FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Reduction of non-regression time through artificial intelligence

Mariana Lopes Silva

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Jaime Cardoso

Supervisor at Natixis: Pedro Reis

July 27, 2020

Reduction of non-regression time through artificial intelligence

Mariana Lopes Silva

Mestrado Integrado em Engenharia Informática e Computação

July 27, 2020

Resumo

Natixis é uma empresa francesa criadora de soluções financeiras personalizadas, que possui várias linhas de negócio e milhares de clientes cujas *trades* devem ser geridas devido à constante evolução dos sistemas. O processo que inclui a verificação de mudanças nas trades quando uma evolução do sistema ocorre é traduzido em tempo de testes de não regressão, uma das preocupações do ecossistema Natixis. Essas mudanças são as diferenças encontradas nos resultados de testes financeiros entre o ambiente de produção e teste. Os *Business Analysts* são os responsáveis pela análise e identificação dessas diferenças, tornando este processo muito demorado. Simplificá-lo permitiria a redução da carga de trabalho desses analistas, ao mesmo tempo que diminuiria o tempo de introdução no mercado das evoluções funcionais.

A presente tese visa criar um protótipo de uma ferramenta que através de uma análise exploratória dos resultados dos testes financeiros, que constituem um grande e complexo conjunto de dados, reconhece em que dimensões dos dados ocorrem as diferenças após uma evolução do sistema. Assim, os dados chegam aos *Business Analysts* de maneira mais eficiente, permitindo-lhes uma interpretação mais rápida e fácil do resultado dos testes.

Deste modo, foi implementado um algoritmo, baseado em Árvores de Decisão, capaz de sumarizar e identificar as diferenças capturadas pelo enorme conjunto de dados que é derivado da comparação dos resultados de testes entre os ambientes de produção e teste. A solução inclui um protótipo de uma ferramenta que representa a árvore resultante do algoritmo de uma forma dinâmica e interativa para o utilizador.

A avaliação teve como objetivo perceber o grau de simplicidade da árvore gerada e consequentemente o seu grau de interpretabilidade para espcialistas financeiros. A implementação do protótipo foi realizada em conjunto com os *Business Analysts*, a fim de garantir os seus principais requisitos.

Para concluir, este projeto permitiu reduzir o tempo do processo de não-regressão, ajudando os *Business Analysts* na interpretação e análise dos resultados dos testes.

Keywords: Inteligência Artificial, Árvores de Decisão, High-dimensional Data, Financial Testing

Abstract

Natixis is a French company, creator of customized financial solutions, that has multiple business lines and thousands of clients, whose trades have to be managed, due to the constant evolution of the systems. The process that includes the verification of changes in trades when a new system evolution occurs is translated into non-regression test time, a concern of the Natixis ecosystem. These changes are the differences found in the results of financial tests between the production and testing environment. The Business Analysts are those responsible for the analysis and identification of those differences, leading to a very time-consuming process. Simplifying this would allow the reduction of these analysts' workload, while also decreasing the time-to-market for functional evolutions.

This thesis is aimed at creating a prototype of a tool that executes an exploratory analysis of the results of financial tests, which are a significantly large and complex dataset, in order to recognise in which data dimensions do the differences occur after a system evolution. Doing this enables the data to reach the Business Analysts in a more efficient manner, allowing them to interpret test results more quickly and easily.

In order to do this, an algorithm was implemented, based on decision trees, that is able to summarise and identify the differences captured by the large dataset that comes from the comparison of the results of the tests between the production and test environment. The solution includes a prototype of a tool that represents the resulting decision tree of the algorithm in a dynamic and interactive way.

The evaluation aimed to understand the degree of simplicity of the generated tree and, consequently, its interpretability for finance experts. The implementation of the prototype was done alongside with experts in finance, in order to ensure that satisfies their main requirements.

To conclude, this project intends to reduce the non-regression testing time by helping the Business Analysts in interpreting and analysing the test results.

Keywords: Artificial Intelligence, Decision Trees, High-dimensional Data, Financial Testing

Acknowledgements

I would like to show my gratitude to Natixis, for providing me a challenging environment for the development of this work and especially to my Supervisors at the institution, Pedro Reis and Sylvain Descamps, for all the guidance and advice given throughout this journey at Natixis. I also thank my Supervisor, Jaime Cardoso, for the help, advice and for always proposing new solutions.

To my parents for always being there for me, nothing would have been possible without them. To my sister, for all the support and for having always believed in me. To my boyfriend, for never letting me give up and helping me unconditionally. To my grandparents, for all the love.

To my friends who supported me during my academic career, Francisca, Luís and Xavier, for the help and knowledge sharing. A very special thanks to Francisca, for all the strength and friendship she gave me every day.

Last but certainly not least, I would like to mention my friends' squad, Cristiana, Daniela, Margarida, Jéssica and Vanessa, for giving me the strength to continue working and for friendship.

Mariana Lopes Silva

"Great things are not done by one person.
They are done by a team of people."

Steve Jobs

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AI	Artificial Intelligence
BA	Business Analyst
CORELS	Certifiable Optimal RulE ListS
FP	False Positives
FP	False Negatives
ID3	Iterative Dichotomiser 3
IG	Information Gain
KDD	Knowledge Discovery in Data
PCA	Principal Component Analysis
SQL	Structured Query Language
TP	True Positives
TN	True Negatives
WOE	Weight of Evidence
IV	Information Value

Chapter 1

Introduction

This study is part of a curricular dissertation project in a company environment, and it is a proposal to improve and reduce the time of the company's non-regression process.

This initial chapter provides an introduction to this dissertation by defining its scope, motivations, and objectives. Section 1.1 outlines the problem of non-regression process on Natixis ecosystem. Section 1.3 delineates the main forces driving this work and introduces the problem tackled by this thesis. Section 1.4 describes what this project aims to achieve. Lastly, Section 1.5 explains the structure of the rest of the document.

1.1 Context

Natixis is a French corporate and investment bank of Groupe BPCE, that has multiple business lines and many clients that include corporations, financial institutions, sovereign and supranational organizations. In order to guarantee the robustness and integrity of the systems and client trades[1], the company uses the non-regression testing process to verify whether a new or modified feature operates correctly, in other words, if any expected or unexpected changes occurred in trades after the new feature has been introduced into the system. It should be noted that the term regression used is in the sense of checking if there has been any setback in the Natixis system, and it is not related to the term regression used in Machine Learning. When applying non-regression testing, it is only checked the evolving unit or module instead of the whole product. Due to the large quantity of data to be tested in Natixis' systems, this process can save resources and time.

In Natixis, during the non-regression process, three types of test reports that correspond to a financial data, namely Profit and Loss Test, Hedge Test and Stress Test are generated for the reference and the test environments of the system. This process aims to compare the differences in specific values of the referred reporting financial data between those two environments, in order to verify the compatibility of old and new software upgraded versions. This work will focus only on the data generated by the Profit and Loss Test, which is explained in more detail in Chapter 2.

[1]Trade is a term used in finance to represent the transactions of goods or services. [Hay20]

1.2 Problem Definition

As mentioned previously, the non-regression process includes a comparison between the financial data of the reference and test scenarios of the system, to check if any expected or unexpected changes occurred in trades after the new feature has been introduced into the system. For example, the trade *x* takes the profit and loss value of 2500€ in the production environment and 2600€ in the testing environment. It can be seen that there is a difference of 100€ in that trade between the two scenarios. Several other trades have these discrepancies. Thus, during the search for those differences, which can represent inconsistencies or irregularities in trades, it is necessary to involve one or more Business Analysts to identify them manually and to understand in which characteristics of the trades they happen. As previously mentioned, in this project, the focus is just on the differences that are observed in profit and loss values.

The financial data that comes from the mentioned test reports represent a huge and complex dataset, so it can be hard for Business Analysts to identify the possible differences and draw conclusions from them. The expert interpretation of Business Analysts for decision making in the non-regression process is imperative. However, the fact that this comparative analysis between the two scenarios still is manual makes the process very time-consuming, subjective and more prone to errors. For this reason, the significant amount of time devoted to the non-regression process is one of the biggest concerns in the Natixis ecosystem.

1.3 Motivation

The role of Business Analysts in the non-regression process is crucial to ensure that the new feature operates correctly and do not negatively affect the behaviour of the trades. However, this methodology with the exclusive human responsibility can take much time and be subject to more failures. It translates directly into a loss of competitiveness since it affects the time to market for functional evolutions. So, it is imperative to speed up this process to a multinational financial service firm like Natixis can answer to market demands even more effectively.

Therefore, the present project arises in this scope and environment and should focus on implementing a prototype of a tool that helps on the decision making of Business Analysts in the analysis of differences, not replacing the fundamental role they play but helping them to make the process faster and more reliable.

1.4 Objectives

The main goal of this dissertation is to help Business Analysts in decision making on the non-regression testing process, through the identification of which data dimensions support the differences in profit and loss values of the trades, between the production and test scenarios when a new system evolution occurs.

The mentioned data dimensions correspond to the attributes and respective values of the multidimensional dataset that are the result of Profit and Loss test reports. It is assumed that each column of a data set represents a particular *attribute* and the rows of a data set list the *values* for each attribute/column. So, the specified data dimensions represent the financial properties where the trades are inserted.

To achieve the objectives mentioned above of helping Business Analysts is proposed the implementation of an algorithm, from the dataset that contains the results of the comparison of profit and loss values between the test and reference scenarios. This algorithm defines, in a simple and intuitive way, which data dimensions the differences occur in, by summarizing the combination of dataset variables related to those differences. This information reaches Business Analysts, allowing them to interpret and analyze the test results more quickly and easily.

Summing up, the solution should be a prototype of a tool that allows Business Analysts to have a fast interpretation of the comparison of the results in the Profit and Loss reports generated before and after the introduction of system evolution. Thus, Business Analysts will be able to identify and analyze the differences and, consequently, understand which ones are meaningful and which can be the ignorable ones.

Finally, this project is intended to reduce the non-regression test time, resulting in the speeding up of the interactions with external systems, clients (traders, sales, brokers) and with the other teams within Natixis. Consequently, that reduction in time also translates to a considerable gain in capacity and a decrease in the required working hours, once the workload on Business Analysts and the time-to-market for functional evolutions are reduced.

1.5 Outline

This dissertation is divided into seven chapters, including this introductory chapter that contextualizes the reader with the problem motivation and the objectives that this work intends to explore. In Chapter 2 (p. 5) is described the environment that supports this thesis. Chapter 3 (9) presents the state-of-the-art regarding decision support tools for Business Analysts and the relevant literature review associated with high dimensional data analysis and methods that help in summarization and extraction of relevant information of datasets. Chapter 4 (p. 21) presents the problem statement and introduces the proposed solution and validation. Chapter 5 (p. 25) contains a detailed description of the implemented solution for the problem. Chapter 6 (p. 39) includes a discussion of the results. At last, the document ends in Chapter 7 (p. 49) with the conclusions of the work and points out future work.

Chapter 2

Technical background

This chapter introduces the technical background that served as a basis for the fundamental choices in the development of the algorithm, detailing essential concepts to this application. Section 2.1 explores the subject of non-regression tests in finance, specifically the report generated by the Profit and Loss test that is used for the construction of the dataset. Section 2.2 introduces the concept of Artificial Intelligence and Data Mining process that helps to summarize and extract meaningful information to support decision making. Section 2.2.2 details the concept of high dimensional data and present some ways to deal with them. Finally, Section 2.3 summarizes this chapter.

2.1 Non-Regression Tests

Non-regression testing only focuses on the evolving features or modules to check whether those changes have the desired behaviour and if they are compatible with the previous version of the product (before the introduction of the new functionality) [Poo20]. Through this process, the companies can save resources and time, since not the whole application is tested, as in the case of regression testing. [Vas20].

From this process, can be generated reports with updated values or information of any nature after the product update. In this work, the report that is considered is the Profit and Loss that is detailed below.

2.1.1 Profit and Loss Test

The profit and loss (P&L) statement or income statement is a financial report of a company that summarizes the revenues and expenses incurred during a specified period [Inc20] [Ros20]. The P&L statement reveals the past or future profitability of a business and the information about a company's ability to create profit by generating sales and managing expenses [Sho20].

According to [Inc20], the main categories that can be found on the P&L include Sales, Cost of Goods Sold, Selling, General & Administrative Expenses, Marketing and Advertising, Technology, Interest Expense, Taxes and Net Income. This report is based on accounting principles that include revenue recognition (revenue is often recognized before cash is received), matching (expenses are matched to revenues during the period those revenues are earned), and accruals (income and expenditures should be recorded during the periods they occur, not when cash is received).

Profit and Loss Attribution Test is a back-testing method for evaluating a bank's risk management models [Inv20]. The test compares the hypothesis front-office pricing models and the theoretical profit and loss calculated by the bank's own risk management models. If the difference between that two P&L's is too big or too variable, then a breach is counted. Four breaches within one year will force the trading desk onto the Standardised Approach (SA) [Ris20a].

This test helps banks evaluate their strategies, their financial strength, the attractiveness of investing in them or acquiring the entire business. Besides that, Profit and Loss report can also justify losses incurred as a result of unforeseen conditions they could not help [Inv20].

2.2 Artificial Intelligence

In Computer Science, Artificial Intelligence (AI) focuses on the simulation of intelligence in computer systems by enabling them to perform human mental activities [Nil14]. In a few words, AI aims to build smart machines [In20], by using algorithms to achieve an explicit purpose or solution.

Artificial Intelligence has applicability in the most diverse areas or industries, so companies each time more use it. Correctly, in Finance and Economics, AI is commonly used to detect frauds, to organize and monitor some processes and operations or to manage the trades market [Wik20b].

2.2.1 Data Mining

Data mining or Knowledge Discovery in Data (KDD) consists of extracting useful information from a huge and complex dataset by finding out structures and patterns in data [GA10]. This technique is crucial for Artificial Intelligence systems since it provides them with suitable and relevant data [Hel20].

The data mining process is composed of the following steps [FPSS96]:

1. The first step is getting an **overview of the application domain** and establishment of the data mining objectives accordingly to the business environment.

2. The second step is about **data collection and understanding**. In this phase, the data set is selected respecting the required goals, and it can be from multiple sources.

3. The third step includes the **data preprocessing** and can be very time-consuming. In this stage, the data is cleaned by filling the missing or unknown values and removing the unwanted noise through smoothing. After this data transformation, the resulting dataset can already be modelled.

4. The fourth step comprises the **definition of variables** to characterize the data set. To achieve this, the dimensionality reduction techniques can be applied.

5. The fifth step is the **modelling**, where it is chosen the mathematical model or function that corresponds to the initial purpose, such as summarization, classification or clustering.

6. The sixth step corresponds to the **selection of the algorithm** to be used that adapts to the data type (categorical, numerical or continuous) and the final business goal.

7. The seventh step corresponds to the choice of **data mining technique**. Those referred techniques can be divided into the following groups:

   - **Classification**: This method helps to classify data in different and predefined classes.
   - **Clustering**: Identifies the groups of data items based on the degree of similarity. This process helps to understand the differences and similarities between the data.
   - **Regression**: Analyzes the relationship between variables. It is used to identify the likelihood of a specific variable, given the presence of other variables.
   - **Summarization**: Provides a compact description for a subset of data. More sophisticated functions involve summary rules, multivariate visualization techniques, and functional relationships between variables. Summarization functions are often used in interactive exploratory data analysis and automated report generation.
   - **Association Rules**: This data mining technique helps to find the association between two or more items, intending to discover a hidden pattern in the data set.
   - **Outlier detection**: This data mining technique refers to the observation of data items which deviate from other samples and can be used to anomaly or fraud detection.
   - **Sequential Patterns**: This technique finds out similar patterns or trends in data.
   - **Prediction**: This method, as its name suggests, predicts the classes based on past events. This technique can be applied in conjunction with the other mentioned methods.

8. The eighth step consists of **evaluation or interpretation** of the extracted results. In this phase, it is checked if the identified patterns are meaningful for the initial business goals.

9. The last step is the **deployment** when the knowledge acquired during all process is applied to the business operations. Reports with lessons learned are also created at this final stage.

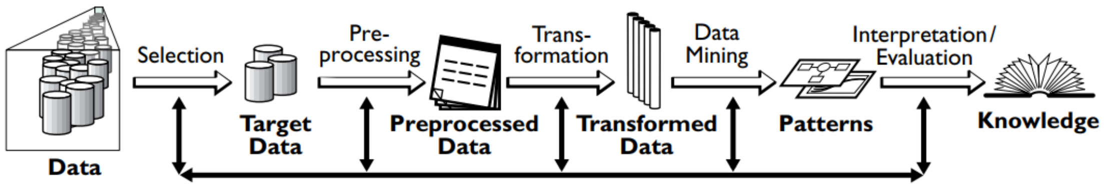


Figure 2.1: Steps of the Knowledge Discovery in Data. Source: [FPSS96]

2.2.2 High-Dimensional Data

After the explanation of AI and Data Mining process, it is useful to understand the concept of high-dimensional data that is an increasingly common pattern in the daily life of artificial intelligence. High-dimensional data means that it is characterized by multiple dimensions [AI20], so working with this data can become extremely hard. Areas as finance deal with this challenging issue and behavioural scientists need powerful, effective analytic methods to glean maximum scientific insight from these data [Pen20].

To facilitate the processing of data by the computer, many algorithms can be used to find a lower-dimensional subspace [AI20], because for high-dimensional data sets, reducing the dimensionality is an important way for reducing the problem dimension [BKK96]. This reduction of dimensionality simplifies the data processing and ensures that the integrity of data is maintained [To20]. Thus, there are many methods to project higher dimensional data into lower dimensions, such as **Principal Component Analysis**, that aims to reduce a large number of variables. Clustering methods for grouping data can be another solution to treat high-dimensional data.

There are also other methods to deal and extract the primary information from a high-dimensional dataset without reducing the dimension, such as **Decision Trees** algorithms or **CORELS** algorithm that is based on a list of rules. Both techniques can produce human-readable structures and for this reason, are interesting to the considered application.

2.3 Summary

This chapter described significant concepts that will support the understanding of this report. It was divided into two main sections, one in the finance area and another in the data analysis area.

The first one presents the description of non-regression tests and the importance of the Profit and Loss report for a company. This type of reports is the data set to be used. The other one introduces the concept of Artificial Intelligence and high dimensional data analysis.

Chapter 3

Literature Review

The chapter reflects the state-of-the-art regarding useful algorithms that help in summarizing and interpreting complex and high-dimensional datasets, to support the analysis process and consequently in decision making. Some of those methods were not used in the implementation, but they were essential keys to achieve the best solution. In section 3.2, it is also presented the relevant literature review associated with the non-regression process in finance and tools that can help Business Analysts on decision making in the finance field.

3.1 High Dimensional Data Analysis

Nowadays, all the current problems have to deal with multidimensional data, making up a challenge the decision making from data. Also, Business Analysts are confronted with the diversity of data types and multidimensionality that characterize financial data. To overcome this issue, there are many algorithms dedicated to the summarization and organization of data to extract knowledge from them, allowing a straightforward way for human interpretation. Instead of working on large-scale data, there are data mining techniques, such as extraction of the relevant subspace, dimensionality reduction or sampling methods, used to increase the interpretability and efficiency of the data analysis [LN14]. Section 3.1.1 defines the PCA technique, section 3.1.2 explains the Decision Tree concept and describes the ID3 algorithm and section 3.1.3 introduces the CORELS method.

3.1.1 PCA (Principal Component Analysis)

Principal Component Analysis is a technique used for detecting patterns in high-dimensional data through the reduction of the number of its dimensions [Smi02]. In this way, PCA intends to compress the data to get the main information from it, by expressing the data with a new set of variables (that are the principal components) [AW10]. This statistical method can be used to accelerate the learning process of an algorithm and reduce the memory needed to store data.

To perform PCA, it is necessary a preprocessing of the data, namely, mean normalization. From each data dimension, it is subtracted its mean. The mean result of the dataset is zero [Smi02].

Considering this training set with m dimensions: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ and the **mean normalization** as $\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)}$. Each $x_j^{(i)}$ is replaced by $x_j - \mu_j$. After that, to reduce data from m dimensions to k dimensions, it is calculated the covariance matrix $(\frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T)$ that describes the degree of relationship between two features. The next step on the algorithm is to calculate the highest eigenvalues and eigenvectors of the covariance matrix to get the rows that characterise the data [Smi02]. The features with lower variance or significance (smallest eigenvalues) can be rejected because the lost information is not much. Eigenvectors represent directions and eigenvalues represent the relative importance of the directions [DKZ13]. The number of the selected eigenvectors will be the number of dimensions of the resulting data [Smi02]. Then, it is built the matrix of the selected eigenvectors that are the principal components. Finally, the new dataset is calculated according to those k principal components.

Generally, PCA seeks the linear combinations of the original variables such that the derived variables (called principal components) capture maximum variance [ZHT06].

The following picture represents an example of the application of PCA. The left graph is original data X, and the right graph is the original data transformed so that the axes are the principal components. It should be noted that the two charts show the same data and the principal components are perpendicular to one another because they are linearly independent [Bre20a].

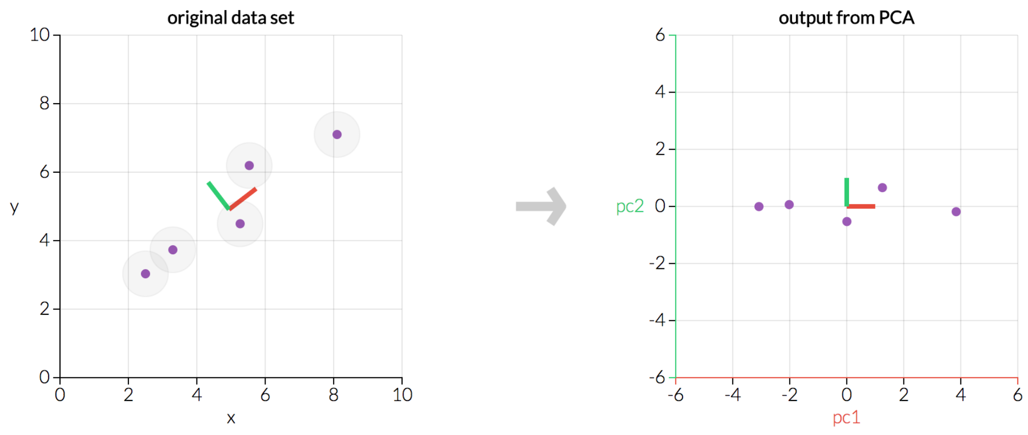


Figure 3.1: Example of the application of PCA

The feature transformation applied by PCA can represent a drawback for this work, since the new features that are a linear combination of the original features may not be adequate to interface with Business Analysts. The solution to be presented to them must be as clear as possible, without much subjectivity and the PCA result, for the reason mentioned above, can be hard to interpret.

3.1.2 Decision Trees

A decision tree is a method that splits a dataset as a hierarchical group of relationships logically arranged in a tree-like structure, and it is used to classify data into pre-defined classes [NEM09] [SP16] [TTM15]. A decision tree comprises a root node (the initial node which hasn't any input branches and it is labelled with an attribute name of a dataset), **decision nodes** (the internal nodes that split into one or more branches and they are marked with the attribute names), **branches** (that represents the possible values of the selected attribute that can be a decision node or the root node) and **leaf nodes** (the nodes that don't split, when all attributes were computed, and they are labelled with the class name that the sample belongs) [dJG14] [SP16]. That basic structure of a Decision Tree is shown in Figure 3.2.

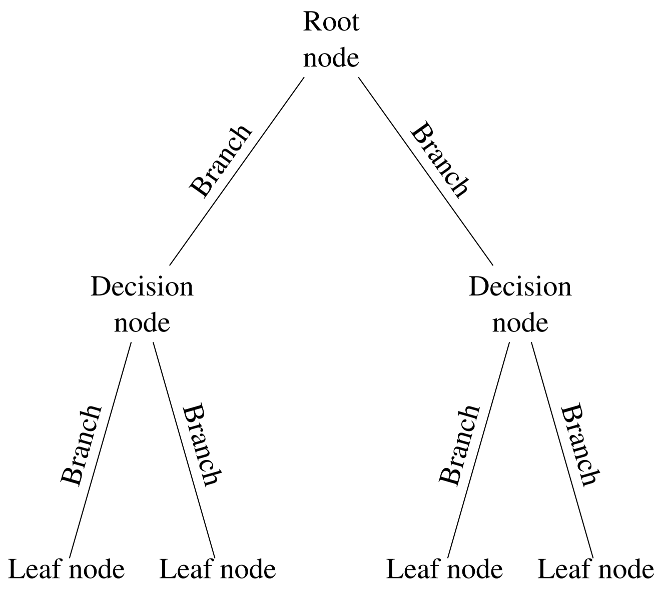


Figure 3.2: Basic structure of a Decision Tree

Decision trees have many real applications, more particularly in decision analysis field assisting in the definition of a strategy to reach some decision or goal. In machine learning, decision trees are commonly used for classification problems by producing logical rules of classification through the representation of a tree-like model of decisions. The paths from the root node to leaf nodes mean classification or decision rules. There are many reasons to justify the high popularity of the method [BMCY11] [NEM09] [Wik20a]:

1. This decision support tool can deal with complex problems by providing a simple and understandable representation, easy to explain and interpret.

2. The tool can handle categorical and numerical variables.

3. The tool can be linked with other decision methods.

4. The outcomes of the tree can be customized by the inferences of experts and their intuitions.

The process of build a decision tree uses a recursive top-down approach [BMCY11], starting from the root node of the tree with the whole dataset and going through branches and decision node until the leaf node is reached, which indicates the corresponding class. For that, the decision tree algorithms use a heuristic evaluation function and a stopping criterion function [Leo17]. In the next section, this process called Top-Down Induction of Decision Tree, used as the basis for many decision tree induction algorithms will be described in more detail [dJG14].

3.1.2.1 Decision Tree Top-Down Induction Process

Consider the dataset in Table 3.1, that represents a classification problem to determine whether or not we should play tennis. Each line of the table means one observation that is characterized by four attributes (outlook, temperature, humidity and windy) and it is classified into two types of classes (yes and no) that answers to the first question about playing tennis.

Table 3.1: dataset example from [Bre20b]

Outlook	**Temperature**	**Humidity**	**Windy**	**PlayTennis**
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

This dataset is the input given to the decision tree induction algorithm which is a greedy search recursive algorithm, i.e., at each decision node it seeks to find an optimal local solution to achieve the optimal global solution [dJG14] and comprises the following points:

1. If every observation in a subset of the dataset belongs to the same class or if the subset is very small, then a leaf node is created with the corresponding class [SP16].

2. On the other hand, if the set of observations belongs to more than one class, it is selected by a heuristic evaluation function the best attribute (that will be a decision node of the tree), and it is split into subsets (the new branches) according to their different values.

3. The algorithm is called recursively, and it is applied to the resulting subsets of the point 2 until the stopping criterion function is reached and the leaf nodes have fixed the corresponding class [NEM09] [SP16].

3.1.2.2 Heuristic Evaluation Function

The heuristic evaluation functions distinguish different approaches to build a decision tree. As mentioned above, the algorithm at each decision node tries to find the best feature that ensures the creation of subsets, where objects are distributed less randomly. [BMCY11] [NEM09] [ABE06]. In general, the evaluation functions trying to minimize the degree of impurity of the leaf nodes by measuring the homogeneity of the target variable within the subsets [dJG14]. A node is **pure** if all examples that are part of that node belong to the same class. A node is **impure** if the opposite occurs, if not all examples of the node belong to the same class.

These metrics can be based on entropy and information gain calculations giving rise to the ID3 or C4.5 algorithms, on Gini impurity or variance reduction used by the CART algorithm, on distance or dependence [dJG14]. Some of these function are presented in the following subsections.

3.1.2.3 ID3 (Iterative Dichotomiser 3)

The ID3 (Iterative Dichotomiser 3) algorithm is one of the most popular classification algorithms based on decision trees and developed by Ross Quinlan [SP16]. ID3 is commonly used in machine learning and natural language processing fields to make complex scenarios into simple decision trees. The algorithm is used to generate a decision tree by splitting the dataset into nodes in such a way as any example is classified by following the path along the tree represented by the decision nodes, their branches and finishing with leaf nodes that have the final label in the same way as any other decision tree algorithm [TTM15].

ID3 distinguishes itself by applying the entropy as an impurity measure, which is described below [dJG14]. Therefore the algorithm works as follows:

1. Calculates the entropy for each attribute;
2. Calculates the Information Gain for each attribute that is the difference between the entropy of the current set S ($E(S)$) and entropy for each attribute;
3. The attribute with the lowest entropy (that is, the least variation in values) or the highest information gain is chosen as the root of the tree, splitting the tree from this attribute (A);
4. For each possible value of attribute A:
   (a) If only one class C can be reached, that class is a leaf of the tree;
   (b) If it is possible to reach several classes, the algorithm is repeated as a subset of data referring to that value a_j (possible value j for the attribute A) and the two trees are joined.

The process is recursively repeated on each subset until one of these stopping conditions occurs:

- Every point in the subset have the same class.
- Although not all of the points have the same class, all attributes have already been selected.

- There are no further points in the subset.

[SP16]

Applying ID3 to the dataset from Table 3.1, the resulting decision tree is represented in Figure 3.2.

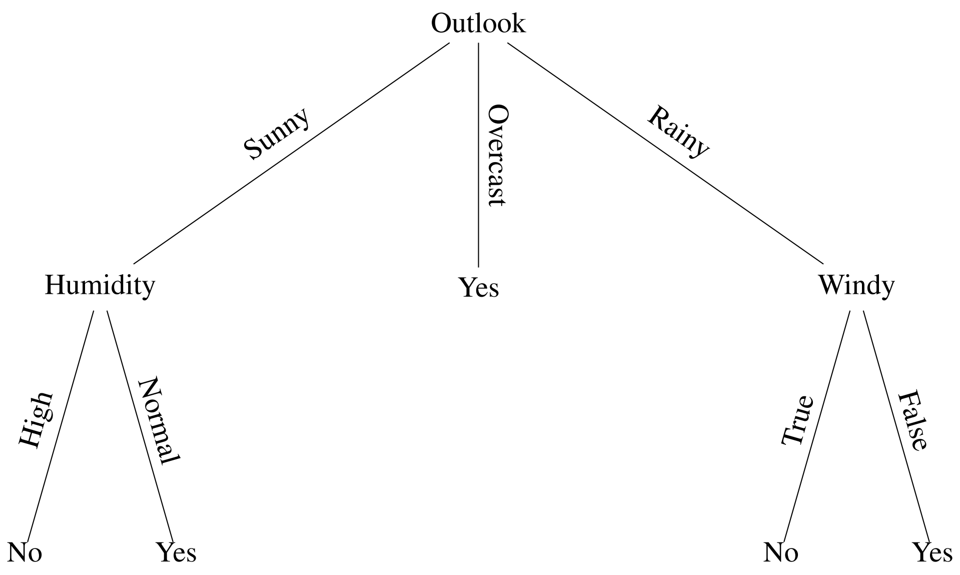


Figure 3.3: Decision Tree for the tennis classification problem

ID3 have attractive benefits to the concerned application [YN10]:

- Easy construction of the tree.
- Handles with categorical and numerical data.
- Classification speed is relatively high.
- Deals with huge datasets and make them simpler.

Entropy

Originally derived from thermodynamics, the entropy is a property of a system, and its value is part of the state of the system [Dem07]. Generically, the entropy is a measure of the disorder of the system [NFB+19], as such, the higher uncertainty, the higher entropy. In information science, the entropy means how heterogeneous a dataset is, for instance, when a dataset has various classes with the same likelihood to occur, the maximum entropy is achieved, and less information is provided, once there is no certainty when trying to predict the label of an instance. [Sim19] says that a random distribution has high entropy because if it is compacted, it is impossible not to lose important information.

Using the same dataset (3.1) of the previous section that has two classes (yes and no), the value of entropy range between 0 and 1 where 0 represents a pure node (all instances belong to the same class) and 1 is the highest degree of impurity of a node (the number of instances that have "yes" as the label is the same as the number of instances that have "no") [dJG14].

The ID3 algorithm uses the **Shannon entropy** which formula is presented in 3.1. Shannon entropy uses the probability distribution of events to represent the value of the information present in each instance of data [SP16].

$$H(C) = \sum_{i=1}^{N} -p(c_i) \times \log_2(p(c_i)) \tag{3.1}$$

Where:

- N - the number of possible values for the class C
- c_i - the possible value i for the class C
- $p(c_i)$ - the proportion of the number of instances with c_i to the number of instances in set (probability of occurring c_i)
- $\sum_{i=1}^{N} p(c_i) = 1$

IG (Information Gain)

Information Gain is related to the maximum amount of information that can be transmitted by a specific dataset. This measure is necessary to determine if an attribute A is suitable to split the tree, by showing how much certainty was added in the set when that attribute is chosen to be a decision node. The formula that is shown below (5.1) is defined by the difference between the entropy of the parent node and the sum of the entropy of the resulting child nodes, multiplied by their probabilities [dJG14].

$$IG(C,A) = H(C) - H(C|A) \tag{3.2}$$

Where:

$$H(C|A) = \sum_{j=1}^{M} p(a_j) \times [-\sum_{i=1}^{N} p(c_i|a_j) \times \log_2(p(c_i|a_j))] \tag{3.3}$$

- M - the number of possible values for the attribute A
- N - the number of possible values for the class C
- a_j - the possible value j for the attribute A

- c_i - the possible value i for the class C
- $p(a_j)$ - the proportion of the number of instances with value a_j to the number of instances with attribute A (probability of occurring a_j)
- $p(c_i|a_j)$ - the proportion of the number of instances with value a_j in class c_i to the number of instances with attribute A (probability of occurring a_j with c_i)
- $\sum_{j=1}^{M} p(a_j) = 1$
- $\sum_{i=1}^{N} p(c_i|a_j) = 1$

3.1.3 CORELS (Certifiable Optimal RulE ListS)

CORELS is a branch-and-bound algorithm that finds the optimal rule list of the list of rule antecedents of a dataset. This model is a supervised learning method, that can be an alternative to decision tree algorithms [ALSA+17] since it ensures an interpretable result and a simple construction approach.

That technique distinguishes itself by the way it receives the dataset. CORELS is inputted not a list of instances with their attributes and respective values but a list of rule antecedents (the concept of **Rule Lists** is described below).

3.1.3.1 Rule Lists

Rule Lists or Decision Lists are a collection of rules composed of if-then statements of features or conjunction of features that capture a set of observations [LSAA+18], as can be seen in the example below.

If age < 20 then predict health = very good
If salary > 5000 then predict house = big
If likes do sport and age < 25 then risk of cardiovascular incident = 10%

This structure allows a straightforward interpretation of the predicted model, contrasting with black-box models as Neural Networks that aren't human interpretable [LS17]. Thus, the algorithm can work with all type of data (continuous, categorical or numerical) through the transformation of the features into categories that could then become binary features.

3.1.3.2 Objective Function

The objection function (shown below in 3.4) consists of the sum of the loss function and a constant (regularization parameter, K) proportional to the length of the rule list [LSAA+18]. The parameter K is used to penalize longer rule lists and to prevent overfitted results. The loss function ℓ measures the number of instances that are incorrectly labeled [LS17].

$$R(d,x,y) = \ell(d,x,y) + \lambda K \tag{3.4}$$

Where:

- d - the rule list
- x - the training data features
- y - the training data labels
- ℓ - the fraction of samples whose classes are incorrectly predicted (misclassified samples)
- λ - the regularization parameter (an accuracy penalty)
- K - the length of the rule list

CORELS intends to minimize the objective function to maximize accuracy and use the advantages of the regularization constant (λ). To achieve reasonable runtimes, the algorithm draws a branch-and-bound search strategy, allowing to prune the search space as much as possible.

3.1.3.3 Lower Bound

As mentioned above, CORELS calculate a lower bound for each prefix (that is a rule list without the label), by computing the loss of the prefix plus the accuracy penalty proportional do the length of the prefix. *Lower bound* means the best possible result of a prefix for the objective function [LS17].

On the one hand, if the *lower bound* of a prefix is higher than the smallest objective value found until now, the prefix and its children can be prune from the tree. On the other hand, if the *lower bound* of a prefix is smaller than the smallest objective value found until now, that prefix will be added to the data structure that saves the rule lists, that is a priority queue [VK20].

$$lowerbound(p,x,y) = \ell(p,x,y) + \lambda K \tag{3.5}$$

Where:

- p - the prefix
- x - the training data features
- y - the training data labels
- ℓ - the loss function of the prefix
- λ - the regularization parameter (an accuracy penalty)

- K - the length of the prefix

Although this method has known advantages to the concerned application, finding an optimal rules list for datasets that are large and whose attributes have many types of values, requiring several columns as input, can result in long lists of rules which are difficult to interpret for Business Analysts.

3.2 Decision support tools for Business Analysts

Due to the large amount of financial data that experts have to deal, financial decision problems can become extremely complicated and time-consuming if their interpretation is exclusively made for humans. Thus, the issue of helping Business Analysts in decision making always was an object of study, and some methods have been proposed to support their actions.

Methodologies based on statistical analysis, multi-criteria decision, rough sets (that deal with imprecision and uncertainty in decision making), artificial intelligence (such as neural networks) or other data mining techniques have been used to help Business Analysts in a wide variety of financial classification problems. The statistical methods can join their multiple decision variables and then achieve inferences, but they still have many obstacles in the formulation of the decision classes. The rough set theory is a technique that deals with uncertainty in the decision-making process through the establishment of the decision rules based on the past choices and priorities of the decision-maker. The expert systems technology that simulates the reasoning of a professional in a specific area has already been used in finance, understandably providing evaluations and forecasts. However, it is necessary much time to calibrate those systems. The neural networks were also considered in the financial field since they can simulate the human brain and predict results. However, their implementation and interpretation of the results can be arduous and time-consuming. Another known methodology for financial decision problems is the use of Multi-criteria Decision aid methods (MCDA) that evaluates several conflicting decision variables and incorporates the preferences of the analyst [ZD98].

The article [DZ10] proposed a multi-criteria methodology developed in an integrated decision support system that provides useful analysis hypothesis and reports for the expert analysts evaluate and rate the banks' performance. That user-friendly tool can be used daily for analysts, since facilitating the interpretation of data and consequently support their conclusions and inferences.

Another multi-criteria method combined with decision support systems, named FINCLAS (FINancial CLASsification) was proposed by [ZD01]. The software integrates financial modelling tools with preference disaggregation methods, allowing the expert analysts to develop decision models according to their preferences and policies from the structure of the problem until the final decision. That system provides minimization of the cost and the time of a decision-making process in a financial classification problem, that can be the evaluation of a corporate or country performance, the credit risk evaluation, capital investments, among others.

Despite the existence of these support tools for Business Analysts, there isn't a tool only oriented towards the non-regression process. In Natixis, this process is manually done, through the comparison of the generated report files, that include finance calculations, to check (in case of the Profit and Loss test) whether the profit-and-loss values of the trades are constant between the production and test scenarios. That comparison is usually made with the help of Microsoft Excel.

3.3 Conclusions

Some solutions and algorithms related to high dimensional data analysis and summarization were introduced. This review of the state-of-the-art was completed with a survey about decision support tools to helps business analysts which established that there are very few works in the field of decision support in the non-regression process, thus highlighting the innovative aspect of this thesis.

Chapter 4

Problem Statement

This chapter aims to describe the issue addressed in this work and presents the approach to tackle it. Section 4.1 details the time consuming process in identifying differences from the non-regression process that motivates this thesis. Section 4.2 is a concise description of the proposed solution. Section 4.3 explains the main challenges related to the problem. Section 4.4 introduces the thesis statement. Section 4.5 details the research questions addressed. Section 4.6 explains the validation methodology used to measure the impact of this work. Finally, Section 4.7 summarizes this chapter.

4.1 Problem Summary

When a new evolution is introduced in the Natixis system, it is necessary to check if there are some unexpected changes in financial trades. This validation occurs in the non-regression process. During that process, three types of test reports are generated for the production and test scenarios of the system. It should be noted that this work only focuses on the Profit and Loss test report (explained in Chapter 2). Thereby, it is from these reports that it is possible to identify the referred differences between those scenarios. The differences are identified by the comparison of the Present Value[1] of the Profit and Loss report. For this, the company is compelled to involve Business Analysts to manually tag those differences and figure out which ones are not expected, by understanding their source, which is defined by the properties of the trades where they occur.

Consequently, that manual process of identifying deviations between the two scenarios and understanding in what dimensions of trades those differences occur can be very time-consuming. Thus, implementing a prototype of a tool that helps Business Analysts in recognition of the differences, in a straightforward and easy way to interpret, represents the main aim of the thesis.

[1]**Present value** is the amount of cash right now of some amount of money in the future. [Far20]

4.2 Proposal

This thesis is aimed to reduce the time of non-regression process by supporting the Business Analysts task of identifying and analysing possible differences in profit and loss values of the trades, after a system evolution.

In order to do this, the thesis works in implementing a prototype of a tool that presents to the user, in an intuitive way, the data, which come from the comparison of the profit and loss values between the reference and test scenarios, organized in such way to represent of which their dimensions supporting the resulting differences of that comparison.

For this, the project aims at developing an algorithm based on Decision Trees for the exploratory analysis of the large dataset of finance calculations. Decision Trees algorithms meet the main requirements of the problem. The following list describes the reasons that support the choice of the model to be based on Decision Trees:

- Decision Trees help in detection and visualization of the relationships and patterns between the attributes of the dataset;
- Some algorithms of Decision Trees can deal with categorical data;
- The data pre-processing step for Decision Trees requires less effort and time compared to other algorithms;
- These type of models are very intuitive and easy to explain;
- Decision Trees can deal with huge datasets and make them simpler.

Thus, the goal is to split the dataset, collected from the Profit and Loss reports which contain all trades and a specific class that indicates if there is a difference or not, into a decision tree for analysis of the trade dimensions where changes occur in both scenarios. The resulting tree diagram used to represent the data should provide to Business Analysts a straightforward interpretation of the source of possible unexpected deviations in trades.

4.3 Fundamental Challenges

The problem mentioned above addresses some challenges that are important be noted. Dealing with high dimensional data as finance calculations brings some issues related to the Data Mining process. Besides, the strict finance context of the problem gives an increased responsibility. Therefore, this work implies an understanding of several concepts, and many decisions will need to be made. Below, it is listed some challenges tackled in the thesis:

1. Recognition of the attributes of the trades, which in addition to being essential in the interpretation of differences by Business Analysts, facilitate the process of splitting the tree.
2. A user-friendly representation of the implemented decision tree, ensuring quick and easy interpretation for Business Analysts.

3. Assurance that the conclusions of the tool are not misplaced and cause Business analysts to make serious mistakes. The algorithm will handle critical information, so the margin of error has to be as short as possible.

4.4 Thesis Statement

In this work, it is admitted that an algorithm based on Decision Trees allows a clear and easy identification to the Business Analysts, during the non-regression testing process, of the expected or unexpected changes in trades after a new system update. Therefore, the following hypothesis is proposed for this thesis:

"Creating an algorithm based on Decision Trees to support Business Analysts in the non-regression testing process in the identification of the possible irregularities after a new system evolution"

4.5 Research Questions

From the thesis statement specified above, some research questions emerge, particularly:

RQ1: *Are Business Analysts able to identify irregularities after a system evolution using a tool that generates a tree of the data?*

RQ2: *Is it possible to create an easy-to-understand decision tree aiming at support Business Analysts and reduce the non-regression testing time?*

RQ3: *Would Business Analysts use an algorithm that generates a tree of the data collected from the non-regression process?*

4.6 Validation Methodology

As previously described, the main goal of this project is to represent, in an understandable way, the resulting data from Profit and Loss report of the non-regression process, to identify the differences in Profit and Loss value between the test and reference scenarios and to associate those disparities to the properties of the trades, when a system upgrade occurs. Therefore, an algorithm based on Decision Trees to represent the data set must be implemented.

The decision tree should be evaluated for its ability to represent the data for Business Analysts in a simple and intuitive way while at the same time ensuring that essential data for interpretation is not lost. Thus, the decision tree representation should be implemented alongside the Business Analysts so they can validate the usability of the prototype. The algorithm implemented should be compared with different approaches to evaluate it in terms of simplicity and straightforwardness of splitting the data. The verification if no data is lost can be done through the help of confusion matrices, showing the improvement of the accuracy while not losing any data.

4.7 Conclusions

The goal of this work is to help Business Analysts during the non-regression process, through the identification of which data dimensions support the differences in profit and loss values of the trades, between the production and test scenarios when a new system evolution occurs. As a result of this thesis, it is expected a prototype of a tool that generates a decision tree of the data collected from the Profit and Loss report during the non-regression testing process.

Chapter 5

Implementation

This chapter details the solution and methodology used to build a prototype of a tool that helps Business Analysts during the non-regression process. Section 5.2 details the data preparation phase of this data mining process. Section 5.3 presents the algorithm description. Section 5.4 covers the methodology used to represent the resulting tree of the algorithm in a user-friendly way. Finally, Section 5.5 draws some conclusions from the solution.

5.1 General Overview of the Solution

The first step in implementing the algorithm based on Decision Trees is to select and prepare the dataset, ensuring that it is suitable in the identification and analysis of the differences between the two scenarios. The next step is to design the algorithm based on Decision Trees adapted and optimized to the resulting dataset, in order to ensure a simple data structure. Afterwards, the generated tree can be represented in a user-friendly way to Business Analysts, to identify the dimensions (attributes and respective values) of the differences.

5.2 Data Preparation

As mentioned before, during the non-regression process, the Profit and Loss reports (explained in Section 2.1.1) are generated, giving rise to the source of the dataset. These reports constitute a huge and complex database of finance calculations. Therefore, the initial stage was the capture of a subset of that database that compares the profit and loss value between the production and test scenarios in order to obtain a dataset that identifies the existence of differences for each trade with a specific set of properties. The following diagram illustrates the procedure for obtaining the dataset to be used in the algorithm.

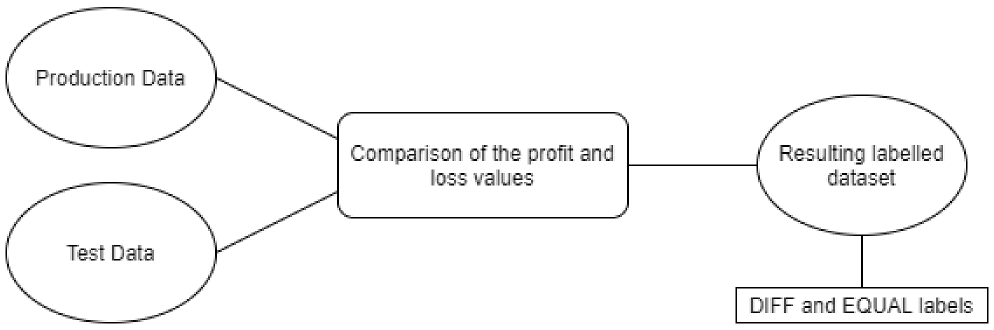

Production Data
Test Data
Comparison of the profit and loss values
Resulting labelled dataset
DIFF and EQUAL labels

Figure 5.1: Diagram that illustrates the procedure for obtaining the dataset

For that, SQL queries were made to the mentioned database that joins the same instances of trades in production and test scenarios and for each of them compares the value of profit and loss indicating whether there are discrepancies in these values. The columns of the results of the initial queries were all properties that characterize the trades, resulting in dozens of attributes in which some of them, according to financial experts, are not relevant in the perception of differences in the non-regression process.

Thus, after consulting the opinion of financial experts and an exploratory analysis of the data (described in Chapter 6), the dataset includes the following attributes: *Type*, *Book*, *CCy*, *Commodity*, *Desk*, *PorS* and *ProducType*. The following list presents a brief description of each one of these attributes that characterize the financial trades.

- **Type:** This attribute refers to the different types of existing Trades (BOND, Money Market, Repo, among others). Each type has its financial meaning.
- **ProducType**: It is a subtype of trade.
- **Book:** A trading Book is the portfolio of financial instruments (equities, debt, commodities, etc) of a bank [Che20a][Ris20b].
- **Desk:** A trading Desk contains several Books. It is where the sale and purchase operations occur [Gan20].
- **CCy:** It is the currency in which the trade is contratualized and can be EUR (euro currency), USD (american dollar), GBP (british pound), among others.
- **Commodity:** A commodity index measures the performance of a basket of commodities, by tracking its price and the investment return [Che20b].
- **PorS:** The PorS attribute indicates if it is a Purchase or a Sell, taking the values P and S, respectively.

In section A of the Appendices, a table with information about the attributes and their values can be found.

Each column of the resulting labelled dataset represents an attribute (a trade property, that is explained above), each row contains the values for each attribute and can represent more than one instance of trades, because there are different trades which have the same values for the selected attributes. So, each row can have samples that have no difference in the comparison of values and, therefore, belong to the *EQUAL* class and samples that, on the other hand, have differences and belong to the *DIFF* class. As each row have a multi-label, that is, have points labelled as *DIFF* and points labelled as *EQUAL* simultaneously, the dataset have three additional columns: an "Equal" column that contains the *EQUAL* points number for each row, a "Diff" column that represents the number of *DIFF* points for each row and finally, to facilitate the calculations in the algorithm, an "Equal+Diff" column that is the total number of samples for each row. The table below shows three possible rows of the resulting dataset.

Type	Book	CCy	Commodity	Desk	PorS	ProducType	Diff	Equal	Equal+Diff
MM	2TAA	EUR	FIXED	2TA	S	EXOFX	0	1	1
FUT	43HV	EUR	INDEX	2TA	P	EXOFX	1	0	1
REPO	641B	USD	LIBOR	2TA	S	EXOEM	1	1	2

Table 5.1: Example of possible rows in the resulting dataset

5.3 Algorithm Description

Based on the literature review, algorithms based on decision trees answer the primary needs of this problem, as described in Chapter 4. This type of algorithms can deal with categorical data (sometimes called nominal data, in which the different categories do not have an intrinsic order), can deal with complex problems providing an intuitive and easy way to represent them and can be customized by the inferences of the experts in the issue. Thus, the implemented algorithm is based on Decision Trees (the concept is explored in 3.1.2) and takes advantage of the entropy and information gain calculations of the ID3 algorithm (explained in 3.1.2.3).

Due to the size and complexity of the dataset, considerable adaptations were made to the classic decision tree algorithm, to optimize the execution time and give priority to the representation of the points labelled as *DIFF*. The following list enumerates the main keys of the proposed algorithm and strategic changes to the classic ID3:

- Due to the extensive data set, the proposed algorithm uses an **iterative approach**, contrasting with the recursive version of ID3. The time complexity of a recursive implementation is much higher (generally exponential) than the time complexity of an iterative one (generally polynomial or logarithmic). For each recursive call, certain values are added to the stack, and for that reason, there is usually more overhead associated and can be very expensive in memory space. Functions that just iterate make no such demands on stack space and can be more efficient where memory is limited [Pat20].

- The **criteria to split the tree** does not follow the usual one used in Decision Trees that consists of dividing the decision node (the attribute) into all its values. The attributes of this dataset have many values. In order to ensure the simplicity of the tree, it was considered not to split each chosen attribute into all its values at once. The heuristic evaluation function chooses the attribute-value pair to split the tree into the selected value and the remaining values joined together for that attribute. In this way, there is no such fast growth in the number of nodes in the tree. The used splitting criteria is explained in Section 5.3.1.

- The algorithm uses a **global array**, called impure candidates' list, that stores the information associated with each possible impure leaf of the tree. An impure leaf is a node with no children, which can still be split and in which not all of its instances are part of the same class. For each impure leaf, it is stored its subset of data, the parent node (if it exists), the best attribute-value pair which splits that leaf and the returned number by the heuristic evaluation function. Thus, at each iteration is selected, from that array, the node to split the tree based on the highest number retrieved by the heuristic evaluation function. The iterative process of the algorithm is possible through this global array.

- The proposed **heuristic evaluation function**, that finds the best feature to split the tree, not only focuses on reducing the degree of impurity of the leaf nodes through the Information Gain but also on minimizing the depth of the tree. This measure aims to provide a better reading of the tree, reducing its total depth. The function is detailed in Section 5.3.3.

- As mentioned above, each row of the dataset does not represent just one trade. More than one trade can have the same values of the selected attributes. Thus, each row can have a multi-label, that is, trades labelled as *DIFF* and trades labelled as *EQUAL* values. The main goal is to identify the dimensions of the *DIFF* trades. Thus, when a leaf of the tree that cannot be further split has DIFF and EQUAL instances simultaneously, by default, the *DIFF* label is assigned with a degree of purity given by the number of *DIFF* instances divided by the total number of instances of that leaf.

- Since the main goal is to represent the dimensions of the trades labelled as *DIFF*, during the entropy calculation process, for each attribute, the values (if any) that only present observations labelled as *EQUAL* (pure values) are joined as a single value, called *"EQUAL VALUES"*, saving time in entropy calculations and merging the leaves that are not so important to see represented. This optimization is also justified by the fact that the data set is extremely unbalanced. As shown in the pie chart below, 99.8% of the dataset points are labeled *EQUAL* and 0.2% *DIFF*.

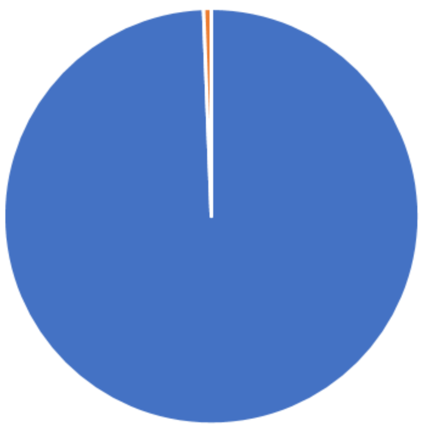


Figure 5.2: Pie chart that demonstrates the percentage of observations for each class

- The process of building the tree is iteratively repeated until one of these stopping condition occurs:
  - The global purity of the tree (explained in 5.3.4) is 1.
  - The impure candidates' list is empty.

The algorithm works as follows:

1. Calculates the entropy (explained in 5.3.2):
   (a) For all possible pairs formed for each attribute A and one of its values a that contains at least one point labelled as *DIFF*, $H(C|(A, a_{withdiff}))$
   (b) For each attribute A and all its values that only have points labelled as *EQUAL* joined, $H(C|(A, a_{equal}))$;
2. Calculates the Information Gain for each (A, a) pair that is the difference between the Entropy of the current set, $H(C)$, and the entropy for each pair, $H(C|A, a)$;
3. The (A, a) pair is chosen by the heuristic evaluation function (described in 5.3.3). Thus, the attribute A is the root of the tree, and the value a is the branch of the root node;
4. Initialize the impure candidates' list with the whole data, its (A, a) winning pair and the respective number returned by the heuristic evaluation function;
5. The following steps of building the tree are iteratively repeated until one of the mentioned stopping condition occurs:
   (a) Extract from the impure candidates' list the best pair candidate, which is the one with the highest return of the heuristic evaluation function.

(b) According to the selected pair, the new candidates are created. The creation process of new candidates is described in 5.3.1;

(c) For each of the new candidates, the purity is assessed:

- If the purity is 1, that is, if the number of *DIFF* or *EQUAL* points is equal to the total number of points of the candidate subset, the new candidate becomes a leaf of the tree with the class *DIFF* or *EQUAL*, respectively.
- Otherwise, if the purity is less than 1:
  - Repeat the steps 1 and 2 to find out the best pair to split the tree from the new candidate.
  - The information about the new candidate is added to the impure candidates' list.

The impure candidates' list and the tree that results from the first iteration are shown below:

Impure candidates' list
Impure leaf: $a1$
Subset: $a1 \subseteq A$
Best pair candidate to split: $(B, b1)$
Value gained if splitted: x
Impure leaf: $\overline{a1}$
Subset: $a1 \nsubseteq A$
Best pair candidate to split: $(C, c1)$
Value gained if splitted: y
Subset: $a1 \nsubseteq A$
Best pair candidate to split: $(A, a2)$
Value gained if splitted: z

Table 5.2: The information stored by the impure candidates' list in first iteration

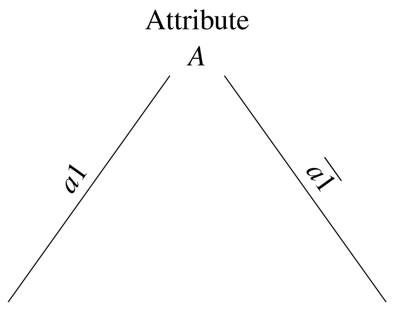


Figure 5.3: Tree after first iteration

5.3.1 Splitting Criterion

In this algorithm, the heuristic evaluation function doesn't choose just an attribute to split the tree in all its values. The heuristic evaluation function chooses the attribute-value pair to split the tree in the value selected and in the remaining values joined of that attribute (shown in Figure 5.3). Thus, at each iteration, three types of candidates can be added to the impure candidates' list, as shown in Table 5.2. From this, the tree can be split in three different ways, depending on the "type" of the pair that is selected from the impure candidates' list. Considering the list present in Table 5.2, the following situations can be assumed:

1. Assuming that $(A, a1)$ is selected, it is split in its best candidate $(B, b1)$ and the information about the new possible candidates is added to the impure candidates' list, as shown in Table 5.3. The resulting tree is represented below:

Impure candidates' list
Impure leaf: $\overline{a1}$ Subset: $a1 \nsubseteq A$ Best pair candidate to split: $(C, c1)$ Value gained if splitted: y
Impure leaf: $b1$ Subset: $b1 \subseteq B$ Best pair candidate to split: $(D, d1)$ Value gained if splitted: x
Impure leaf: $\overline{b1}$ Subset: $b1 \nsubseteq B$ Best pair candidate to split: $(E, e1)$ Value gained if splitted: t
Subset: $a1 \nsubseteq A$ Best pair candidate to split: $(A, a2)$ Value gained if splitted: z
Subset: $b1 \nsubseteq B$ Best pair candidate to split: $(B, b2)$ Value gained if splitted: w

Table 5.3: The information stored by the impure candidates' list when is selected the leaf $a1$

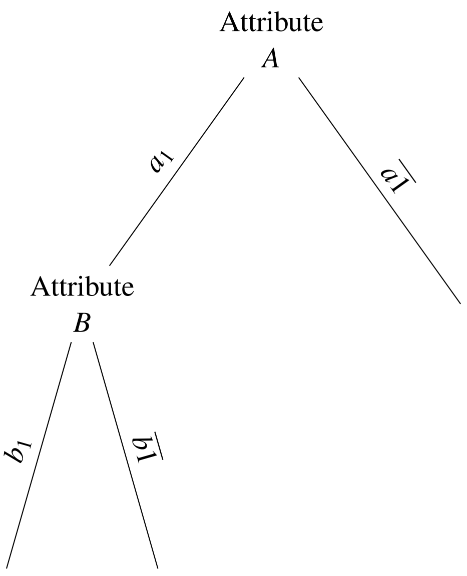


Figure 5.4: Resulting tree when the A attribute with a_1 value is selected

2. Assuming that $(A, \overline{a1})$ is selected, it is split in its best candidate $(C, c1)$ and the information about the new possible candidates is added to the impure candidates' list, as shown in Table 5.4. The resulting tree is represented below:

Impure candidates' list
Impure leaf: $a1$ Subset: $a1 \subseteq A$ Best pair candidate to split: $(B, b1)$ Value gained if splitted: x
Impure leaf: $c1$ Subset: $c1 \subseteq C$ Best pair candidate to split: $(D, d1)$ Value gained if splitted: t
Impure leaf: $\overline{c1}$ Subset: $c1 \nsubseteq C$ Best pair candidate to split: $(E, e1)$ Value gained if splitted: y
Subset: $c1 \nsubseteq C$ Best pair candidate to split: $(C, c2)$ Value gained if splitted: w

Table 5.4: The information stored by the impure candidates' list when is selected the leaf $\overline{a1}$

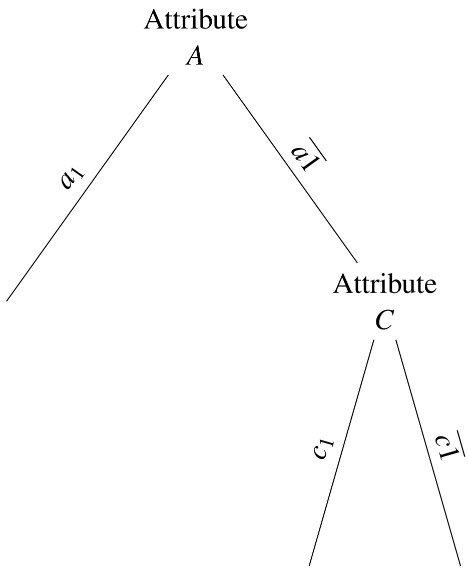


Figure 5.5: Resulting tree when the A attribute with $\overline{a1}$ value is selected

3. Assuming that $(A, a2)$ is selected, the branch that contains $\overline{a1}$ is updated to $\overline{a1} \cap \overline{a2}$ and a new branch $a2$ is created. The information about the new possible candidates is added to the impure candidates' list, as shown in Table 5.5. The resulting tree is presented below:

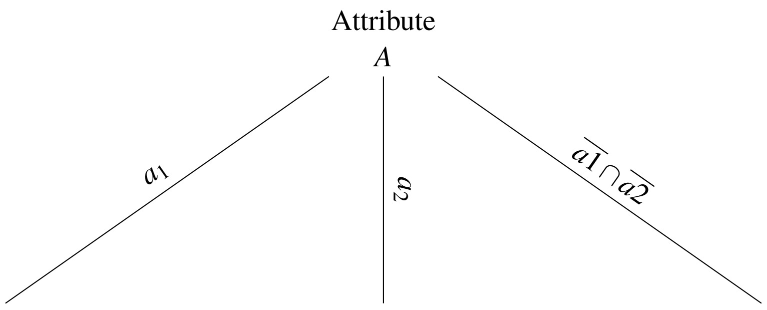


Figure 5.6: Resulting tree when a new value, a_2, of the A attribute is selected

Impure candidates' list
Impure leaf: $a1$ Subset: $a1 \subseteq A$ Best pair candidate to split: $(B, b1)$ Value gained if splitted: x
Impure leaf: $a2$ Subset: $a2 \subseteq A$ Best pair candidate to split: $(D, d1)$ Value gained if splitted: t
Impure leaf: $\overline{a1} \cap \overline{a2}$ Subset: $(a1, a2) \nsubseteq A$ Best pair candidate to split: $(E, e1)$ Value gained if splitted: y
Subset: $(a1, a2) \nsubseteq A$ Best pair candidate to split: $(A, a3)$ Value gained if splitted: w

Table 5.5: The information stored by the impure candidates' list when is selected the leaf $a2$

5.3.2 Adapted Entropy and Information Gain Calculations

The developed algorithm uses the Information Gain concept of the ID3 algorithm, that is defined, as indicated previously in Subsection 3.1.2.3, by the difference between the Shannon entropy of the parent node and the conditional entropy that is the sum of the entropy of the resulting child nodes, multiplied by their probabilities [dJG14]. In the implemented algorithm, it is measured the entropy of a pair formed by an attribute and one of its values. Thus, the adapted calculation of the information gain of the attribute A and a value a is:

$$IG(C,(A,\ a)) = H(C) - H(C|(A,a)) \tag{5.1}$$

Where:

H(C) is defined in 3.1

$$H(C|(A,a)) = \sum_{j \in a,\overline{a}} p(A_j) \times [-\sum_{i=1}^{N} p(c_i|A_j) \times \log_2(p(c_i|A_j))] \tag{5.2}$$

- A_j - represents the possible value j for the Attribute A, which can take just two values: a (the attribute A takes the value a) and $\overline{a}$ (the attribute A takes all its remaining values that not includes a)
- N - the number of possible values for the class C. For this specific dataset, $N = 2$ ($DIFF$ and $EQUAL$ classes)
- c_i - the possible value i for the class C (i can take the $EQUAL$ and $DIFF$ values)
- $p(A_j)$ - the proportion of the number of instances with value j to the number of instances with attribute A (probability of occurring j)
- $p(c_i|A_j)$ - the proportion of the number of instances with value j in class c_i to the number of instances with attribute A (probability of occurring j with c_i)

5.3.3 Heuristic Evaluation Function

This algorithm distinguishes itself by the heuristic evaluation function that combines the depth cost with information gain. The goal is to choose the candidate that offers the highest Information Gain, while not contributing to a significant increase in the total depth of the tree, to ensures the creation of an easier-to-read tree. Thus, the heuristic evaluation function (*HEV*) is defined as:

$$HEV(i) = \frac{IG_i}{\log DC_i} \tag{5.3}$$

Where:

- IG_i - the information gain of splitting the attribute-value pair i
- DC_i - the depth cost of splitting the attribute-value pair i

The depth of a node is the number of edges from the node to the tree's root node. So, it is considered that the total depth of the tree is:

$$\sum_{i=1}^{N} d(x_i) \tag{5.4}$$

Where:

- $d(x_i)$ - represents the depth of the leaf i
- N - the total number of leaves of the tree

The following figure shows an example of a tree, where each node indicates its corresponding depth. For that tree, the total depth is seven.

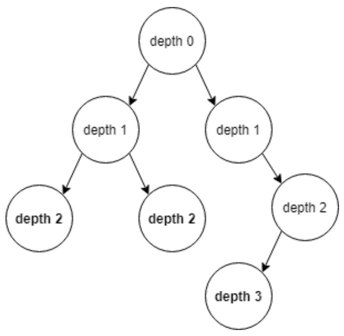


Figure 5.7: Example of a tree with the depths of each node.

Before splitting the tree, it is possible to know the depth cost of choosing the leaf, as shown below. If the leaf is split the first time, its depth cost is:

$$DC_i = (d_i + 1) \times n_i - d_i \tag{5.5}$$

Where:

- d_i - the depth of the parent node of the node i
- n - the number of child nodes to be expected after splitting the node i that is 2

If a leaf was already split, its depth cost is:

$$DC = (d_i + 1) \tag{5.6}$$

Where:

- d_i - the depth of the parent node of the node i

5.3.4 Global Purity of the Tree

Global purity of the tree is the weighted sum of the leaves purity divided by the total number of instances of the dataset, where the weight is the number of instances of that leaves, which formula is presented below:

$$GP = \sum_{i=1}^{N} \frac{p_i \times s_i}{l} \tag{5.7}$$

Where:

- N - the number of leaves
- p_i - the purity of the leaf i
- s_i - the number of instances of the leaf i
- l - the length of the data (total number of instances in dataset)

To summarize the description of the algorithm, its pseudocode is shown below:

Algorithm 1: Adapted Iterative Dichotomiser 3

```
Input: Dataset
Output: Decision Tree
ILL ← impure leaves list
root pair ← returned by the function FindWinnerPairAttributeValue(whole data)
Initialize the ILL with all data, the root pair and its $IG/\log(DC)$
while ILL is not empty and current tree purity < 1 do
    $(A, a1)$ ← Extract from the ILL the pair with the highest $IG/\log(DC)$
    Create a node for the $A$ attribute (if it not exists)
    Create the branch for the $a1$ values
    Create the branch for the $\overline{a1}$ value (if it exists)
    forall Newly created Branches do
        purity ← purity calculation
        if purity is 1 or cannot be further divided then
            if Quantity of DIFF observations > 0 then
                Create a DIFF Leaf
            else
                Create an EQUAL Leaf
            end
        else
            $(B, b1), x$ ← returned by the FindWinnerPairAttributeValue(branch subset)
            [A = a1, (B, b1),x] ← New info added to ILL
        end
    end
    if If there are values for A attribute that have not yet split the tree and the pair
      extracted does not include all the remaining values of A then
        $(A, a2), y$ ← returned by the FindWinnerValue(subset, A)
        [A, (A, a2),y] ← New info added to ILL
    end
    current tree purity ← Calculate global purity of the tree
end
Function FindWinnerPairAttributeValue(data):
    Calculate the information gain and the depth cost for all possible pairs of data
    return pair with the highest $IG/\log(DC)$ and $IG/\log(DC)$
Function FindWinnerValue(data,attribute):
    Calculate the information gain and the depth cost for all values of the attribute
    return value with the highest $IG/\log(DC)$ and $IG/\log(DC)$
```

It should be noted that it was used the *Any Python Tree Data* package to help in the construction of the tree data structure.

5.4 Tree Representation

After generating the decision tree using the algorithm described in the previous section, it was necessary to represent it in a user-friendly way. Thus, the Business Analysts can, intuitively, analyze the differences, to know in which dimensions they occur and to understand whether they are expected or not.

This part was developed using D3.js, a JavaScript library, to manipulate documents based on data and to allow their visualization, through powerful visualization mechanisms. From the package *Any Python Tree Data* already mentioned, it was possible to export the resulting tree to a JSON file. In section B of the Appendices, an excerpt from the JSON file generated can be found. Through that file, the D3.js library can read the tree and representing the nodes dynamically and interactively in web browsers.

From there, some functionalities suggested by Business Analysts were implemented, in order to build a prototype of a tool that allows, through the tree representation, a summarization of the data to distinguish between the dimensions of the differences and the non-differences. Based on the opinion of Business Analysts, who will be the end-user, the following features have been implemented:

- Colour assignment to nodes according to the purity range and the class to which they belong (green for pure *EQUAL* nodes, red for pure *DIFF* nodes, yellow for the nodes that have purity less than 0.5 and orange for the ones that have purity more significant than 0.5).
- The function of expanding and hiding the nodes. By clicking on the node, it can be expanded or hidden. As the tree has many nodes, this feature becomes essential for better visualization.
- Boxes with information appear when the users hover over a node. This information is about the specific node, namely the name of the attribute it represents, its total number of instances and the number of DIFF and EQUAL instances.
- Adaptation of the node size to the number of instances it has.

The results of this prototype will be demonstrated in Chapter 6.

5.5 Conclusions

This chapter described the details regarding the implementation of the tool. It details the initial and crucial stage of gathering and preparing the dataset. Besides, the keys that characterize the implemented algorithm were described and justified. However, there are still possible improvements.

Finally, the methodology used to represent the tree, more conveniently and easily for Business Analysts, was presented.

Chapter 6

Results

This chapter intends to show the results obtained in the project and how the implemented algorithm was validated. Section 6.1 presents the exploratory data analysis made to select the attributes of the dataset. Section 6.2 describes the results that were obtained from the performed validation process in the algorithm and compares the final algorithm with some approaches implemented in the beginning. Section 5.4 provides an overview of the final representation of the tree and the validation by the Business Analysts. Section 6.4 contains some final comments over the methodology used and the results obtained.

6.1 Exploratory Data Analysis

As described in Chapter 5, the first step of the construction of the algorithm was to define which columns have to be part of the dataset. Trades have multiple properties, but according to the Business Analysts, some of them are not important in recognising the differences in the non-regression process. Thus, the choice was based on the experience and domain knowledge of the analysts, as well as on the results obtained through the Weight of Evidence (WOE) and Information Value (IV) feature selection techniques.

These techniques were chosen to try to understand which attributes are more related to the target class. WOE describes the relationship between a value of an attribute and a binary dependent variable, that are the target classes. IV measures that relationship's power [San20].

Considering the dataset used in the project whose target classes are the *DIFF* class and the *EQUAL* class, the **Weight of Evidence** of a value *j* is defined as [Cor14]:

$$WOE(j) = \ln\left(\frac{f_{EQUAL}(j)}{f_{DIFF}(j)}\right) \tag{6.1}$$

Where:

1. $f_{EQUAL}(j)$ - Percentage of $EQUAL$ observations that fall into value j
2. $f_{DIFF}(j)$ - Percentage of $DIFF$ observations that fall into value j

The **Information Value** of an attribute i is defined as:

$$IV(i) = \sum_{j \in S} \frac{f_{EQUAL}(j) - f_{DIFF}(j)}{100} * WOE(i,j) \tag{6.2}$$

Where:

1. j - Value of the attribute i
2. S - Represents the set of values for attribute i
3. $WOE(i,j)$ - Weight of Evidence of the value j for the attribute i

Although this model does not aim to predict, these techniques help to perceive the relationship of the variables with the class it defines. According to the conventional scale when [Lis20]:

- IV < 0.02, the attribute is useless for modeling (separating the *DIFF* from *EQUAL* classes);
- 0.02 < IV < 0.1, the attribute has an weak relationship to the *DIFF/EQUAL* odds ratio;
- 0.1 < IV < 0.3, the attribute has a medium strength relationship to the *DIFF/EQUAL* odds ratio;
- 0.3 < IV < 0.5, the attribute has a strong relationship to the *DIFF/EQUAL* odds ratio;
- IV > 0.5, can be a suspicious relationship.

The Table 6.1 shows the information value calculated for each attribute. It should be noted that these attributes already come from a pre-selection by Business Analysts, as mentioned in Chapter5. Some of the results contradict the expectations of the experts.

Attribute	**Information Value**
PorS	0.016
CCy	0.266
ProducType	0.458
Desk	1.763
Dealid	0.013
Book	0.952
Commodity	0.422
StructureId	0.015
Type	1.329

Table 6.1: Information Value for each attribute

According to the scale, the attributes PorS, StructureId and Dealid must be disregarded for the final dataset. However, the PorS attribute was maintained, after discussion with financial experts.

They considered that attribute fundamental to understand the differences in trades. The high Information Value obtained for the Desk, Book and Type attributes can be justified by the fact that they present several different values. Information value increases as number of values increases for an independent variable. With this, the resulting dataset includes the following attributes: *PorS*, *CCy*, *ProducType*, *Desk*, *Book*, *Commodity* and *Type*.

6.2 Algorithm Validation

Although the implemented algorithm does not have the goal of predicting results, but rather explain the dataset in order to provide a quick interpretation of the results of the non-regression process, the concept of confusion matrix was used to validate the decision tree. This concept is used in classification problems as predictive models to measure the performance between the predicted outputs and the real ones. Although, the implemented algorithm is a classification problem since it tries to group the data under a specific criterion and according to similar features and attributes [Bri20]. Thus, through the help of the confusion matrices concept, it is measured at each iteration the following values:

Adapted Confusion Matrix	**DIFF instances in initial data**	**EQUAL instances in initial data**
DIFF instances captured by the tree	True Positives	False Positives
EQUAL instances captured by the tree	False Negatives	True Negatives

Table 6.2: Confusion Matrix

Where:

- **True Positives:** *DIFF* instances that are correctly classified.
- **False Positives:** *EQUAL* instances included in leaves classified as *DIFF*.
- **True Negatives:** *DIFF* instances included in leaves classified as *EQUAL*.
- **True Positives:** *EQUAL* instances that are correctly classified.

By calculating these values at each iteration, it was possible to represent the graphs below. These graphs validate some fundamental requirements in the result of the tree, namely the non-existence, during the whole process, of false negatives (as shown in Figure 6.3). It is essential to transmit to the Business Analysts all observations that present any difference, even if it is minimal. On the other hand, the existence of false positives is not so relevant, and it is justified by the fact that each row of the used dataset can have more than one observation. As mentioned in Chapter 5, some distinct trades can be selected for the same row since they have the same dataset attributes. Thus, one row can have observations classified as *EQUAL* and *DIFF* simultaneously, and it cannot be further split. In those cases, the resulting leaf is impure, but it is classified as *DIFF*, to ensures that no *DIFF* observation escapes. It is possible to notice in Figures 6.2 and 6.4, the decrease in false positives and, in contrast, the growth of true negatives, respectively, demonstrating the good performance in the construction of the tree.

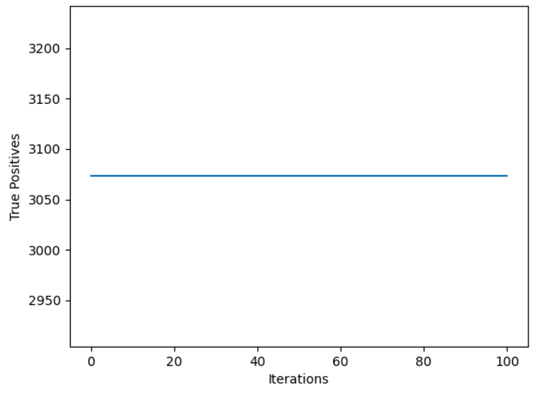


Figure 6.1: Evolution of the number of **True Positives**

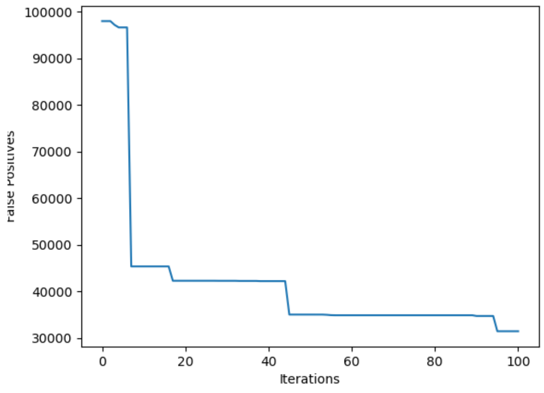


Figure 6.2: Evolution of the number of **False Positives**

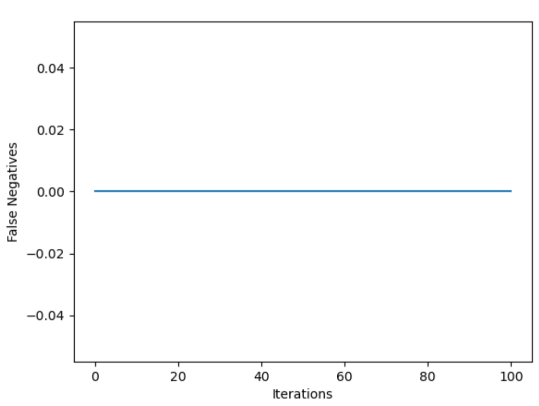


Figure 6.3: Evolution of the number of **False Negatives**

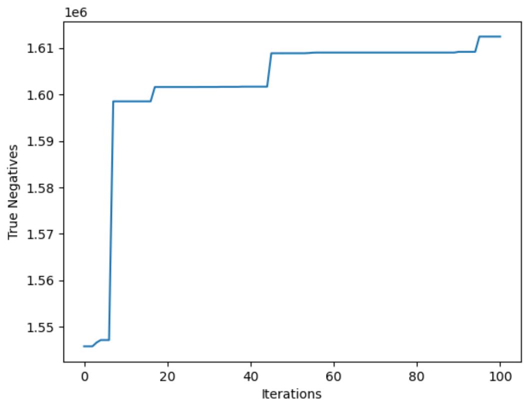


Figure 6.4: Evolution of the number of **True Negatives**

The following table presents the final results of the Confusion Matrix, which validates the imperative condition of not having false positives, that is, not having *DIFF* observations labelled as *EQUAL*.

Adapted Confusion Matrix	**DIFF instances in initial data**	**EQUAL instances in initial data**	
DIFF instances captured by the tree	TP = 3073	FP = 31445	Total = 34518
EQUAL instances captured by the tree	FN = 0	TN = 1612422	Total = 1612422
	Total = 3073	Total = 1643867	

Table 6.3: Final results of the Confusion Matrix

6.2.1 Comparison between different approaches

As explained in Chapter 4, some changes were applied to the classic ID3 algorithm. However, throughout the implementation, other approaches have experimented, and the next graphs justify and validate the choice of the final approach.

The first approach used for comparison differs from the final approach in the criterion it uses to split the tree. In the first approach, the attribute that is chosen to split the tree is divided into all its values. So each of them will represent a new branch of the tree. Its heuristic evaluation function only uses the Information Gain to select the winner attribute, that is, the one that has the highest number of Information Gain. This approach, like the final one, uses an iterative strategy to build the tree. The following table 6.4 summarizes the main differences between the two approaches.

	First Approach	**Final Approach**
Split Criterion	Choose an attribute to split the tree into all its values	Choose an attribute-value pair to split the tree
Heuristic Evaluation Function	IG	IG / log(Depth)

Table 6.4: Differences between the two approaches

The decision tree validation measures chosen are based on the degree of simplicity of the tree, in order to assess its interpretability for the users. Thus, the evolution of total depth degree and number of leaves during the tree construction process was evaluated and traced for both approaches. The global purity of the tree over the time was also evaluated, to assess which of the two approaches, with the same stopping conditions, achieves a higher degree of purity. The resulting decision tree aims to inform the BA's in which types of trades do the differences occur, and in this sense, it is essential to give them accurate information. The accuracy of this explanatory model can be assessed by the global purity of the tree (explained in Chapter 5). The more homogeneous (higher purity degree) are the leaves, the results shown to the users become more accurate. It is also shown the comparison of the running time of both algorithm approaches.

It was used the same dataset in both approaches, in order to understand the impact of the different split criterion and the heuristic evaluation function. The total number of instances of the dataset is 1 646 940.

6.2.1.1 Total Depth Degree

As mentioned earlier, the total depth of a tree can be considered to assess the readability of a tree. The higher depth of the tree, the more difficult it becomes to read the tree. The graphs below (Figures 6.5 and 6.6) shows the evolution of the total depth of the tree over time. From them, traced with the same scale, it can be seen that in the first approach, the total depth is much greater than in the final approach.

This discrepancy is due, on the one hand, to the tree split criterion, that in the first approach only the "best" attribute is chosen, which in turn is split into all its values, while in the final approach the "best" pair attribute-value is chosen. On the other hand, the final approach takes into account the cost of depth when selecting the "best" pair attribute-value in the heuristic evaluation function.

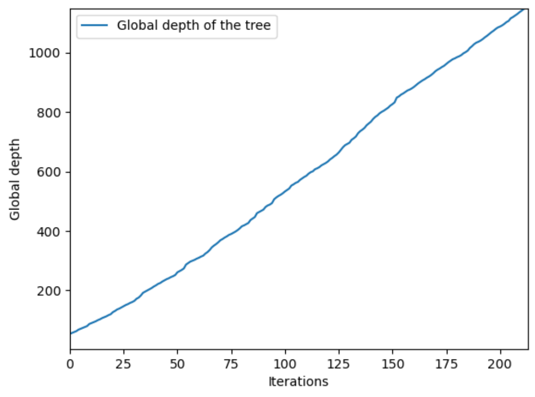


Figure 6.5: **First approach:** Evolution of the total depth of the tree

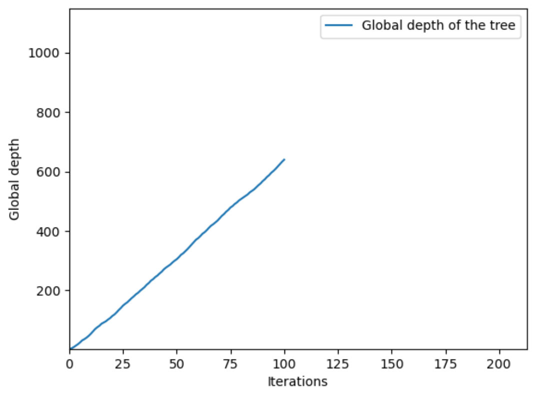


Figure 6.6: **Final approach:** Evolution of the total depth of the tree

Thus, the final approach ensures a lower total depth of the tree, contributing to a better interpretation of it.

6.2.1.2 Leaves Number

The leaves number of a tree can also be indicative of its simplicity. The table 6.5 shows that the first approach, with the same stopping conditions, has more than triple of the leaves number of the final approach. The charts of the Figures 6.7 and 6.8 also shows that in the final approach, the growth of the leaf tree is much less than in the first approach.

The main reason for this difference is the tree's split criterion. In the first approach, it is observed that in the first iteration the tree already has almost 50 leaves, since the attribute chosen to be the root of the tree has 50 different values, many of them without relevant information and that does not offer much when dividing the tree. In contrast, in the final approach, the attribute is chosen together with one of its values which brings more information to the tree, avoiding a sharp growth in the number of leaves. Thus, this measure also proves the simplicity of the implemented algorithm.

Leaves number of the final tree	
First Approach	Final Approach
331	102

Table 6.5: Total number of leaves of the final tree for each approach

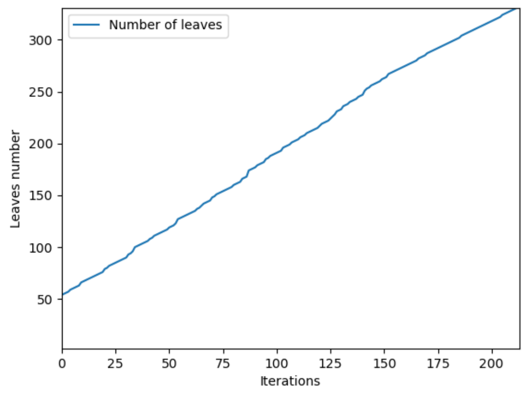


Figure 6.7: **First approach:** Evolution of the number of leaves

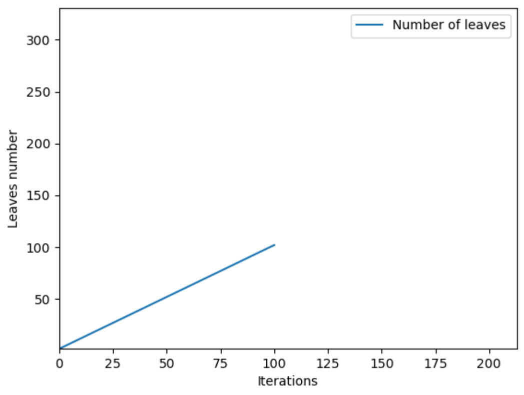


Figure 6.8: **Final approach:** Evolution of the number of leaves

6.2.1.3 Purity Degree

It is desired to guarantee the simplicity of the tree at the same time that it provides accurate information. The global purity of the tree can be interpreted as a measure of the accuracy and quality of the tree. The calculation of global purity is presented in Section 5.3.4. From the graphs presented below (Figures 6.9 and 6.10), it can be seen that the first approach achieves a greater degree of the global purity. However, in the final approach, the tree reaches a higher purity in significantly less time than the first approach. This fact proves that the tree split choices in the final approach are more efficient in the first approach.

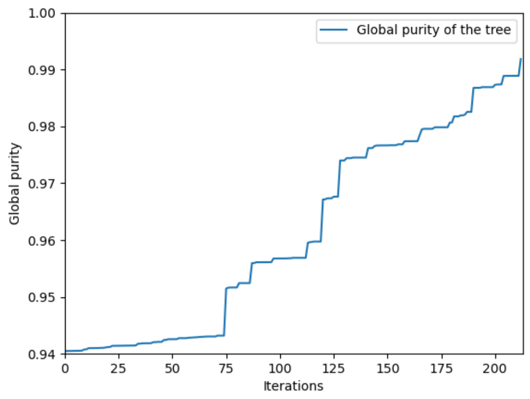


Figure 6.9: **First approach:** Evolution of the global purity of the tree

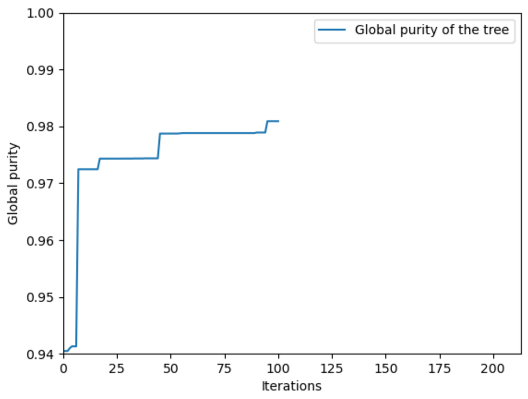
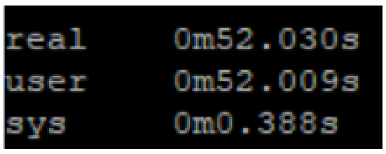


Figure 6.10: **Final approach:** Evolution of the global purity of the tree

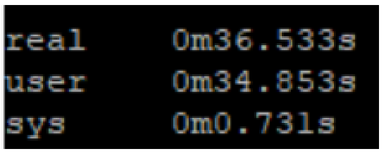
6.2.1.4 Running Time

In addition to the resulting tree having to be presented clearly, one of the main requirements to be evaluated is the running time of the algorithm. The tree must reach the Business Analysts in a timely manner. As we are dealing with a huge dataset, with more than one million instances, some changes were done to achieve this, that were mentioned in Chapter 5. For example, during the entropy calculation, and since the dataset has much more observations with *EQUAL* label than with *DIFF*, the values of an attribute that only have observations labelled as *EQUAL* are merged into a single value, saving time on calculations. In the figures 6.11 and 6.12 it is possible to observe the running times for both approaches. The final one is faster than the first one, which proves that the split criterion of the tree construction used for the final version not only ensures a greater simplification of the tree but it is also more efficient in terms of time. In even bigger data sets, the discrepancy between the two approaches may be higher.



```
real    0m52.030s
user    0m52.009s
sys     0m0.388s
```

Figure 6.11: **First approach:** Running time



```
real    0m36.533s
user    0m34.853s
sys     0m0.731s
```

Figure 6.12: **Final approach:** Running time

6.3 Tree Representation

As previously mentioned, the final result of the project is a prototype of a tool that represents the generated tree in an interactive and dynamic way. The final user, the Business Analysts, was involved in the whole process, by giving feedback about the requirements that the tool should have.

The figure below illustrates the aspect of the implemented prototype. As can be seen, each node has a different colour that depends on the purity and the class which belongs. As previously mentioned, each row in the dataset can contain several instances of trades in which some represent differences and others do not. Thus, some nodes that still have *EQUAL* and *DIFF* instances and are not able to be divided anymore are, by default, labelled as *DIFF* leaves, since it is imperative to deliver to the Business Analysts all dimensions where differences (*DIFF* instances) occur. For pure leaves, the green colour is assigned to label the *EQUAL* class and the red colour for the class *DIFF*. For impure leaves, the orange colour is assigned when the number of *DIFF* instances is more than 50% of the node and the yellow colour for the rest. In this way, the user can quickly interpret where differences are present.

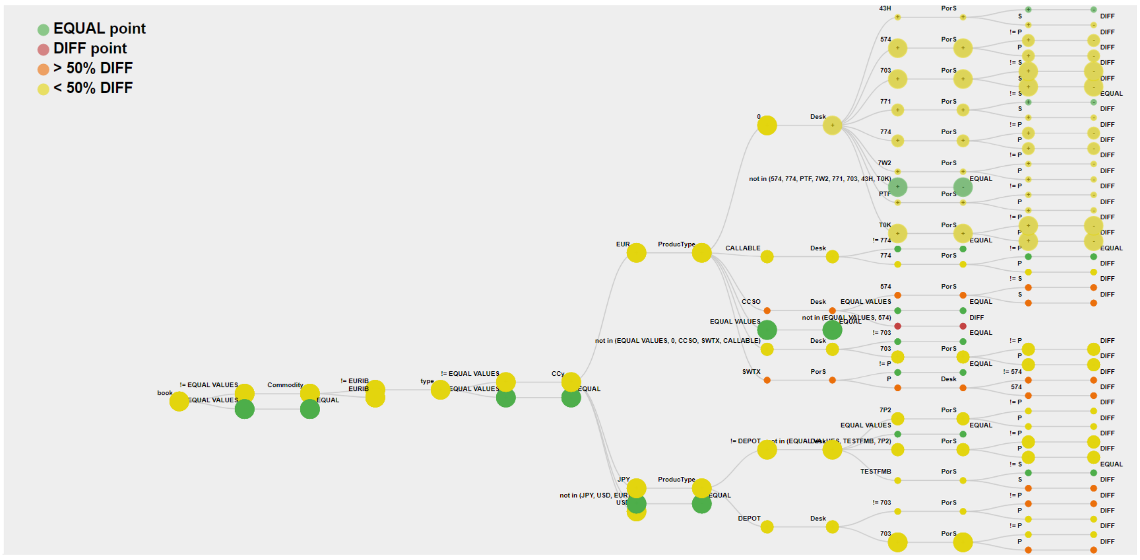
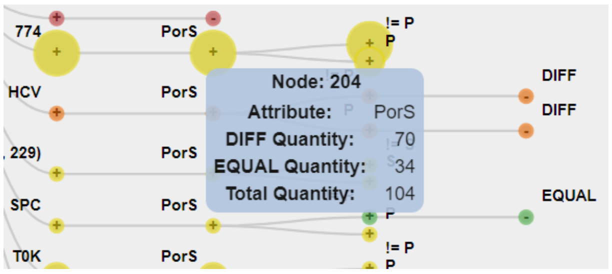


Figure 6.13: The tree representation using D3.js library

Figure 6.14 shows the information box that appears when the user hovers over a node. With that box, it is possible for the user to understand the type of attribute and the number of *DIFF* and *EQUAL* instances it supports.



Node: 204
Attribute: PorS
DIFF Quantity: 70
EQUAL Quantity: 34
Total Quantity: 104

Figure 6.14: Information box when the user hovers over the node

6.4 Conclusions

This section detailed the exploratory data analysis done in the data preparation and the validation process used to evaluate the algorithm. It also presents the tool prototype for representing the tree. The results obtained in the comparison between the final approach and the one that differs in the split criterion and the heuristic evaluation function show that the final approach ensures a more considerable measure of simplicity and effectiveness in terms of time.

Chapter 7

Conclusions and Future Work

In order to check if there was any setback in the system, the process of non-regression is an essential process for the proper functioning of the Natixis system. The results that come from this process have to be analyzed by Business Analysts to understand if there is any regression in the financial trades that is expected or not. This manual process is very time-consuming, so it is a concern for the company. Thus, it was developed a prototype that supports the Business Analysts on the interpretation of those results, aiming at reducing the time required in the non-regression process of the company.

In line with this global objective, in this thesis, it was first explained the background concepts of non-regression and data mining process. Afterwards, it was described the current state-of-the-art regarding useful algorithms that help in summarizing and interpreting complex and high-dimensional data sets. Later, it was defined the problem as well as the proposed solution and what this work aims to accomplish. Following, it was explained the implementation details of the solution described previously.

This chapter wraps this document, reflecting on the conclusions of this work. Section 7.1 describes the main difficulties faced during the development of the thesis. Section 7.2 enumerates the main contributions of this work. Finally, Section 7.3 introduces some relevant topics to enhance the developed solution.

7.1 Main Difficulties

There were several difficulties while researching and developing this algorithm. The first difficulty felt was in preparing the data. The data set comes from the financial calculations made in the profit and loss test. Numerous attributes characterize the trades, and the first challenge was to understand which ones should be part of the data set to respond to the final objective.

Secondly, one of the biggest challenges and difficulties of this project was to understand which data mining model would deal with the type of dataset variables and be suitable for the final objective, to help business analysts in the analysis and interpretation of differences found in the non-regression process. Finally, the representation of the tree in an intuitive way that meets the main requirements of business analysts was another difficulty.

7.2 Main Contributions

The main contributions of this project are the development of a prototype of a tool that supports Business Analysts during the non-regression process. The implemented prototype, by assisting in analysis on test results, will help Business Analyst in identifying the differences between the production and test scenarios and their causes. This factor results in a considerable gain in efficiency and a decrease in the required working hours, once the workload on Business Analysts is reduced.

Thus, the main outcomes of that project are the reduction of the non-regression test time and the consequent decrease of the time-to-market for functional evolutions.

7.2.1 Summary Research Questions and Hypothesis

After detailing the how the algorithm and the tool prototype was developed as well as the results that were obtained during the validation, the hypothesis and research questions detailed in Chapter 4 can be answered.

RQ1: *Are Business Analysts able to identify irregularities after a system evolution using a tool that generates a tree of the data?*

The implemented decision tree allowed, from an exploratory analysis of the dataset, to detect and visualize relationships between its variables with the *DIFF* and *EQUAL* classes, clearly showing in which dimensions of the data these possible irregularities occur. By calculating the confusion matrix, it was possible to prove that the resulting tree identifies all instances that are labelled as *DIFF*, with False Negatives never existing, one of the fundamental requirements of this model. Besides, the implemented prototype allows Business Analysts to recognize, effectively, the dimensions of the data supporting the differences.

RQ2: *Is it possible to create an easy-to-understand decision tree aiming at support Business Analysts and reduce the non-regression testing time?*

In Chapter 5 are presented measures that assess simplicity and, therefore, the ease of interpreting the generated tree. In comparison with another approach, it is clear that the generated tree, even though the dataset is huge, has a relatively low degree of depth, as well as its number of leaves, which proves its power of reading and interpretability. With this, the Business Analysts can identify, efficiently, in which properties of trades do the differences occur after a system evolution and, consequently, reduce the non-regression time.

RQ3: *Would Business Analysts use an algorithm that generates a tree of the data collected from the non-regression process?*

Business Analysts followed the entire process of building the algorithm and implementing the prototype to represent the tree. They mentioned from the beginning to the end of work, important requirements and aspects to be taken into account and improving, showing satisfaction with the final result.

7.3 Future Work

In the future, improvements and additional functionalities may be applied to the implemented algorithm and the tool prototype.

One of the features that will be relevant in the construction of the tree is the introduction of the pruning technique to be applied to the leaves of the tree which do not identify differences. These technique aims to reduce the size of the decision tree and, consequently, the complexity of it, one of the keys to a correct interpretation of the results by the Business Analysts.

Another aspect that can be improved is generalizing the algorithm to cover all the tests that are done in the non-regression process, such as Stress Test or Hedge Test, and not just the Profit and Loss test. The attributes chosen to constitute the dataset can also be changed, to improve the generalization that is made by the tree.

It is also possible to improve the representation prototype by adding the possibility for the user to customize the maximum global purity value of the tree. Thus, the user manipulates this stop condition and personalizes the tree based on its level of confidence. For more detailed information, they can put a high value. For a more general view of the tree, they can set a lower value, reaching one of the stop conditions more quickly.

When a change is made to the system, the non-regression process does not aim only at identifying differences in calculations linked to trades between the test and production scenarios. This process also aims to realize which trades have been lost and added in both scenarios. In the future, the tree could consider these cases and specify in which scenario they occur.

Finally, the tool prototype has to be validated by a higher number of Business Analysts and incorporated in Natixis system.

Appendix A

Dataset Variables

The following table indicates some information about the dataset variables used in chapter 5, which is particular to the comparison of two specific scenarios. Thus, using two other scenarios, the numbers indicated in the column "Number of types of attribute values" may slightly change.

Table A.1: Dataset variables

Attribute	**Values**	**Type**	**Number of types of attribute values**
Type	AB_CDS, BASKET_TRANCHE_CDS, BOND, SN_CDS, SN_CLN, SWAP, SWOP, etc	Categorical	25
ProductType	SWTX, TN, TRPNB, TRS, TSS, VANIL, VLFVB, VLFVL, VMNRNA, VMRNA, XCCYREPO, etc	Categorical	133
Book	1476, 147Y, 1MAA, 20AL, 20QA, 215A, 215B, 21PF, 21PG, etc	Categorical	2005
Desk	147, 190, 1MA, 203, 208, 209, 20L, 20K, 22A, 21T, 21U, 23W, 23Y, etc	Categorical	1004
CCy	AED, ARS, AUD, BHD, BRL, CAD, CHF, CLF, CLP, CNH, etc	Categorical	50
Commodity	Z_KNFP9J49, ZURREGFLOATZ49A, ZTIBO, ZIGGOBVTLC24AU30EUR, etc	Categorical	26568
PorS	P, S	Categorical	2

Appendix B

Generated tree

Following is shown the excerpt of the implemented tree in a JSON file.

```
{
  "children": [
    {
      "children": [
        {
          "diff": 0,
          "identifier": 4,
          "leaf": 1,
          "name": "EQUAL",
          "not_in": 0,
          "purity": 1,
          "quantity": 1545846,
          "quantity_diff": 0,
          "quantity_equal": 1545846}],
      "diff": 0,
      "identifier": 2,
      "leaf": 0,
      "name": "EQUAL VALUES",
      "not_in": 0,
      "purity": 1.0,
      "quantity": 1545846,
      "quantity_diff": 0,
      "quantity_equal": 1545846
    },
    (...)
    ]
```

References

[ABE06] Nahla Ben Amor, Salem Benferhat, and Zied Elouedi. Qualitative classification with possibilistic decision trees. In *Modern Information Processing*, pages 159–169. Elsevier, 2006. Cited on page 13.

[AI20] Deep AI. *High-Dimensional Data*, (accessed Feb 1, 2020). https://deepai.org/machine-learning-glossary-and-terms/high-dimensional-data. Cited on page 8.

[ALSA$^+$17] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data. *The Journal of Machine Learning Research*, 18(1):8753–8830, 2017. Cited on page 16.

[AW10] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. Cited on page 9.

[BKK96] Stefan Berchtold, Daniel A Keim, and Hans-Peter Kriegel. The x-tree: An index structure for high-dimensional data. In *Very Large Data-Bases*, pages 28–39, 1996. Cited on page 8.

[BMCY11] Bernadette Bouchon-Meunier, Giulianella Coletti, and Ronald R Yager. *Modern information processing: from theory to applications*. Elsevier, 2011. Cited on pages 11 and 13.

[Bre20a] Matt Brems. *A One-Stop Shop for Principal Component Analysis*, (accessed Jan 24, 2020). https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c. Cited on page 10.

[Bre20b] Frederico Breno. Play tennis, version 1, 2018 (accessed Jan 20, 2020). https://www.kaggle.com/fredericobreno/play-tennis. Cited on pages xiii and 12.

[Bri20] Brilliant.org. *Classification Problems*, (accessed Jun 29, 2020). https://brilliant.org/wiki/classification/. Cited on page 41.

[Che20a] James Chen. *Trading Book*, 2019 (accessed Jun 26, 2020). https://www.investopedia.com/terms/t/tradingbook.asp. Cited on page 26.

[Che20b] James Chen. *Commodity Index*, 2020 (accessed Jun 25, 2020). https://www.investopedia.com/terms/c/commodityindices.asp. Cited on page 26.

[Cor14] Fair Isaac Corporation. Building powerful, predictive scorecards. 2014. Cited on page 39.

[Dem07] Yaşar Demirel. Fundamentals of equilibrium thermodynamics. *Nonequilibrium Thermodynamics: Transport and Rate Processes in Physical, Chemical and Biological Systems, 2nd, Amsterdam: Elsevier*, pages 1–52, 2007. Cited on page 14.

[dJG14] Johny Emanuel de Jesus Gueirez. Árvores de decisão desequilibradas para deteção de erros em transações de comércio externo usando técnicas de data mining. 2014. Cited on pages 11, 12, 13, 15, and 32.

[DKZ13] Gintautas Dzemyda, Olga Kurasova, and J Zilinskas. Multidimensional data visualization. *Methods and applications series: Springer optimization and its applications*, 75:122, 2013. Cited on page 10.

[DZ10] Michael Doumpos and Constantin Zopounidis. A multicriteria decision support system for bank rating. *Decision support systems*, 50(1):55–63, 2010. Cited on page 18.

[Far20] Farlex. *Present value*, (accessed Jun 22, 2020). `https://financial-dictionary.thefreedictionary.com/Present+Worth`. Cited on page 21.

[FPSS96] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996. Cited on pages xi, 6, and 7.

[GA10] Megha Gupta and Naveen Aggarwal. Classification techniques analysis. In *National Conference on Computational Instrumentation*, pages 128–131, 2010. Cited on page 6.

[Gan20] Akhilesh Ganti. *Trading Desk*, 2019 (accessed Jun 25, 2020). `https://www.risk.net/definition/trading-book`. Cited on page 26.

[Hay20] Adam Hayes. *Trade*, (accessed Jun 17, 2020). `https://www.investopedia.com/terms/t/trade.asp`. Cited on page 1.

[Hel20] Software Testing Help. *Data Mining Vs Machine Learning Vs Artificial Intelligence Vs Deep Learning*, (accessed Jun 20, 2020). `https://www.softwaretestinghelp.com/data-mining-vs-machine-learning-vs-ai/`. Cited on page 6.

[In20] Built In. *Artificial Intelligence*, 2019 (accessed Jun 20, 2020). `https://builtin.com/artificial-intelligence`. Cited on page 6.

[Inc20] CFI Education Inc. *Profit and Loss Statement (PL)*, (accessed Feb 2, 2020). `https://corporatefinanceinstitute.com/resources/knowledge/accounting/profit-and-loss-statement-pl/`. Cited on page 5.

[Inv20] Capital Com SV Investments. *PL attribution*, (accessed Feb 1, 2020). `https://capital.com/p-l-attribution-definition`. Cited on page 6.

[Leo17] Leslie C Leonard. Web-based behavioral modeling for continuous user authentication (cua). In *Advances in Computers*, volume 105, pages 1–44. Elsevier, 2017. Cited on page 12.

[Lis20] ListenData. *Weight of Evidence (WOE) and Information Value (IV) Explained*, 2019 (accessed Jul 1, 2020). `https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html?m=1#What-is-Information-Value-IV-`. Cited on page 40.

[LN14] Feifei Li and Suman Nath. Scalable data summarization on big data, 2014. Cited on page 9.

[LS17] NL Larus-Stone. Learning certifiably optimal rule lists: A case for discrete optimization in the 21st century. 2017. *Undergraduate thesis, Harvard College*, 2017. Cited on pages 16 and 17.

[LSAA$^+$18] Nicholas Larus-Stone, Elaine Angelino, Daniel Alabi, Margo Seltzer, Vassilios Kaxiras, Aditya Saligrama, and Cynthia Rudin. Systems optimizations for learning certifiably optimal rule lists. In *SysML Conference*, 2018. Cited on page 16.

[NEM09] Robert Nisbet, John Elder, and Gary Miner. *Handbook of statistical analysis and data mining applications*. Academic Press, 2009. Cited on pages 10, 11, 12, and 13.

[NFB$^+$19] Søren Nors Nielsen, Brian D Fath, Simone Bastianoni, Joao C Marques, Felix Muller, Bernard D Patten, Robert E Ulanowicz, Sven Erik Jørgensen, and Enzo Tiezzi. *A new ecology: systems perspective*. Elsevier, 2019. Cited on page 14.

[Nil14] Nils J Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 2014. Cited on page 6.

[Pat20] Dushyant Pathak. *Difference between Recursion and Iteration*, 2018 (accessed Jun 28, 2020). `https://www.geeksforgeeks.org/difference-between-recursion-and-iteration/`. Cited on page 27.

[Pen20] PennState. *High-Dimensional Data Analysis*, (accessed Feb 1, 2020). `https://www.methodology.psu.edu/ra/var-sel/`. Cited on page 8.

[Poo20] Poonam. *Is Non-Regression Testing Exist?*, 2017 (accessed Jun 19, 2020). `https://www.testorigen.com/is-non-regression-testing-exist/`. Cited on page 5.

[Ris20a] Infopro Digital Risk. *PL attribution test*, (accessed Feb 3, 2020). `https://www.risk.net/definition/pl-attribution-test`. Cited on page 6.

[Ris20b] Infopro Digital Risk. *Trading book*, (accessed Jun 26, 2020). `https://www.risk.net/definition/trading-book`. Cited on page 26.

[Ros20] Sean Ross. *Balance Sheet vs. Profit and Loss Statement: What's the Difference?*, (accessed Feb 2, 2020). `https://www.investopedia.com/ask/answers/121514/what-difference-between-pl-statement-and-balance-sheet.asp`. Cited on page 5.

[San20] Pavan Sanagapati. *Weight of Evidence(WOE) Information Value(IV)*, (accessed Jul 1, 2020). `https://www.kaggle.com/pavansanagapati/weight-of-evidence-woe-information-value-iv`. Cited on page 39.

[Sho20] Business Plan Shop. *Profit & Loss (PL)*, (accessed Feb 5, 2020). `https://www.thebusinessplanshop.com/en/help/glossary/profit-and-loss`. Cited on page 5.

[Sim19] Steven Simske. *Meta-Analytics: Consensus Approaches and System Patterns for Data Analysis*. Morgan Kaufmann, 2019. Cited on page 14.

[Smi02] Lindsay I Smith. A tutorial on principal components analysis. Technical report, 2002. Cited on pages 9 and 10.

[SP16] Henrik Starefors and Rasmus Persson. Mlid: A multilabelextension of the id3 algorithm, 2016. Cited on pages 10, 11, 12, 13, 14, and 15.

[To20] Statistics How To. *Dimensionality High Dimensional Data: Definition, Examples, Curse of*, (accessed feb 2, 2020). https://www.statisticshowto.datasciencecentral.com/dimensionality/. Cited on page 8.

[TTM15] Domenico Talia, Paolo Trunfio, and Fabrizio Marozzo. *Data analysis in the cloud: models, techniques and applications*. Elsevier, 2015. Cited on pages 10 and 13.

[Vas20] Nataliia Vasylyna. *Non-Regression testing, what is it?*, 2011 (accessed Jun 19, 2020). https://blog.qatestlab.com/2011/03/20/non-regression-testing-what-is-it/. Cited on page 5.

[VK20] Aditya Saligrama Vassilios Kaxiras. *CORELS*, (accessed May 23, 2020). https://corels.eecs.harvard.edu/corels/. Cited on page 17.

[Wik20a] Wikipedia. *Decision tree*, (accessed April 10, 2020). https://en.wikipedia.org/wiki/Decision_tree. Cited on page 11.

[Wik20b] Wikipedia. *Artificial intelligence*, (accessed Jun 20, 2020). https://en.wikipedia.org/wiki/Artificial_intelligence#Applications. Cited on page 6.

[YN10] Liu Yuxun and Xie Niuniu. Improved id3 algorithm. In *2010 3rd International Conference on Computer Science and Information Technology*, volume 8, pages 465–468. IEEE, 2010. Cited on page 14.

[ZD98] Constantin Zopounidis and Michael Doumpos. Developing a multicriteria decision support system for financial classification problems: The finclas system. *Optimization Methods and Software*, 8(3-4):277–304, 1998. Cited on page 18.

[ZD01] Constantin Zopounidis and Michael Doumpos. A preference disaggregation decision support system for financial classification problems. *European Journal of Operational Research*, 130(2):402–413, 2001. Cited on page 18.

[ZHT06] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006. Cited on page 10.