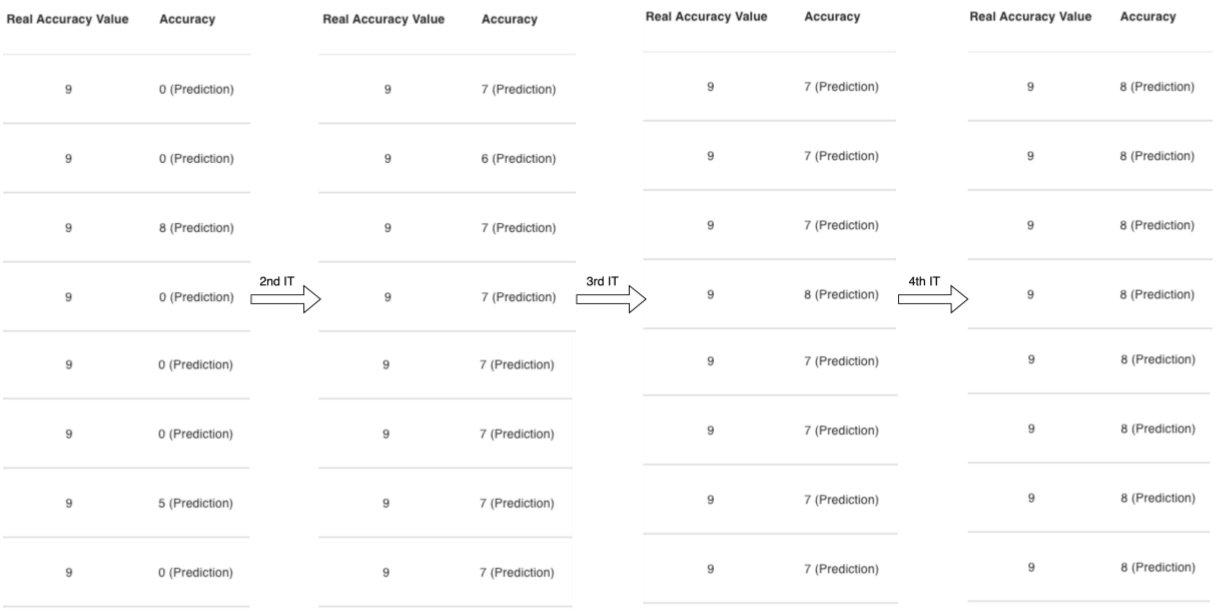
For demonstration purposes a model was first trained with the 234 initially validated instances and then the predictions were updated (Figure 34). Then, through the interfaces represented above, the interaction between human and system was simulated. For this, 5 instances that had not yet been validated were selected in an arbitrary way and feedback was provided on them.

The process was then repeated over several iterations until the predictions made by the system came close to the expected value (Figure 35).


| Real Accuracy Value | Accuracy |
|---|---|
| 9 | No prediction |
| 9 | No prediction |
| 9 | No prediction |
| 9 | No prediction |
| 9 | No prediction |
| 9 | No prediction |
| 9 | No prediction |
| 9 | No prediction |

| Real Accuracy Value | Accuracy |
|---|---|
| 9 | 0 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 5 (Prediction) |
| 9 | 0 (Prediction) |

Figure 34 - Result of the predictions made by the system in the first iteration of the simulation


2nd IT
3rd IT
4th IT

| Real Accuracy Value | Accuracy |
|---|---|
| 9 | 0 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 0 (Prediction) |
| 9 | 5 (Prediction) |
| 9 | 0 (Prediction) |

| Real Accuracy Value | Accuracy |
|---|---|
| 9 | 7 (Prediction) |
| 9 | 6 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |

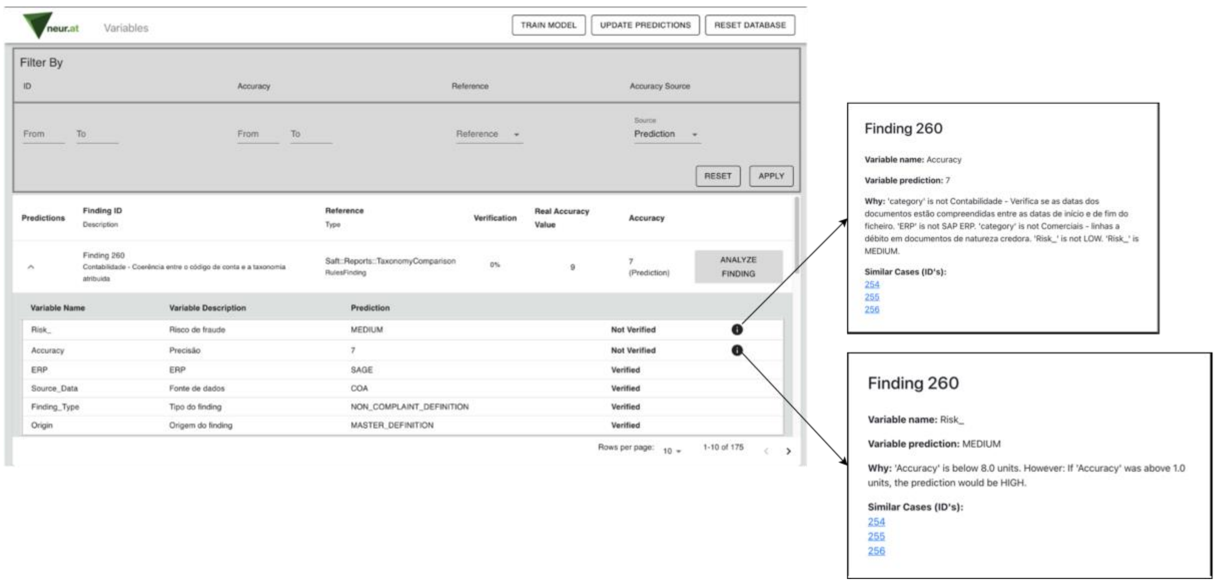
| Real Accuracy Value | Accuracy |
|---|---|
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |
| 9 | 7 (Prediction) |

| Real Accuracy Value | Accuracy |
|---|---|
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |
| 9 | 8 (Prediction) |

Figure 35 - Evolution of the system over the iterations of interactions with the human expert

At the end of the fourth iteration, the predictions made by the system are already much closer to the expected value, demonstrating the evolution of the system over time. Obviously, in a real-life scenario, where the data is much more diverse, it is not expected that the system evolves so quickly.

In order to demonstrate all the features of this version, a new variable, ‘Risk_’ was then created. From this moment on, the system started considering this new variable and making predictions for its value. When the value assigned to a dynamic variable comes from a prediction made by the system, it is possible to obtain an explanation of why the predicted value is that one, as can be seen in Figure 36.



| Predictions | Finding ID / Description | Reference / Type | Verification | Real Accuracy Value | Accuracy |
|---|---|---|---|---|---|
| ^ | Finding 260 Contabilidade - Coerência entre o código de conta e a taxonomia atribuida | Saft::Reports::TaxonomyComparison RulesFinding | 0% | 9 | 7 (Prediction) |

| Variable Name | Variable Description | Prediction | |
|---|---|---|---|
| Risk_ | Risco de fraude | MEDIUM | Not Verified |
| Accuracy | Precisão | 7 | Not Verified |
| ERP | ERP | SAGE | Verified |
| Source_Data | Fonte de dados | COA | Verified |
| Finding_Type | Tipo do finding | NON_COMPLAINT_DEFINITION | Verified |
| Origin | Origem do finding | MASTER_DEFINITION | Verified |

Finding 260

**Variable name:** Accuracy

**Variable prediction:** 7

**Why:** 'category' is not Contabilidade - Verifica se as datas dos documentos estão compreendidas entre as datas de início e de fim do ficheiro. 'ERP' is not SAP ERP. 'category' is not Comerciais - linhas a débito em documentos de natureza credora. 'Risk_' is not LOW. 'Risk_' is MEDIUM.

**Similar Cases (ID's):**
254
255
256

Finding 260

**Variable name:** Risk_

**Variable prediction:** MEDIUM

**Why:** 'Accuracy' is below 8.0 units. However: If 'Accuracy' was above 1.0 units, the prediction would be HIGH.

**Similar Cases (ID's):**
254
255
256

Figure 36 - Examples of explanations concerning the predictions made by the system

With the development of this demo version it was then possible to somehow validate the requirements involved in the proposed solution.

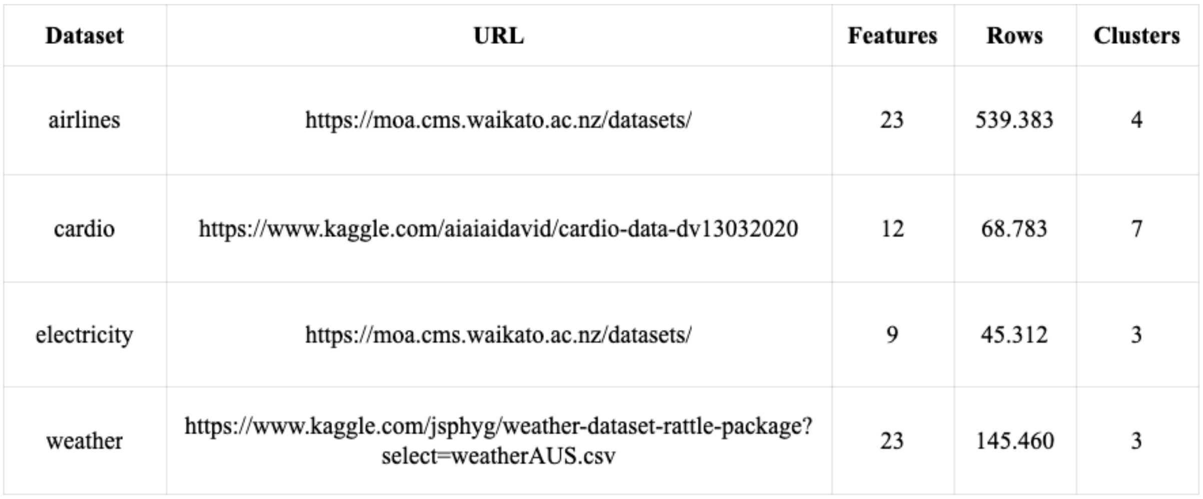
## 4.3. Active Learning

For validation purposes and given that the data being used in the context of this project is proprietary and sensitive, it is now described the methodology followed for validating the approach using some publicly available datasets.

The dataset is acquired and pre-processed, which includes typical tasks such as variable normalization, data cleaning, or converting variables into the appropriate type (e.g. numeric to nominal). The raw data is then split into three different sets. The first contains a single column with the dependent variable (the variable that the human expert will label, and that the model will predict). The rest of the data (the set of independent variables) is deemed the unlabeled dataset. The third set of data is an empty dataset, that will hold the labeled data as it is provided by the human expert. Optionally, this labeled dataset may already contain some labeled data at the start of the process.

Next, the k-means algorithm is run, to find the relevant clusters of data. The goal of using this unsupervised learning algorithm is to find groups of instances that are similar so that the human expert can be guided into labeling diversified instances and thus contribute with potentially more knowledge. To simulate a real scenario, clusters are searched using the unlabeled dataset, i.e., ignoring the existence of the labels, which are only used for quantifying the performance of the models trained.

The value of *k* is estimated iteratively and deterministically (i.e. *k* =1,2,3...): the value of *k* that has the lowest within-cluster sum of squares is selected. Numeric columns are also standardized to have a mean of zero and unit variance, in order to avoid mistaking a larger variance in certain variables with true contribution. Table 1 describes the four datasets used in this work, as well as the number of clusters found in each one.

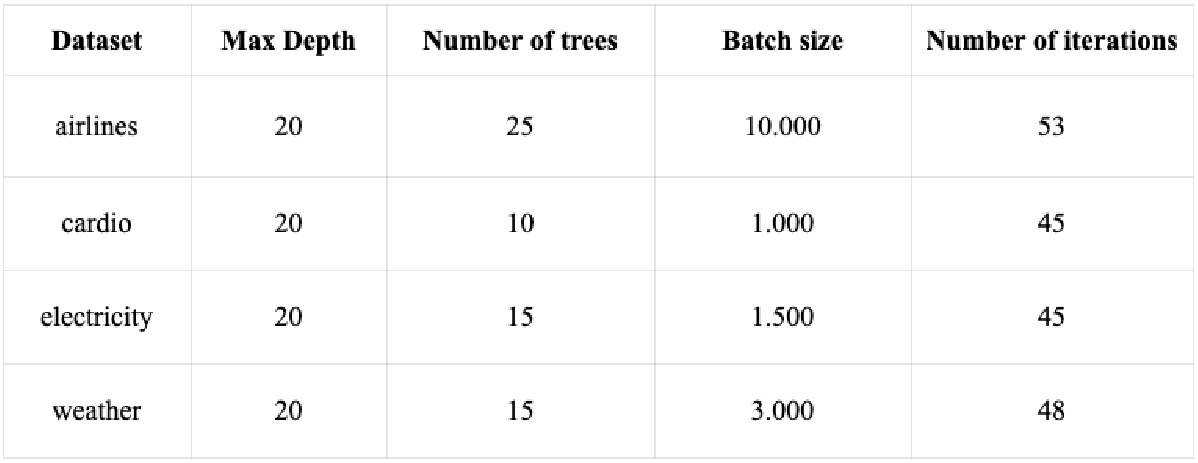


| Dataset | URL | Features | Rows | Clusters |
|---|---|---|---|---|
| airlines | https://moa.cms.waikato.ac.nz/datasets/ | 23 | 539.383 | 4 |
| cardio | https://www.kaggle.com/aiaiaidavid/cardio-data-dv13032020 | 12 | 68.783 | 7 |
| electricity | https://moa.cms.waikato.ac.nz/datasets/ | 9 | 45.312 | 3 |
| weather | https://www.kaggle.com/jsphyg/weather-dataset-rattle-package?select=weatherAUS.csv | 23 | 145.460 | 3 |

Table 1 – Characterization of the 4 datasets used and the number of clusters found in each one.

Once the number of clusters is determined, the process moves into the iterative phase, in which the goal is to simulate the action of a human expert. In each iteration a batch of *b* instances is selected, following the process described in Section 3.2. The actions of the human expert are simulated by adding the corresponding labels to the selected instances and adding those new instances to the labeled dataset. That is, we assume that the human expert labels all the instances suggested by the system.

After each batch of labeled data is added to the labeled dataset, a new model is trained with all the available labeled data. The goal of this work is not to find the most accurate model, but rather to validate the proposed approach. Thus, all models were trained using a Random Forest algorithm in which each tree has a maximum depth of 20 levels. The number of trees used in each case was defined intuitively based on the size of the dataset, as detailed in Table 2.



| Dataset | Max Depth | Number of trees | Batch size | Number of iterations |
|---|---|---|---|---|
| airlines | 20 | 25 | 10.000 | 53 |
| cardio | 20 | 10 | 1.000 | 45 |
| electricity | 20 | 15 | 1.500 | 45 |
| weather | 20 | 15 | 3.000 | 48 |

Table 2 - Main parameters used in the four experiments

This model is evaluated twofold: using 5-fold cross validation, producing the cross-validation metrics, and using the data remaining in the unlabeled dataset (for which the labels are known), producing the test metrics shown in the same picture. The two types of performance metrics are calculated since at the beginning and end of the process the labeled and unlabeled datasets, respectively, may be unrepresentative given their small size. Thus, considering both metrics provides a more reliable notion of the quality of the models over time.

To compare with the proposed approach, which we deem **balanced,** we run a similar process for each dataset, which we deem **arbitrary**. The goal is to analyze the evolution of the system if the human expert follows an arbitrary policy, which is not controlled by the system, to select and label instances. Several policies can be implemented. In this section we compare the proposed approach with one in which the human expert labels data by cluster, randomly. In both cases, the process ends when the unlabeled dataset is empty.

The performance of the system is evaluated by looking at how the performance metrics of the models evolve over time, as more labeled data are provided. Specifically, we consider two main criteria:

- **Accuracy** – Average of the absolute difference between the RMSE in each iteration and the RMSE of a model trained with all the data. This allows to estimate how good the model is in a given moment and over time, when compared to the final model.
- **Reliability** – Represents the variability of the RMSE over time, measured as the standard deviation of the RMSE. The lower the standard deviation, the more reliable the model is.

These criteria are then analyzed, comparing the proposed balanced scheme with the arbitrary one, for the already presented datasets.

The methodology followed in this evaluation is, however, slightly different to traditional ML experiments. Traditionally, one seeks to find the model with the smaller error or, in streaming problems, to find the sequence of models whose error is smaller over time.

In this case, the goal is to evaluate whether the proposed strategy for labeling instances will contribute to a better model, sooner in time or over time. Better, in this case, is defined as a model that minimizes the distance of the error metric (e.g. RMSE) to its final value. That is, we know, since we have access to the whole datasets, the RMSE of a model trained with the whole dataset. This allows us to know, at a certain point in time in the experiment, how far we are from its optimum value. For this reason, a very small value of RMSE might be as bad as a very large value: we are merely interested in the distance to its final value. Indeed, a given model with a small RMSE might be wrongfully characterized as good at a given moment, namely to an unrepresentative training or testing dataset. For this reason we look at two metrics, as described previously, which characterize the models' accuracy and reliability over time.

Figure 37 shows the evolution of the RMSE over time (calculated on the unlabeled data) in the four datasets. The orange line represents the RMSE of the proposed approach whereas the blue line represents an arbitrary labeling of the data. The same color-coding was used in the remaining figures. From the figure, it is possible to observe that the proposed approach holds error values that are generally closer to the final one (higher accuracy) and with less variations (higher reliability).

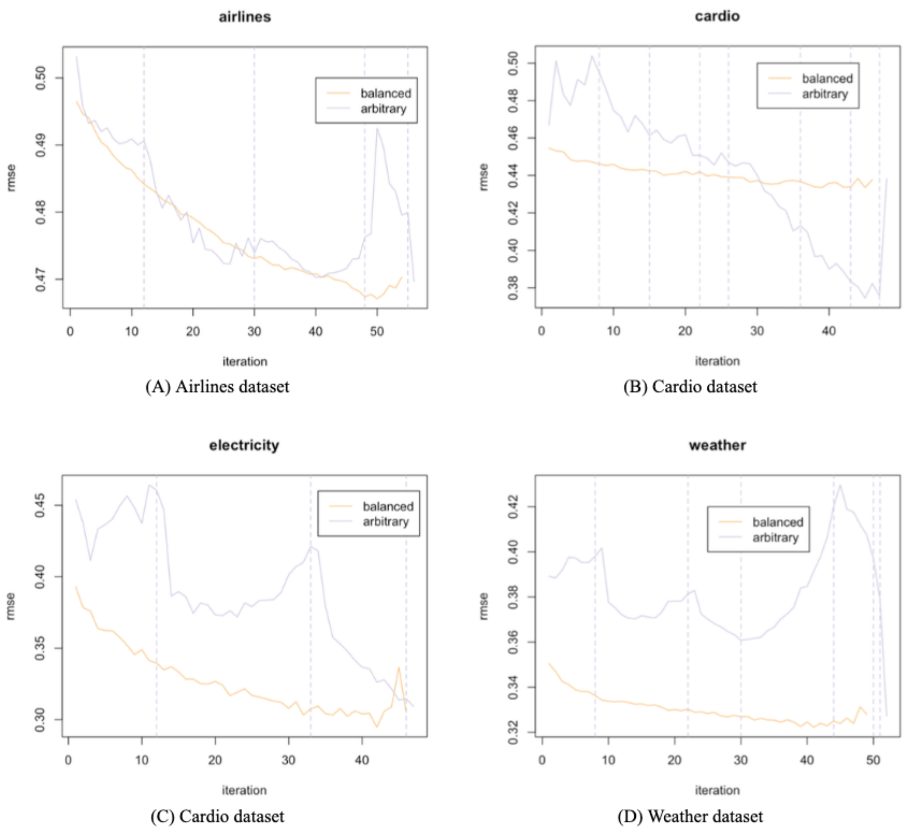
A similar conclusion can be drawn when the models are evaluated through 5-fold cross validation (Figure 38), eventually with the exception of the Electricity dataset (Figure 38C), in which the arbitrary approach followed holds values of RMSE that are generally closed to the final one, despite some variations.

These results are made clearer to observe in Figure 39 and 40. The former represents the distance of the RMSE in each model trained to the final RMSE, over time (with the RMSE measured through 5-fold cross validation). With the exception of the Electricity dataset (Figure 39), as already noted, the proposed approach tends to have significantly smaller distances. The variation of this distance is also smaller in the proposed approach, which shows that it provides a more reliable measure of error over time.

Figure 40 summarizes these data (distance of the RMSE to the final value) in the form of boxplots, to make the differences observed between both approaches easier to observe.

Table 3 summarizes, in a more objective fashion, the results that have been discussed so far. Each cell of the table represents the average or standard deviation of the absolute differences between the final RMSE and the RMSE of each model trained, for each of the two policies tested (balanced and arbitrary) and for both validation methods (test set and cross-validation). The goal is to compare both policies for each dataset and validation method. In this comparison, a lower average value in a certain policy means that its RMSE over time has been closer to the final one (higher accuracy). Similarly, a lower value of standard deviation in a policy means that it is more reliable, in the sense that there are fewer significant variations in its value over time. These represent the two main criteria put forward previously.

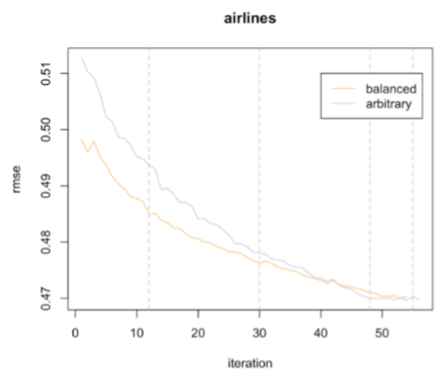
From Table 3 it is possible to conclude that the balanced policy is generally better than the arbitrary one, as the values for the former are significantly lower in 14 out of the 16 cases analyzed. Indeed, only in the 2 pairs highlighted in Table 3 (average and standard deviation for the electricity dataset in cross-validation) are there values higher for the balanced  policy than for its counterpart.



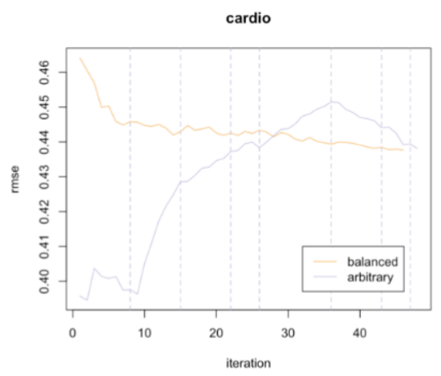
(A) Airlines dataset (B) Cardio dataset

(C) Cardio dataset (D) Weather dataset

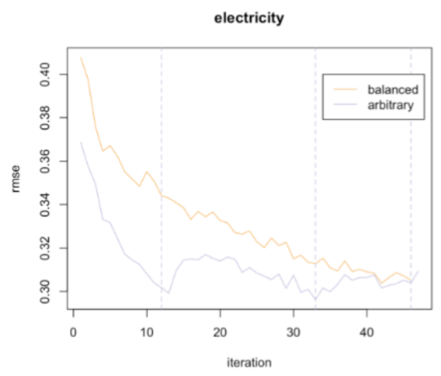
Figure 37 – Evolution of RMSE (calculated on the remaining unlabeled data) on the different problems, over time.



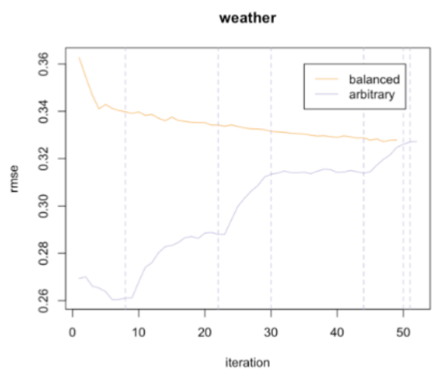
(A) Airlines dataset



(B) Cardio dataset

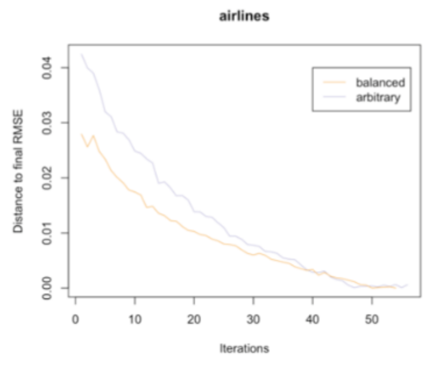


(C) Cardio dataset

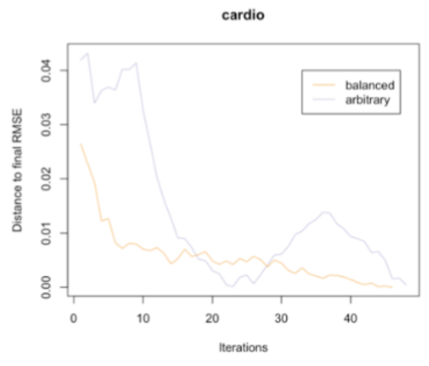


(D) Weather dataset

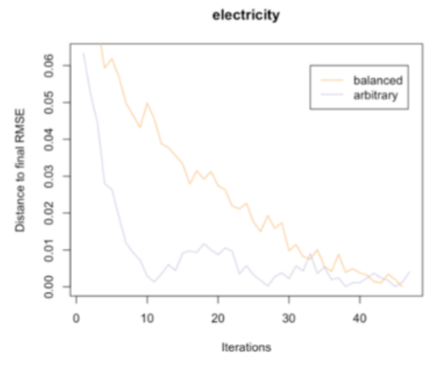
Figure 38 - Evolution of RMSE (5-fold cross validation) on the different problems, over time.



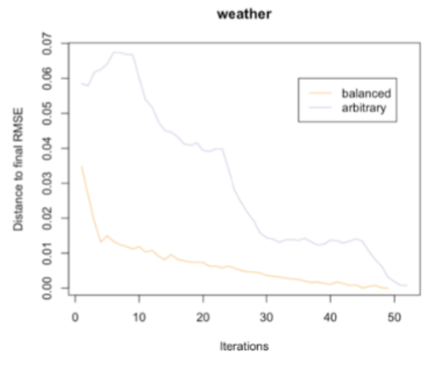
(A) Airlines dataset



(B) Cardio dataset

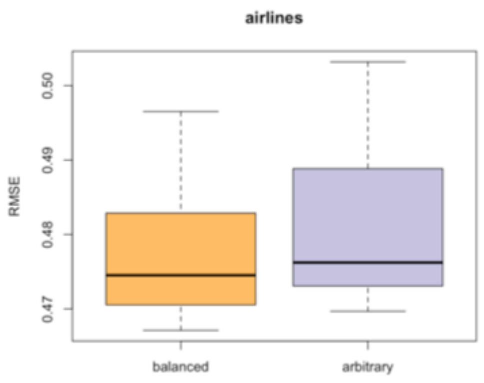


(C) Cardio dataset

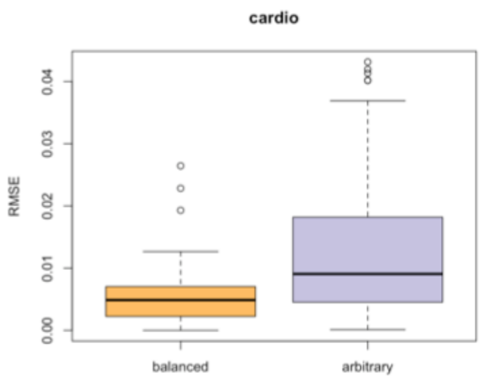


(D) Weather dataset

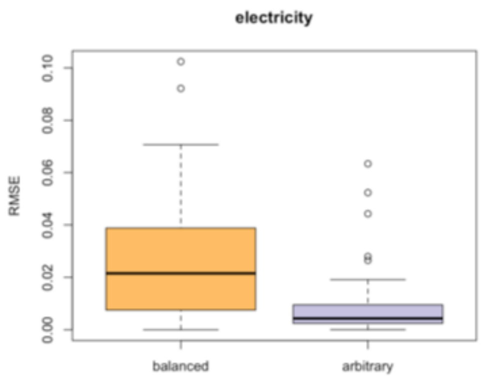
Figure 39 - Absolute difference to the final RMSE (5-fold cross-validation) for each problem, over time.



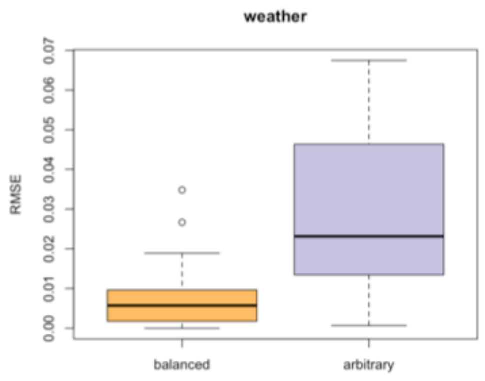
(A) Airlines Dataset



(B) Cardio Dataset



(C) Electricity Dataset



(D) Weather Dataset

Figure 40 - Distribution of the absolute difference to the final RMSE (5-fold cross validation) for each problem, over time.

| | Test | | | | Cross-validation | | | |
|---|---|---|---|---|---|---|---|---|
| | $\overline{X}$ | | $\sigma$ | | $\overline{X}$ | | $\sigma$ | |
| **dataset** | b | a | b | a | b | a | b | a |
| airlines | 0.0075 | 0.0096 | 0.0074 | 0.0084 | 0.0092 | 0.0124 | 0.0079 | 0.0119 |
| cardio | 0.0046 | 0.0310 | 0.0041 | 0.0191 | 0.0058 | 0.0140 | 0.0054 | 0.0136 |
| electricity | 0.0225 | 0.0840 | 0.0227 | 0.0438 | 0.0269 | 0.0092 | 0.0242 | 0.0132 |
| weather | 0.047 | 0.0544 | 0.0047 | 0.0189 | 0.0069 | 0.0306 | 0.0068 | 0.0214 |

Table 3 - Comparison of the proposed balanced approach (b) with the arbitrary one (a) for the four datasets in terms of the distance between the final RMSE and the RMSE in each iteration (average and standard deviation).

# 5. Conclusions and Future Work

## 5.1. Conclusion

In recent years, the field of ML has become increasingly relevant. This is mainly due to major changes in the social context, in which data are seen as something with increasing value, much because of the information that can be extracted from them. Thus, data has begun to be collected on a large scale by all companies in a search for competitive advantages. This has made traditional technologies and algorithms obsolete, as they are not able to process large volumes of data. As a result, the demand for ML solutions has been increasing, largely because of the ability of certain algorithms to categorize data and identify relevant patterns within them.

In fact, ML algorithms have evolved to address the challenges presented by this large volumes of data, which may change over time. For this, it is essential that models are able to adapt over time in order to stay up to date. One way to ensure not only that this happens, but also an increase in speed in terms of learning and adaptation, is by following approaches where there is interaction with a human expert. In these approaches, human experts provide feedback to the system, which leads to the system learning quicker.

Throughout this paper, the implementation of a system that follows this kind of approach is described, which, although being of general purpose, was developed in the context of a project in the domain of financial fraud detection. The main objective of this system is to allow the identification of true positive cases, mitigating the occurrence of false positives and thus facilitating the work of human experts.

To this end, the human expert is an integral part of the system, being responsible for providing feedback, which the system integrates and uses in its learning process. However, the main innovation of this system in relation to existing ones is that it is possible for the human expert to suggest new knowledge to the system, either through a new variable or a new value for an existing variable. The system is able to integrate the new knowledge and use it in the next iteration of model training.
The fact that the entire ML pipeline of the system is managed through Apache Airflow allows all processes to be automated, without the need for human intervention.

Finally, one of the last most relevant functionalities implemented was the possibility of using AL after test results proved that the performance of the models was better when the instances to be audited were chosen by the system and not arbitrarily.

In conclusion, in this project a solution was developed that, although not fully implemented (in the sense that it is not yet integrated into Col.bi), represents innovative variables in relation to existing solutions in the domain.

## 5.2. Future Work

As this is a project in which the developed solution seeks to complement the work done by an already existing software, Col.bi, it is obvious that the main future task, and the most interesting one, would be its complete integration. In this way it would probably become much easier to identify existing limitations and possible improvements to the implemented solution.

Other than that, it would be interesting to integrate into the developed solution the Model Performance Prediction module, developed by another colleague involved in a similar project. At this moment, all existing models are trained when running the existing DAG on Airflow. However, the fact that new data exist does not necessarily mean that there will be new patterns: the new data may be very similar to already existing one. The integration of this module would allow the system to only retrain a given model if some of the performance metrics of the model were expected to change by a minimum threshold, allowing a saving in resources.

As mentioned earlier, the choice of the algorithm to be used to train the models was based on a set of characteristics favorable to it in the context of the project. In the future, it would be interesting to test different algorithms, in order to compare and find the one with better performance.

## 5.3. Scientific Results

- 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 im- proved 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

- M. Sousa and D. Carneiro, "Interactive Learning in decision-support: an application to Fraud Detection", Iberian Conference on Information Systems and Technologies, CISTI. 2021.
  **DOI:** https://doi.org/10.23919/CISTI52073.2021.9476552
  **Abstract.** Usually, Machine Learning systems are seen as something fully automatic. Recently, however, 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.

- D. Carneiro, M. Sousa, G. Palumbo, M. Guimarães, M. Carvalho, and P. Novais, "Continuously learning from user feedback", Information Systems and Technologies, pp. 579-588, 2022.
  **DOI:** https://doi.org/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.

- D. Carneiro, P. Veloso, M. Guimarães, J. Baptista, and M. Sousa, "A Conversational Interface for interacting with Machine Learning models", 2022.
  **Available at:** 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.

## Acknowledgments

This work was supported by the Northern Regional Operational Program, Portugal 2020, and by the European Union, through the European Regional Development Fund (ERDF), through a research scholarship under the Neurat project (project number 39900), and by the Foundation for Science and Technology, under project UIBD/04728/2020.

# Bibliography

[1] "What are the Panama Papers," *The New York Times*, Apr. 04, 2016. https://www.nytimes.com/2016/04/05/world/panama-papers-explainer.html (accessed May 22, 2022).

[2] M. Forsythe, "Paradise Papers Shine Light on Where the Elite Keep Their Money," *The New York Times*, Nov. 05, 2017. https://www.nytimes.com/2017/11/05/world/paradise-papers.html (accessed May 22, 2022).

[3] S. P. Freedberg, S. Alecci, W. Fitzgibbon, D. Dalby, and D. Reuter, "How Africa's richest woman exploited family ties, shell companies and inside deals to build an empire," *International Consortium of Investigative Journalists*, Jan. 19, 2020. https://www.icij.org/investigations/luanda-leaks/how-africas-richest-woman-exploited-family-ties-shell-companies-and-inside-deals-to-build-an-empire/ (accessed May 22, 2022).

[4] P. Muncaster, "Global Fraud Hits £3.2 Trillion," *Infosecurity Magazine*, May 22, 2018. https://www.infosecurity-magazine.com/news/global-fraud-hits-32-trillion/ (accessed May 22, 2022).

[5] "Covid-19. Fraude financeira dispara 60,5% desde março com aumento de transações 'online,'" *Observador*, Nov. 24, 2020. https://observador.pt/2020/11/24/covid-19-fraude-financeira-dispara-605-desde-marco-com-aumento-de-transacoes-online/ (accessed May 22, 2022).

[6] S. Sagiroglu and D. Sinanc, "Big data: A review," in *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems, CTS 2013*, 2013, pp. 42–47. doi: 10.1109/CTS.2013.6567202.

[7] J. Karczewski, "Machine Learning Models vs. Rule Based Systems in fraud prevention," *Nethone*. https://nethone.com/pt/post/machine-learning-models-vs-rule-based-systems-in-fraud-prevention (accessed May 23, 2022).

[8] A. Dresch, D. P. Lacerda, and J. A. V. Antunes, *Design science research: A method for science and technology advancement*. Springer International Publishing, 2015. doi: 10.1007/978-3-319-07374-3.

[9] IEEE Xplore, "IBM Journal of Research and Development," *https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=5288520*.

[10] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers."

[11] "Machine Learning," *IBM Cloud Education*, Jul. 15, 2020. https://www.ibm.com/cloud/learn/machine-learning (accessed Apr. 11, 2022).

[12] I. Muhammad and Z. Yan, "SUPERVISED MACHINE LEARNING APPROACHES: A SURVEY," *ICTACT Journal on Soft Computing*, vol. 05, no. 03, pp. 946–952, Apr. 2015, doi: 10.21917/ijsc.2015.0133.

[13] S. Muhammad Abdulhamid, M. Shuaib, O. Osho, I. Ismaila, and J. K. Alhassan, "Comparative Analysis of Classification Algorithms for Email Spam Detection," *International Journal of Computer Network and Information Security*, vol. 10, no. 1, pp. 60–67, Jan. 2018, doi: 10.5815/ijcnis.2018.01.07.

[14] Chr. Madhuri, Mv. Pujitha, A. Professor, and A. Professor, “House Price Prediction Using Regression Techniques: A Comparative Study; House Price Prediction Using Regression Techniques: A Comparative Study,” 2019.

[15] J. P. Mueller and L. Massaron, “Machine Learning For Dummies.”

[16] IBM Cloud Education, “Unsupervised Learning ,” Sep. 21, 2020. ibm.com/cloud/learn/unsupervised-learning (accessed Apr. 22, 2022).

[17] “K-Means.” https://pypr.sourceforge.net/kmeans.html (accessed Oct. 12, 2022).

[18] B. Vidyavathi, “A Survey on Applications of Data Mining using Clustering Techniques,” 2015.

[19] D. C. G. Putri, J. S. Leu, and P. Seda, “Design of an unsupervised machine learning-based movie recommender system,” *Symmetry (Basel)*, vol. 12, no. 2, Feb. 2020, doi: 10.3390/sym12020185.

[20] J. E. van Engelen and H. H. Hoos, “A survey on semi-supervised learning,” *Mach Learn*, vol. 109, no. 2, pp. 373–440, Feb. 2020, doi: 10.1007/s10994-019-05855-6.

[21] D. Chamberlain, R. Kodgule, D. Ganelin, V. Miglani, and R. R. Fletcher, “Application of semi-supervised deep learning to lung sound analysis,” in *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, Oct. 2016, vol. 2016-October, pp. 804–807. doi: 10.1109/EMBC.2016.7590823.

[22] L. Liu, L. Chen, C. L. P. Chen, Y. Y. Tang, and C. M. Pun, “Weighted Joint Sparse Representation for Removing Mixed Noise in Image,” *IEEE Trans Cybern*, vol. 47, no. 3, pp. 600–611, Mar. 2017, doi: 10.1109/TCYB.2016.2521428.

[23] L. Pack Kaelbling, M. L. Littman, A. W. Moore, and S. Hall, “Reinforcement Learning: A Survey,” 1996.

[24] O. ’Theobald, “ML Categories,” in *Machine Learning For Absolute Beginners*, 2017, pp. 15–21.

[25] S. Heinz, “Using Reinforcement Learning to play Super Mario Bros on NES using TensorFlow,” May 29, 2019. https://towardsdatascience.com/using-reinforcement-learning-to-play-super-mario-bros-on-nes-using-tensorflow-31281e35825 (accessed May 11, 2022).

[26] S. Agrawal, “What is Offline/Batch learning?,” Jun. 03, 2020. https://medium.com/@sanidhyaagrawal08/what-is-offline-batch-learning-c90e7656641a (accessed May 15, 2022).

[27] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, “Online learning: A comprehensive survey,” *Neurocomputing*, vol. 459, pp. 249–289, Oct. 2021, doi: 10.1016/j.neucom.2021.04.112.

[28] S. Still, “Information theoretic approach to interactive learning,” Sep. 2007, doi: 10.1209/0295-5075/85/28005.

[29] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, “Learning under Concept Drift: A Review,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12. IEEE Computer Society, pp. 2346–2363, Dec. 01, 2019. doi: 10.1109/TKDE.2018.2876857.

[30] A. Holzinger *et al.*, “Interactive machine learning: experimental evidence for the human in the algorithmic loop: A case study on Ant Colony Optimization,” *Applied Intelligence*, vol. 49, no. 7, pp. 2401–2414, Jul. 2019, doi: 10.1007/s10489-018-1361-5.

[31] M. Ware, E. Frank, G. Holmes, M. Hall, and I. H. Witten, “Interactive Machine Learning-Letting Users Build Classifiers.”

[32] A. Holzinger, “Interactive machine learning for health informatics: when do we need the human-in-the-loop?,” *Brain Inform*, vol. 3, no. 2, pp. 119–131, Jun. 2016, doi: 10.1007/s40708-016-0042-6.

[33] S. Berg *et al.*, “ilastik: interactive machine learning for (bio)image analysis,” *Nat Methods*, vol. 16, no. 12, pp. 1226–1232, Dec. 2019, doi: 10.1038/s41592-019-0582-9.

[34] A. Holzinger and I. Jurisica, “Knowledge Discovery and Data Mining in Biomedical Informatics: The Future Is in Integrative, Interactive Machine Learning Solutions,” 2014.

[35] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, “Active Learning with Statistical Models,” 1996.

[36] B. Settles, “Computer Sciences Department Active Learning Literature Survey,” 2009.

[37] E. Mosqueira-Rey, D. Alonso-Ríos, and A. Baamonde-Lozano, “Integrating iterative machine teaching and active learning into the machine learning loop,” in *Procedia Computer Science*, 2021, vol. 192, pp. 553–562. doi: 10.1016/j.procs.2021.08.057.

[38] D. Pereira-Santos, R. B. C. Prudêncio, and A. C. P. L. F. de Carvalho, “Empirical investigation of active learning strategies,” *Neurocomputing*, vol. 326–327, pp. 15–27, Jan. 2019, doi: 10.1016/j.neucom.2017.05.105.

[39] P. Donmez and J. G. Carbonell, “Paired-Sampling in Density-Sensitive Active Learning.”

[40] D. D. Lewis, W. A. Gale, I. W. B. Croft, and C. J. van Rijsbergen, “A Sequential Algorithm for Training Text Classiiers,” Springer-Verlag, 94AD.

[41] A. Al-Masri, “How does Linear Regression Actually Work?,” Mar. 19, 2019. https://towardsdatascience.com/how-does-linear-regression-actually-work-3297021970dd (accessed Apr. 04, 2022).

[42] M. Brijain, R. Patel, M. Kushik, and K. Rana, “A Survey on Decision Tree Algorithm For Classification,” 2014. [Online]. Available: www.ijedr.org

[43] “Decision Tree Classification Algorithm.” https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm (accessed May 02, 2022).

[44] M. R. Dileep, A. v. Navaneeth, and M. Abhishek, “A novel approach for credit card fraud detection using decision tree and random forest algorithms,” in *Proceedings of the 3rd International Conference on Intelligent Communication Technologies and Virtual Mobile Networks, ICICV 2021*, Feb. 2021, pp. 1025–1028. doi: 10.1109/ICICV50876.2021.9388431.

[45] R. Dhanapal and Gayathiri P, “CREDIT CARD FRAUD DETECTION USING DECISION TREE FOR TRACING EMAIL AND IP,” 2012. [Online]. Available: www.IJCSI.org

[46] K. P. Sinaga and M. S. Yang, “Unsupervised K-means clustering algorithm,” *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.

[47] L. Vendramin, R. J. G. B. Campello, and E. R. Hruschka, “Relative clustering validity criteria: A comparative overview,” *Stat Anal Data Min*, vol. 3, no. 4, pp. 209–235, Aug. 2010, doi: 10.1002/sam.10080.

[48] Learn by Marketing, “What is K-means Clustering,” 2022. https://www.learnbymarketing.com/methods/k-means-clustering/ (accessed May 19, 2022).

[49] T. Kansal, S. Bahuguna, V. Singh, and T. Choudhury, *Customer Segmentation using K-means Clustering; Customer Segmentation using K-means Clustering*. 2018.

[50] K. jae Kim and H. Ahn, “A recommender system using GA K-means clustering in an online shopping market,” *Expert Syst Appl*, vol. 34, no. 2, pp. 1200–1209, Feb. 2008, doi: 10.1016/j.eswa.2006.12.025.

[51] IBM Cloud Education, “Random Forest,” Dec. 07, 2020. https://www.ibm.com/cloud/learn/random-forest (accessed May 12, 2022).

[52] “What is a Random Forest?” https://www.tibco.com/reference-center/what-is-a-random-forest (accessed May 30, 2022).

[53] A. Ziegler and I. R. König, “Mining data with random forests: Current options for real-world applications,” *Wiley Interdiscip Rev Data Min Knowl Discov*, vol. 4, no. 1, pp. 55–63, Jan. 2014, doi: 10.1002/widm.1114.

[54] A. Zheng, “Evaluating Machine Learning Models. A beginner’s guide to key concepts and pitfalls.,” Oct. 21, 2015. https://www.oreilly.com/content/evaluating-machine-learning-models/ (accessed Jun. 11, 2022).

[55] “Confusion Matrix in Machine Learning .” https://dotnettutorials.net/lesson/decision-tree-in-machine-learning/ (accessed Jun. 12, 2022).

[56] R. J. Bolton and D. J. Hand, “Statistical Fraud Detection: A Review,” 2002.

[57] N. R. Prasad, S. Almanza-Garcia, and T. T. Lu, “Anomaly detection,” *Computers, Materials and Continua*, vol. 14, no. 1, pp. 1–22, 2009, doi: 10.1145/1541880.1541882.

[58] A. Sharma and P. Kumar Panigrahi, “A Review of Financial Accounting Fraud Detection based on Data Mining Techniques,” 2012.

[59] M. A. Vasarhelyi, “Application of Anomaly Detection Techniques to Identify Fraudulent Refunds.” [Online]. Available: http://ssrn.com/abstract=1910468

[60] A. S. Sabau, “Survey of Clustering based Financial Fraud Detection Research.”

[61] J. Li, K. Y. Huang, J. Jin, and J. Shi, “A survey on statistical methods for health care fraud detection,” *Health Care Management Science*, vol. 11, no. 3. pp. 275–287, Sep. 2008. doi: 10.1007/s10729-007-9045-4.

[62] M. Ahmed, A. N. Mahmood, and M. R. Islam, “A survey of anomaly detection techniques in financial domain,” *Future Generation Computer Systems*, vol. 55, pp. 278–288, Feb. 2016, doi: 10.1016/j.future.2015.01.001.

[63] K. Tuyls, “Machine Learning Techniques for Fraud Detection Deep Learning for Robust Robot Control View project Game Theory View project.” [Online]. Available: https://www.researchgate.net/publication/254198382

[64] K. Chaudhary, J. Yadav, and B. Mallick, “A review of Fraud Detection Techniques: Credit Card,” 2012.

[65] S. I. Ao and International Association of Engineers., *International MultiConference of Engineers and Computer Scientists : IMECS 2011 : 16-18 March, 2011, the Royal Garden Hotel, Kowloon, Hong Kong*. Newswood Limited., 2011.

[66] R. K. Gopal and S. K. Meher, “A Rule-based Approach for Anomaly Detection in Subscriber Usage Pattern”.

[67] R. B. Basnet, A. H. Sung, and Q. Liu, “Rule-Based Phishing Attack Detection.”

[68] G. I. Webb, R. Hyde, H. Cao, H. L. Nguyen, and F. Petitjean, “Characterizing Concept Drift,” Nov. 2015, doi: 10.1007/s10618-015-0448-4.

[69] M. Herland, T. M. Khoshgoftaar, and R. A. Bauder, “Big Data fraud detection using multiple medicare data sources”, doi: 10.1186/s40537-018-0138-3.

[70] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, “A survey of machine learning for big data processing,” 2016, doi: 10.1186/s13634-016-0355-x.

[71] J. Verbraeken, M. Wolting, J. Katzy, J. Kloppenburg, T. Verbelen, and J. S. Rellermeyer, “A Survey on Distributed Machine Learning,” *ACM Computing Surveys*, vol. 53, no. 2. Association for Computing Machinery, Jun. 01, 2020. doi: 10.1145/3377454.

[72] J. P. Monteiro, D. Ramos, D. Carneiro, F. Duarte, J. M. Fernandes, and P. Novais, “Meta-learning and the new challenges of machine learning,” May 2021, doi: 10.1002/int.22549.

[73] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Trans Pattern Anal Mach Intell*, vol. 35, no. 8, pp. 1798–1828, 2013, doi: 10.1109/TPAMI.2013.50.

[74] K. Weiss, T. M. Khoshgoftaar, and D. Wang Background, “A survey of transfer learning,” 2016, doi: 10.1186/s40537-016-0043-6.

[75] R. A. Bauder, T. M. Khoshgoftaar, and T. Hasanin, “Data Sampling Approaches with Severely Imbalanced Big Data for Medicare Fraud Detection; Data Sampling Approaches with Severely Imbalanced Big Data for Medicare Fraud Detection,” *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, doi: 10.1109/ICTAI.2018.00030.

[76] N. v Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” 2002.

[77] H. He, Y. Bai, E. A. Garcia, and S. Li, “ADASYN: Adaptive synthetic sampling approach for imbalanced learning,” in *Proceedings of the International Joint Conference on Neural Networks*, 2008, pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

[78] G. Baader and H. Krcmar, “Reducing false positives in fraud detection: Combining the red flag approach with process mining,” *International Journal of Accounting Information Systems*, vol. 31, pp. 1–16, Dec. 2018, doi: 10.1016/j.accinf.2018.03.004.

[79] R. Wedge *et al.*, “Solving the ‘false positives’ problem in fraud prediction Automated Data Science at an Industrial Scale.” [Online]. Available: https://blog.riskified.com/true-cost-declined-orders/

[80] D. Carneiro, M. Guimarães, and M. Sousa, “Optimizing Instance Selection Strategies in Interactive Machine Learning: An Application to Fraud Detection,” in *Hybrid Intelligent Systems*, vol. 1375, 2021, pp. 124–133. doi: 10.1007/978-3-030-73050-5_13.

[81] H. Irshad, Q. Mirsharif, and J. Prendki, “Crowd Sourcing based Active Learning Approach for Parking Sign Recognition,” Dec. 2018, [Online]. Available: http://arxiv.org/abs/1812.01081