Using Machine Learning to classify HS codes for Fashion Products

Ivânia Maria Afonso Barbosa

Master's Dissertation

Supervisor from FEUP: Prof. Mário Amorim Lopes

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Master in Engineering and Industrial Management

2021-06-28

Resumo

A HUUB é uma *start-up* tecnológica que conecta as interações entre os vários intervenientes da cadeia de abastecimento da indústria da moda, através de uma plataforma integrada de logística. Acompanhando o crescimento de consumo online, e com o Covid-19 a atuar como um acelador, as marcas de moda apresentam cada vez mais exigências relativamente ao serviço que a HUUB oferece no segmento *Business to Consumer* (B2C). O serviço *Delivered Duty Paid* (DDP), que atualmente não é disponibilizado pela empresa, tem registado uma procura cada vez mais frequente.

Um dos fatores críticos para a disponibilização do serviço de DDP é a verificação da conformidade dos *HS codes* dos produtos. Um *HS code* é um código composto por seis dígitos que codifica informação relativa às características de um determinado produto, sendo utilizado pela alfândega para determinar o valor de impostos e taxas alfandegárias a cobrar por cada encomenda. Neste momento, o processo de classificação dos produtos com um *HS code* é realizado de forma externa à HUUB. O objetivo do projeto passa por construir um modelo de *Machine Learning (ML)* que utilize a informação armazenada na base de dados da HUUB para gerar um *HS code*, permitindo que a empresa detenha o controlo completo sobre a operação de classificação.

Começou-se por enquadrar o problema, focando na compreensão da estrutura hierárquica dos *HS codes*. Seguiu-se um pré-processamento dos dados. O passo seguinte centrou-se no desenvolvimento de um modelo de classificação com a capacidade de captar a taxonomia inerente ao problema dos *HS codes*. Desta forma, foi construído um modelo hierárquico utilizando uma estrutura de Árvore de Classificação Local por Nível (LCL). Esta estrutura envolveu três classificadores independentes, cada um responsável por um nível hierárquico distinto (Capítulo, Cabeçalho e Subcabeçalho).

Para a fase de treino dos modelos de *ML* foram desenhadas duas abordagens distintas. A primeira abordagem (*"Real Data"*) implicou a utilização de dados de saída reais do classificador anterior como dados de entrada para treinar o classificador seguinte. A possibilidade de evitar a propagação de erros serviu como motivação para o desenvolvimento desta abordagem. A segunda abordagem (*"Predicted Data"*) seguiu um método tradicional, através da utilização dos dados previstos do classificador antecedente para o treino do classificador subsequente. Para a análise das diferentes abordagens foram gerados vários cenários, considerando diferentes algoritmos de *ML*, nomeadamente *Naive Bayes*, *Multinomial Logistics Regression*, *Decision Tree* e *Random Forest*.

Ambas as abordagens de treino foram testadas com observações desconhecidas, assegurando que os modelos foram avaliados em condições o mais próximas possível do ambiente de produção. Os resultados demonstraram que o melhor modelo da abordagem *"Real Data"* retornou os melhores resultados, justificados pela diminuição da propagação de erros na fase de treino. Esta conclusão validou a abordagem *"Real Data"*.

A implementação deste modelo em ambiente de produção irá constituir o primeiro passo para a HUUB na integração do serviço de *Delivered Duty Paid* (DDP).

Abstract

HUUB is a technological start-up with a digital logistics integrated platform that operates in the fashion industry, managing the interactions between several players of the supply chain, including fulfillment partners, carriers, and brands. As the online consumption increases each year, with the Covid-19 pandemic working as an accelerator factor, fashion brands are becoming progressively more exigent with the service level HUUB's is offering them in the Business to Consumer (B2C) segment. Delivered Duty Paid (DDP) is becoming a frequently demanded service that HUUB does not currently provide.

One of the critical factors to start offering the DDP service is to verify the compliance of the products' HS Codes. An HS Code is a six-digit code that codifies information about a product's attributes and is used by the customs to determine how much duties and taxes (DAT) each package will be levied. The classification service for the products' HS Code is currently being outsourced. Hence, this project intended to build a Machine Learning (ML) model that leveraged the products' information stored in HUUB's database to generate the six-digit HS code, giving HUUB full ownership over the process.

The initial step was to frame the problem, which included thoroughly comprehending the HS Code hierarchical structure. This was followed by a data preprocessing step. The subsequent phase was to develop a classification model that could capture the taxonomy of the HS Code problem. To that extent, a hierarchical classification model was built, using a Tree Local Classifier per Level (LCL) structure. This structure involved three independent classifiers, one for each hierarchical level (Chapter, Heading, and Subheading).

For the model training, two different training approaches were designed. The first approach ("Real data" approach) entailed using real output data from the previous classifier as input data to train the subsequent one. The prospect of avoiding error propagation was the propulsor behind its development. The second approach ("Predicted Data" approach) followed a traditional method, applying the predicted data from the previous classifier as the input data for the subsequent one. Different scenarios were designed, considering the variations in the training approaches and the dataset distributions, along with the experiment of multiple ML models. This project focused on using the Naive Bayes, the Multinomial Logistic Regression, the Decision Tree, and the Random Forest algorithms.

Both training approaches were tested with unseen observations, ensuring the models were evaluated in an environment as close to the production settings as possible. The results showed that the best model of the "Real Data" approach returned a better performance, resultant from the diminishing in the error propagation between classifiers, validating the "Real Data" training approach.

The deployment of this model will constitute the first step in the Delivered Duty Paid (DDP) service's implementation.

Acknowledgements

I would first like to thank my thesis supervisor, Professor Mário Amorim Lopes, whose expertise was invaluable in the success of this project.

To Jorge Ferreira, my project mentor inside the company, I wish to show my deepest gratitude. His guidance, support, and constant encouragement provided me with the tools that I needed to always steer in the right direction. His insightful feedback pushed me towards a sharpened thinking and brought my work to a higher level. No words can explain the importance of his guidance and for that, I will forever be grateful. My appreciation extends to Vasco Faria, for always helping me whenever I felt stuck and inspire me to be a more curious person. To both of them, a big thank you for sharing this experience with me and highlighting my mornings during these months.

To Mariana Romba, the first friendly face I encountered at HUUB and that I kept coming back to for advice and refreshing conversations. To the Product Team, for making me feel welcomed since the first day, and consistently assuring I had every resource I needed to complete my dissertation. They are an inspiring example of what a team should be.

To HUUB for being a company that allowed me to grow and challenge myself each day. Although the work was fully remote, I always felt connected to HUUB, which was a key factor for the success of this dissertation. A kind word to all the Huubsters who always were so close as just one call away. A special thank you to all the women at HUUB, for being a role model for young girls initiating their journey, proving to us how far we can go one day.

To all the FEUP Faculty of the Integrated Master's Degree in Industrial Engineering and Management. The success and quality of this course are undoubtedly linked to the professors who are passionate about teaching and that prepare their students for the greater challenges entailed in this new chapter.

To Gonçalo, for being the best friend I could ever ask for. Thank you for always encouraging me to push forward and being my biggest supporter. Your advice, ideas, and expertise were invaluable in helping me turning this dissertation into something I am proud of.

To all my friends who gave me five unforgettable years full of companionship and laughter, but especially to Sofia and Beatriz who made Porto feel like a new home. I will always cherish the memories I made at FEUP as the best years so far.

To my family, for shaping the person I am today and for being the moral compass that has always guided me through life's challenges.

To Zé Pedro, who I have the privilege to call brother, for always helping me find the silver lining in every situation.

Finally, to my mother Edna and my father Henrique for their unwavering support and belief in me. I cannot explain how grateful I am for all the opportunities and experiences you provided me with. This dissertation could not have happened if not for your unconditional love and support. If I can allow myself to dream big it is because I have always had you cheering me on the backstage.

To all, the most sincere thank you.

"If I have seen further it is by standing on the shoulders of Giants."

Isaac Newton

Contents

Acronyms and Symbols

B2B	Business to Bussiness
B2C	Business to Consumer
DAP	Delivred at Place
DAT	Duties and Taxes
DDP	Delivred Duty Paid
DL	Deep Learning
EU	European Union
HC	Hierarchical Classification
HS	Harmonized System
LCL	Local Classifier per Level
LCN	Local Classifier per Node
LCPN	Local Classifier per Parent Node
ML	Machine Learning
NLP	Natural Language Preprocessing
WCO	World Customs Organization

List of Figures

List of Tables

Chapter 1

Introduction

Online consumption is registering increasingly higher growth rates each year. The current COVID-19 pandemic has accelerated the transition into the online hemisphere and the fashion industry has not been immune from the effects of the upheavals. As this trend of channel transferring between traditional retail and online services is expected to persist and consumers' expectations are rising, fashion retailers should invest heavily in enhancing their online shopping experience and services. Fulfillment companies and partners are increasingly being requested Delivered Duty Paid (DDP) as a service to improve the online customer delivery experience.

This dissertation, conducted in the Product Team of a technological start-up specialized in managing the interactions in the supply chain, focuses mainly on developing a Machine Learning classification algorithm to assign fashion products with a six-digit HS code. This code is a fundamental component in integrating a DDP service, which constitutes the ultimate goal of the project.

1.1 Motivation

In 2019, retail e-commerce sales worldwide amounted to 3.53 trillion US dollars, and e-retail revenues are projected to grow to 6.54 trillion US dollars in 2022 (Sabanoglu, 2021).

The current COVID-19 pandemic came as an accelerating factor in shifting the consumption culture into the online hemisphere (Kim, 2020) caused by the avoidance of shopping in physical stores, whether due to government restrictions or health concerns (Sheth, 2020).

Despite online shopping offering greater flexibility in the purchasing experience, in terms of time, location, and product variety (Rohm and Swaminathan, 2004), factors such as the immediate desire for possession, the fear of leakage of personal information, and the learning curve that online shopping demands are pointed out as the main obstacles to a mass online shopping conversion. The latter reason is based on the comparison between the upfront investment needed and the benefits of online consumption (Watanabe and Omori, 2020). However, the pressure and lack of options brought on by the pandemic made it almost mandatory for consumers to cross that learning curve. The result has been an increase in the number of consumers who have already invested in the unpleasant task of learning how to shop online and are now prompt to continue shopping (Watanabe and Omori, 2020).

A natural question that follows is whether these newly forced consumer patterns will persist once time eases. Although it has been globally admitted that the shift in consumption patterns due to Covid-19 will be irreversible, some studies point out that consumption volume may return to pre-pandemic levels (Watanabe and Omori, 2020), and there is no overall consensus on the matter.

The fashion industry has not been able to remain unscathed by the changes caused by the Covid-19 pandemic. Major fashion retailers such as Neiman Marcus and JC Penney declared bankruptcy three months after the breakout (Chen, 2020), and sales of sports brands, namely Adidas and Nike, dropped 25% from February to March 2020 (Chen, 2020). On the other hand, brands with a robust e-commerce structure have overcome these difficult times with paramountcy, registering a significant increase in online sales. Puma, for example, enhanced its online logistics before Covid-19 started and is forecasting a 40% increase in e-commerce growth. H&M registered a decline in physical store sales but was able to turn a profit since its online sales rose 30% between March and May 2020 (Reuters, 2020). It is also clear that many companies have understood the lasting impact the pandemic has had on consumer shopping patterns, and industry giants such as Inditex are investing $1 billion to upgrade their IT and online logistics (Kansara, 2020).

As this trend of channel transferring between traditional retail and online services is expected to persist, fashion retailers should invest heavily in improving their online shopping experience and services (Youn et al., 2021), under the penalty of the consumers resuming old habits or losing their market share to competitors that provide a higher quality experience.

One factor that ensures a competitive edge in the aggressive e-commerce environment is a service-focused strategy. A company that provides superior and remarkable services will lead its customers to favorable behavioral intentions, guarantee loyalty, high retention rates, and proliferation of WOM (Rita et al., 2019). Retailers should be conscious that customer expectations of online convenience have increased as a natural response to the service innovations (Duarte et al., 2018). Several studies were conducted to distinguish which e-commerce dimensions affect overall customers' e-service perception of quality. Rita et al. (2019) and Duarte et al. (2018) reached similar conclusions: website design, security/privacy, and delivery are the three most essential dimensions that lead customers to engage with online shopping. The latter factor was identified as the most prominent one, highlighting the importance of companies ensuring the product is delivered in good condition and within the promised time (Rita et al., 2019).

The current context created a window of opportunity for companies to enhance the delivery experience they are currently providing their e-commerce customers. This improvement possibility served as this dissertation's primary motivation and constituted the project's baseline developed henceforth.

1.2 Company Outline

HUUB, a technological start-up founded in 2015, is a digital logistics integrated platform that operates in the fashion industry, managing the interactions between the several intervenients of the supply chain. From the suppliers, fulfillment partners, and carriers to the fashion brands and their final customers, connecting them all on a single platform - SPOKE.

It can be challenging for a fashion brand to manage its entire supply chain when several players are not in contact with each other, having distinct objectives. The result is a complex and non-collaborative framework that lacks visibility. HUUB positions itself in the center of a dynamic ecosystem where every member is connected through SPOKE. Brands are granted a holistic view of their supply chain. HUUB manages their physical and information flows, resorting to data analysis to enhance the brands' value proposition by offering actionable insights. The platform (SPOKE) was designed for internal management, as well as for brands, suppliers, service providers, and partners. It provides brands with an easy interface, allowing them to track and analyze their orders, do inventory management and sync their e-commerce platforms (e.g., Shopify, WooCommerce, or the brand's website) with the rest of their supply chain. The brands' products can be commercialized through both wholesale and e-commerce sales channels. Each of these channels impacts differently the operations required.

There are essentially two different types of brands that partner with HUUB: Small and Medium Enterprise (SMEs) brands and Enterprise brands. A brand with more than 200 orders a day is classified in the last category and usually expects more personalized services, having a higher probability of requesting larger scale functionalities such as DDP or multi-warehousing.

Besides the platform, the start-up offers warehousing and distribution services. Warehousing services include inbound, outbound (pick, pack and ship), and value-added services (such as inventory or item labeling). As for the fulfillment partners, HUUB is currently working to enhance and enrich the value proposition. The objective is to shift the pure commercial relationship to a partnership, where the two players benefit mutually. HUUB offers partnering warehouses access to a new market: small brands already integrated into a standardized operation. This market segment is usually not explored by these partners since small-scale brands are not appellative for their business model. HUUB operates as an aggregator for the warehouses, enabling them to have one point of contact with the capability of managing multiple individual brands.

HUUB has two active partnering warehouses, one in Portugal (Agility) and a new one located in Poland (No Limit). This new hub's location was strategically thought out to be closer to the central European market. In addition, one of the company's most significant brands has its primary markets centered around this new geography. The proximity to these consumption markets brings many benefits concerning logistics, namely fast transit times and cheaper last-mile deliveries.

HUUB also relies on a competitive network of shipping partners to deliver the orders to the final customers.

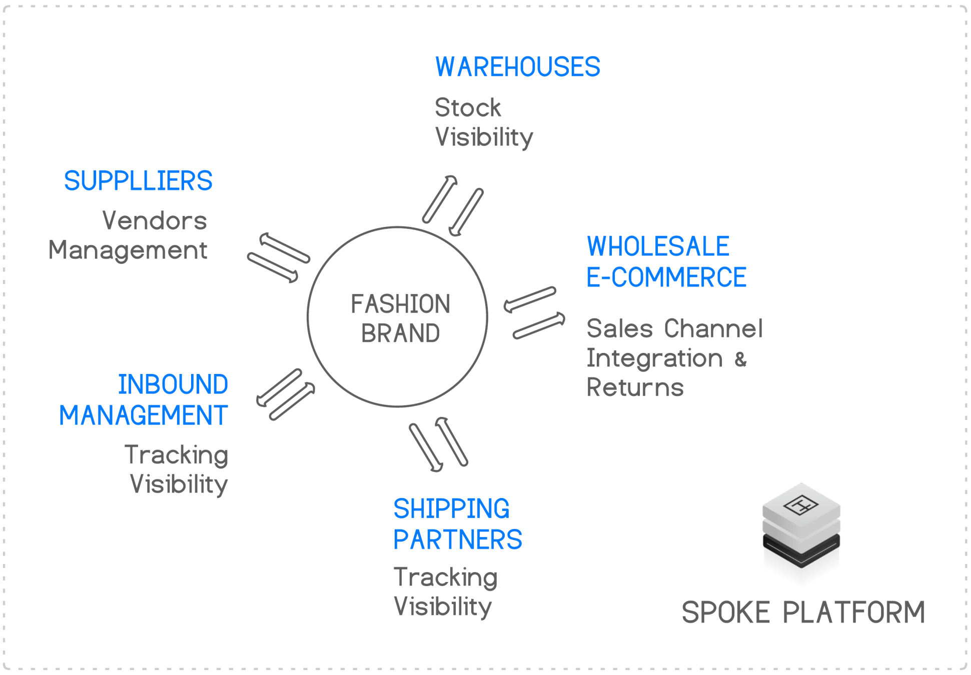


Figure 1.1: HUUB positioning in the Supply Chain

1.2.1 SPOKE Top Level Architecture

To better understand how the information flows and how the players are interconnected, it is necessary to analyze the top-level architecture of the company's information systems, depicted in Figure 1.2.

A brand can either integrate with Shopify, Woo Commerce, or any other e-commerce pre-built platform or develop its own website. Since pre-built online stores usually share the same processes, the universal layer is a service within SPOKE's architecture that helps to do the translation between the online store API (e.g., Shopify API) and the Brand API. On the other hand, if the brand uses its own website or wants to integrate its ERP, a middleware can be built to perform that translation. The ownership of the middleware (i.e., the integration effort) varies, so both HUUB or the brand can be the ones developing the integration.

The third-party logistics partners (3PL) connect to SPOKE through a 3PL API. For this integration, there are also two options available. If the fulfillment partner requires an adaptation to its information system, HUUB has to build a connector between the Warehouse Management System (WMS) and the 3PL API. On the other hand, if the 3PL partner accepts to adjust to HUUB's information system, no adaptations are required and the fulfillment center can connect directly to the 3PL API. SPOKE gathers the information received from multiple sources, empowering a comprehensive management.

The platform aggregates multiple microservices since this solution is highly maintainable and allows for independent and scalable processes (Figure 1.3). KAFKA is a distributed streaming platform used to establish communication between the different microservices. Examples of such services are the Order Management System (OMS), Warehouse Management System (WMS), Inventory Management System (IMS), Catalog, Adress Normalization, Delivery, Carrier Account, Document Generator, and Tracking.

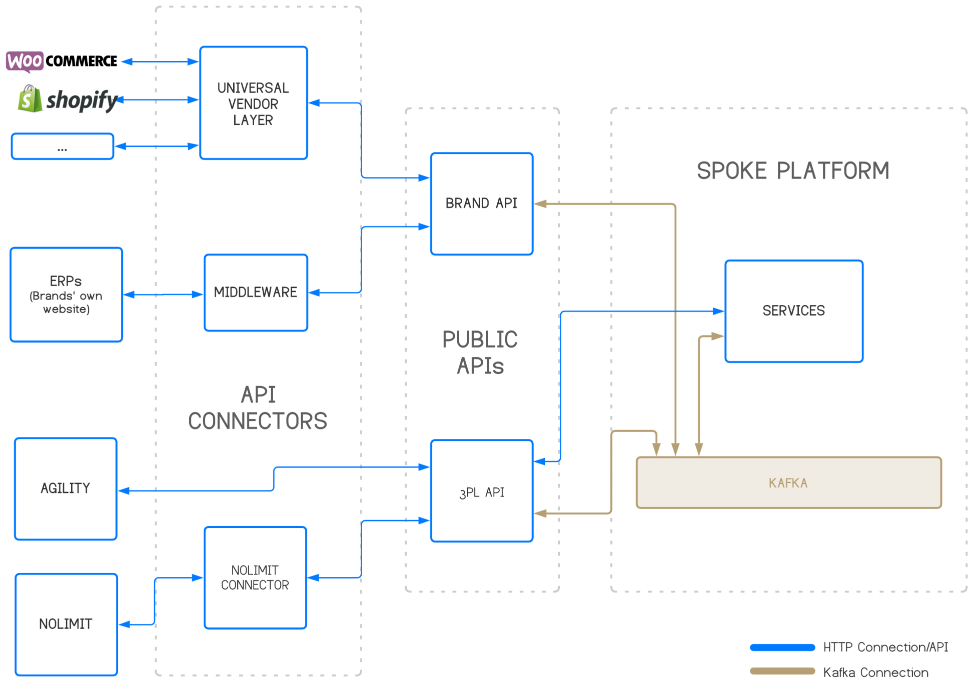


Figure 1.2: Top Level Architecture of SPOKE platform

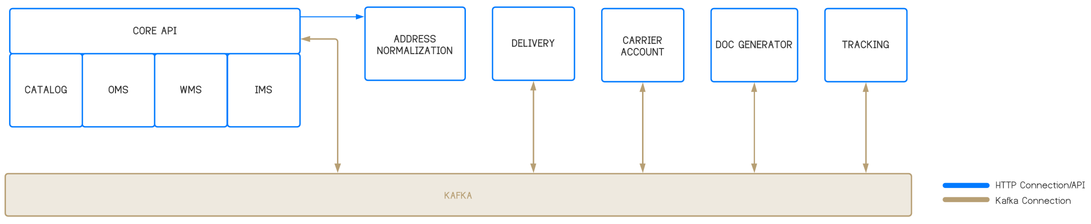


Figure 1.3: SPOKE microservices

1.2.2 Operational Processes

The physical flow of the products is divided into three main processes: reception, picking & packing, and shipping. Once the products arrive at the partner's warehouse, the reception process of the goods is carried out. HUUB is responsible for managing its customers' stock and verifying that the quantities delivered comply with the service levels agreed between them and their suppliers (SLAs). The picking and packing process consists of collecting and packing the products ready for shipment. The shipping process is the last level in the product's physical flow within the warehouse and encloses activities such as confirming quantities, weighing, creating labels, and transport documentation. Once this step is concluded, the items are ready to be shipped. There can also be a reverse logistics flow in the case of a return. Value-Added Services (VAS) can additionally be performed in case of wrong labeling, wrong addresses, or an inventory check, among others.

Once the orders leave the warehouse, a wide range of carriers is available to do the last-mile delivery to the final customer. HUUB's shipping partners network represents a great advantage for the brands as it allows them to negotiate prices that otherwise they would not reach, given the small scale of their businesses. HUUB chooses the carrier based on a decision algorithm that considers each of the orders' weight (through a volume forecast model) and the brand's required service level for the delivery. This algorithm then proceeds to estimate the carrier whose price for that combination is the lowest. The brand is agnostic to this decision process, and for each weight range and service level, it pays a given fixed fee. The difference between the fixed price billed to the brand and the actual value that the carrier charged translates into HUUB's profit margin, which is inherently variable. Suppose there is a negative discrepancy between the amount forecasted and the amount charged by the shipping company, either due to a variation in rates or an incorrect estimation of the order's volume or weight. In that case, HUUB bears the loss of profit, enhancing the importance of the volume forecast model.

1.2.3 Pricing Model

It is also important to mention the pricing model billed to brands, which composes HUUB's revenue streams. There are two primary streams of revenue. The first is a unitary standard price per item transacted, which includes a handling cost for inbound, outbound, and returns, along with a fee for holding stock. This fee is usually defined at the start of each season or at the onboarding stage of each fashion brand. Extra costs can also be added to the initial price if there is a special request (e.g., solicitation for augmentation of warehouse capacity for peak sales periods) or occasional exceptions caused by wrong labeling, wrong addresses, or discrepancies in the inventory. The second flow of revenue regards the customized transportation price, to be set with each brand at the beginning of the season. The difference between the fee charged to the brands and the carrier's actual cost constitutes the profit margin for HUUB.

$$\begin{aligned}Brand\ Price = {} & Shipping\ cost\ (outbound + returns) + Handling\ costs\ (inbound + outbound \\ & + returns) + Stock\ holding\ costs + Cards\ (special\ orders) \\ & + VAS\ (occasional\ exceptions)\end{aligned} \tag{1.1}$$

1.2.4 Size and growth of HUUB

HUUB has now returned from a very successful round of investments and continues to have high growth prospects for the future. HUUB has its operation currently scattered over more than 88 countries and relies on 15 active partner brands that trust its services and strive to grow with it. Whether they are individual clients or physical stores, most end customers are located in Europe, being aligned with HUUB's strategy. However, end customers are scattered worldwide, which requires HUUB to be very competitive and efficient in managing and transporting goods.

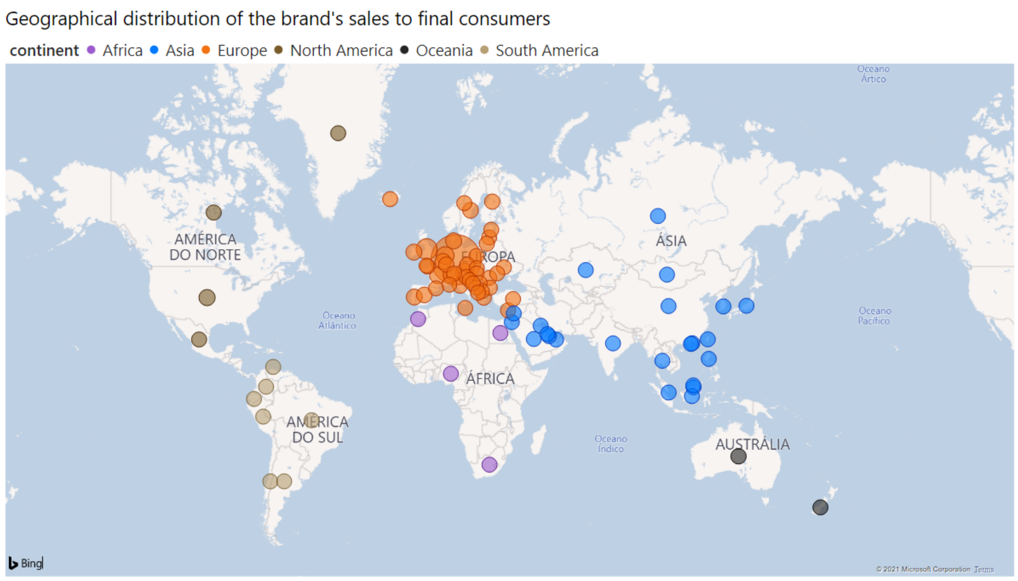

Geographical distribution of the brand's sales to final consumers
continent ● Africa ● Asia ● Europe ● North America ● Oceania ● South America
AMÉRICA DO NORTE
EUROPA
ÁSIA
ÁFRICA
AMÉRICA DO SUL
AUSTRÁLIA
Oceano Atlântico
Oceano Pacífico
Oceano Índico
Oceano Ártico
Bing
© 2021 Microsoft Corporation Terms

Figure 1.4: Geographical distribution of the brands' sales to final consumers

Compared to 2019, 2020 exhibited a growth rate of over 54% regarding the volume of orders sold. This expansion tendency is persisting in the current year, as when comparing the first quarter of 2021 with the homologous period of 2020, a growth rate of 128% was registered.

The ultimate goal of HUUB is to democratize the supply chain, optimize the process, and give the same opportunities to small companies and enterprise fashion brands, allowing them to focus on their core business: creating fashion collections.

1.3 Problem addressed

With the Covid-19 pandemic outbreak across the globe, HUUB recorded an increase in e-commerce order volume. The rising demand for online shopping motivated brands to become increasingly more exigent with the level of service and differentiation HUUB was providing in the e-commerce sales channel.

One feature constantly requested that HUUB is currently unable to respond to is the implementation of the DDP service for the B2C segment. Delivered Duty Paid (DDP) is the method of exporting goods where the seller bears responsibility for all the stages necessary to deliver the products to their final destination, including customs clearance and payment for duties and taxes

(DAT). DDP is identical to the incoterm DAP (Delivered at Place), except that the latter is the end customer's responsibility to pay taxes and duties at customs.

At the moment, HUUB only provides DDP services for wholesale orders. In the B2B domain, the brand can choose whether to send its order by DAP or DDP. If the choice is DDP, it is also the brand's responsibility to estimate how much duty and taxes (DAT) each order will be charged at customs. The estimated value is paid to HUUB, who subsequently ships the order to its final destination. The carrier responsible for the order charges HUUB with the exact amount of DAT collected at customs. Finally, a financial reconciliation between HUUB and the brand takes place to adjust any misestimated value.

For the DDP process, the brand must also provide the HS Code for each of its products. The Harmonized System is a worldwide instrument for goods classification that assigns each product a six-digit code. This number informs how much DAT is charged for each product at the country's customs.

Currently, the generation of the HS Code is either conducted internally by the brand or through an outsourcing company referred by HUUB if the brand is incapable of developing the service in-house. Since the HS Code constitutes a fundamental element in estimating DAT for customs, it is paramount that the merchandise's classification is as accurate as possible.

However, the DDP B2B process cannot be replicated in the B2C segment for the following reasons:

1. the financial reconciliation is a very manual procedure that could not be scaled to match the e-commerce order volume;
2. the brand may not have the capacity to estimate DAT for such a large volume of orders;

For the above reasons, the solution would be for HUUB to have full ownership of the entire DDP process in e-commerce. Hence, a mechanism to provide, or at least verify, the products' HS Codes is fundamental. This verification procedure is critical to assure the accuracy of the duties and taxes estimated, which would fall under HUUB's responsibilities in this new solution. For this reason, HUUB should not rely solely on external information but have an internal mechanism to ensure the quality of the classification of the fashion products. As a result, it would be possible for HUUB to start offering an HS Code Generation service to the brands as a replacement for the current outsourcing solution.

HUUB's inability to offer the DDP service has led to the loss of contracts in the conversion pipeline. This feature was verified to act as a deal-breaker for the contacted brands. Following this reasoning, brands already partnering with the company have also been putting considerable pressure on this solution to be made available.

This problem has yet to be solved since it is a complex and time-consuming process that involves a wide range of different procedures and requires a deep understanding of the matter. Up to now, other issues have proven to be more urgent; however, it is now of critical importance for HUUB to tackle this problem and develop a process to classify fashion products internally. This development would bring HUUB one step closer to implementing the DDP solution

1.4 Project Objectives

The project's objectives were established considering HUUB's value proposition for both the brands and their partners, ensuring the project contributed to its improvement.

The dissertation focus was the development of a model that classifies each product with a six-digit HS code, resorting to Machine Learning techniques. Since the project's implementation involved multidisciplinary domains, the main objective was subdivided into milestones to achieve more measurable results.

The project's first goal was to analyze HUUB through an internal perspective to understand its core processes and determine which requirements would be necessary to implement a DDP service.

The second objective relates to the development of the algorithm itself, resorting to ML. The goal was to create a model that leverages the products' information stored in HUUB databases to generate six-digit HS Codes, which allow the customs to identify which products are included in each shipment.

The last goal of the project, which is out of the scope of this dissertation, consists of integrating the classification algorithm with the information system currently in place at HUUB.

As a more general objective, it is expected that the deployment of the HS Classification system sets the first step in the integration of the DDP process for the B2C segment.

1.5 Methodology

The initial steps of the project followed a top-down approach to have a business perspective of the implications and requirements a DDP service would involve. In addition, a deeper analysis was conducted in the relevant areas to ensure a thorough understanding of the different topics.

Along with this reflection, a survey amongst HUUB's brands was conducted to set the groundwork for the fundamental features regarding the DDP service. As a result of this approach, the internal classification of the HS Codes for the products was identified as one of the critical components for a reliable implementation of the DDP service.

After having stated the problem and understood its background, a careful literature review was conducted to become aware of the state-of-the-art, as well as a deep dive into the technical concepts addressed throughout the dissertation, namely Machine Learning principles, with a particular focus on Hierarchical Classification.

Regarding the classification model development, a simple rule-based algorithm was first constructed to serve as a baseline model for the project. Following that, a detailed hierarchical structure was built, replicating the taxonomy of the six-digit HS Code. This structure was then applied to all the Machine Learning models explored. Finally, several scenarios were tested regarding the ML models to determine the optimal combination of factors that best fit the problem.

In the end, a comparison between all the scenarios was performed, and a verdict was rendered. The comparison was established through the prediction accuracy, as well as other performance indicators, such as the hierarchical precision, recall and f1-score, training and testing time, and model interpretability.

1.6 Thesis Outline

The dissertation is structured in six chapters. The current chapter aims to emphasize the importance and reasoning behind this project. The following chapter is "Literature Review", where the theoretical background and state-of-the-art reviewed are presented.

The third chapter elaborates deeper on the problem description, and it is divided into two main sections. First, the AS-IS situation is presented through a map of processes explaining the current flows of information for the products' HS Codes. The second part describes the project's aims where the TO-BE scenario is presented in terms of the new methodological approaches proposed, defined to bridge the gaps of the AS-IS.

The project's methodology is explained in chapter four. This chapter starts by briefly describing the used dataset and all the preprocessing steps required. Subsequently, the focus is on a thorough explanation of the Hierarchical Classification model assembled for the project. The chapter ends with a summary of the scenarios designed to be tested and analyzed in chapter five.

The fifth chapter depicts the results obtained from applying the proposed models, comparing their performance across the distinct scenarios proposed. The sixth and final chapter, "Conclusions and Future work", provides an overview of the thesis's findings as well as a reflection of its goals. The dissertation closes with the statement of improvement opportunities identified for future work.

Chapter 2

Literature Review

The scientific relevance of this master's thesis is demonstrated in this chapter. This second chapter summarizes the published literature on the HS classification of products and categorization of goods, including the literature written in the field of cross-border e-commerce. It also covers the theoretical background behind the models developed, emphasizing the fundamentals of Machine Learning and hierarchical classification models.

2.1 Customs and Trading globalization

Globalization has paved the way for numerous opportunities towards attaining economic growth and prosperity. However, some barriers still prevent the internationalization of global processes within local governments (Altaheri and Shaalan, 2020). Customs serve as one of the most crucial government agencies in the globalization process, acting as regulators of international trade and commerce, overseeing services and processes. One of the most prominent customs' responsibilities is the insurance of the compliance of goods' classification within each country's regulations (Altaheri and Shaalan, 2020). This matter is of relevance for this project's scope and will be further detailed in this section.

2.1.1 International trading in e-commerce

The international trade of physical goods requires the movement of cargo across borders. For this transaction to happen, and since all export and import cargo is subject to customs control, shippers must obtain approval from the relevant national government authorities (Bergami, 2016). There are substantial barriers when doing commerce through international borders. For small and medium-sized (SME) e-commerce companies, cross-border shipping logistics are highly challenging. These companies require a closed-loop delivery network that fulfills both the delivery terms agreed upon by the customer at checkout and the ability to provide a return service according to the conditions established. As stated in previous chapters, the shipping experience and delivery options are key factors in online consumers' purchasing decisions. Sellers are concerned about logistics and distribution networks as well as cost minimization (Gessner and Snodgrass, 2015).

2.1.2 Incoterms

A key cross-border concern for both customers and companies is the specification of the entity responsible for paying the import duties and taxes at customs (Gessner and Snodgrass, 2015). The Incoterms are eleven terms, first published in 1936 and last updated in 2020 by the International Chamber of Commerce (ICC). They dictate the distribution of obligations and risks between parties in the international trade of goods (Durdağ and Gül, Delipinar, 2021) and facilitate global trade, establishing a common language, globally accepted and standardized.

The choice of incoterm in international contracts severely impacts physical and financial risks for importers and exporters (Bergami, 2016), being extremely important to have a comprehensive knowledge of each terms' implications. There are several incoterms, but the EXW, DAP, and DDP should be underlined as the most relevant ones for this dissertation.

EXW or Ex-works is the incoterm that better shields the seller since he is not responsible for anything once the goods are made available to the buyer. The products are delivered at the seller's premises, and from that point on, the buyer gains ownership of the merchandise and handles all further costs and risks (Davis and Vogt, 2021).

In the Delivered at Place (DAP), the seller carries the goods to the final place of destination, but he is not responsible for the unloading process. His responsibilities extend from packing and export clearance to carriage expenses and terminal costs up to the agreed destination port. Consequently, the buyer is responsible for all costs, duties, and taxes associated with the goods' unloading. The risk is transferred from the seller to the buyer at the final designated place (Davis and Vogt, 2021).

DDP is identical to DAP, except for the responsibility for clearing import customs.

2.1.3 Delivered Duty Paid (DDP)

Delivered Duty Paid (DDP) is the method of exporting goods where the seller bears responsibility for all the stages necessary to deliver the products to their final destination, including customs clearance and payment for duties and taxes. These responsibilities also include payment of administrative fees related to the passing of the goods through customs (Bergami, 2016). This incoterm can be applied for B2B and B2C shipments, but this analysis's focus will be mainly on the online sales channel.

Looking through the online sales perspective, DDP is beneficial for both the end customers and the retailers, presenting itself as a solution to enhance consumers' shopping convenience in the online hemisphere. Because it provides a smooth shipping solution, it allows consumers to save time and energy, avoiding hassles and problems dealing with customs issues. Instead, customers pay the estimated tax and duties upfront, usually during checkout, and prevent future complications. On the other hand, the sellers can increase the value of their market offer for international consumers (Duarte et al., 2018) by offering DDP. This enhancement of the retailers' value proposition is aligned with the strategy mentioned in Section 1.1., which determines favorable consumer behavior in the e-retail market. Another advantage of DDP is the possibility of partners keeping up with retailers that use drop-shipping as a fulfillment method. In this type of order fulfillment, global retailers do not hold stock but pass the responsibility for delivering the goods to the final customers entirely to their supplying partners (Yu et al., 2017). Some retailers that resort to this method demand that the orders are shipped via DDP, making it impossible for partners that do not use this incoterm to remain competitive in this segment.

2.2 Harmonized Commodity Description and Coding System

Facilitating and standardizing global trade is a top priority for customs. The Harmonized Commodity Description and Coding System is a worldwide instrument for product classification, developed by the World Customs Organization (WCO) and used by more than 200 countries (World Customs Organization, 2013). It is seen as the "true language of international trade" and is extensively used by governments, international organizations, and the private sector for several purposes. These include trade policy, rules of origin, monitoring of controlled goods, internal taxes, freight tariffs, international trade statistics and economic research and analysis, with the collection of import duties and taxes being the primary use.

This code is a fundamental piece in the DDP service since it is directly related to the value of duty and taxes each product will be charged at customs. For this reason, the Harmonized Commodity Description and Coding System will be explored in deeper detail in the following sections.

2.2.1 Structure of the Harmonized System

The Harmonized System (HS) is a structured nomenclature with multiple levels that divide goods into chapters (first two digits), headings (third and fourth digits), and subheadings (fifth and sixth digits). The logical structure comprises over 1200 headings within 96 chapters, the latter being organized in 21 sections. Exception chapters include chapter 77, which is reserved for future uses, and chapters 98 and 99, restricted to national use. There are over 5000 precise definitions at the subheadings level, identified by the complete six-digit code (Luppes et al., 2019). Countries also have the freedom to extend their digits further, according to their international needs. Examples of these extensions are the Brazilian NCM (Nomenclatura Comum do Mercosul) (eight-digit code) and the United States HTS (Harmonized Tariff Schedule) (ten-digit code).

The HS Code is used across all industries and classifies an immense variety of products. Each level comprises a textual description and a numerical code. Products can be classified by the material condition or by the function or usage (a combination of the two is also possible) (Weerth,

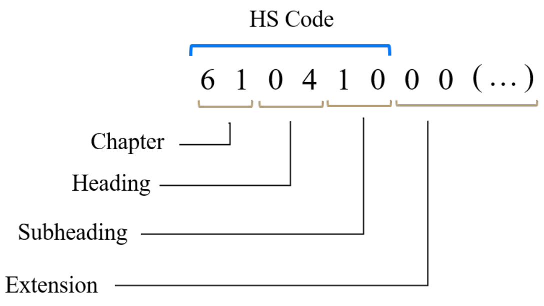


Figure 2.1: HS Code Structure

2008). As the hierarchical structure of the code gets deeper, the products' definitions become more detailed. An example of a description alongside a valid HS code is shown in Table 2.1.

Table 2.1: Example of a correct HS-classification at the six-digit level

Description	**HS Code**
Knitted Cotton T-shirt	610910

HS Classification is the process of finding the Harmonized System (HS) description that best fits each product's characteristics (Ding 2015). The HS also comprises the "Six General Rules for the Classification of Goods" and the "Section and Chapter Notes", both important for clarifying and interpreting the classification of goods (Weerth, 2008).

2.2.2 Example of an HS Code Product Classification

A classification of a knitted women's T-shirt made of cotton was chosen to illustrate how the hierarchical classification system for the HS Code is constructed. This article of clothing belongs to Section XI ("Textiles and Textile Articles") of the Harmonized System, under the chapter "Articles of apparel and clothing accessories, knitted or crocheted". This Chapter has code 61. One level lower is the heading "T-shirts, singlets and other vests, knitted or crocheted". This description corresponds to HS Code 6109. The system goes deeper into finding the subheading group, in this case, related to the material of the product, cotton. This subheading prefaces the six-digit code 610910. This example is illustrated in Table 2.2.

Table 2.2: Hierarchical structure of the Harmonized System

Level	**Example Code**
Chapter (HS-2)	61
Heading (HS-4)	6109
Subheading (HS-6)	610910
Country specific extension (HS-8 and beyond)	61091000

2.2.3 Misclassification of the HS Codes

The misclassification of commodities is often pointed out as the biggest reason for tax and duties non-compliance. This misclassification happens when discrepancies are verified between the supplier's code and the one admitted as correct by the importing country. For instance, the Office of the Auditor General of Canada reported, in 2010, 17% to 30% of misclassification of imported goods. Although these results could not be extrapolated to estimate the revenue the agency would have collected given the correct classification, 2700 compliance verifications (corresponding to 3% of the total value of imports for the fiscal year of 2009-2010) were conducted. This verification translated to C$59 million in additional duties and taxes due to non-compliant imports (Office of the Auditor General of Canada, 2010).

Kappler (2011) traced the primary causes of the discrepancies of the commodities' HS codes to the following factors:

i **HS complexity**
The classification process can be subjective and unclear, usually requiring additional knowledge from HS experts.

ii **Gaps in terminology**
HS commodity descriptions are often highly technical and very domain-specific. There is a gap between how a product is expressed by trade and how it is described in the Harmonized System, making it difficult for a non-expert to correctly find a product classification based on its description.

iii **Reliance on third-party service providers**
Considering that most firms, especially small and medium enterprises, do not have the capital power to develop in-house harmonized classification systems, this function is commonly attributed to third-parties agents or service providers. This outsourcing solution may be cost-effective, but it can prove risky since these entities do not hold the legal or financial liabilities associated with customs compliance. The unbinding aspect of these commercial relationships tempts outsourcing service providers to handle the HS classification as default entry data rather than one of domain-specific management.

iv **Improper tools**
Another relevant factor in the chronic misclassification of goods is the extensive use of keyword search tools that are not able to capture complex commodities descriptions. The search is done disregarding any hierarchy or semantic context, returning only exact or partial term matches that are mostly unrelated and often inaccurate HS Classification options.

Misclassification has repercussions both on governments and private entities. For shippers, non-compliance of goods can lead to a higher or lower import duty rate, as well as unnecessary shipment delays and fines, and other administrative penalties (Ding et al., 2015). For government entities, misclassification can translate into a loss of revenue and inaccurate information for statistical purposes.

In order to prevent discrepancies in the duties and taxes being charged, customs authorities rely on several approaches to monitor and reinforce commodity classification. In an article written for World Customs Journal, Kappler (2011) enumerates several of the monitoring approaches, namely, risk management, post-entry audits, pre-shipments inspections performed by third-party companies and the application of incentives and penalties for non-compliance.

2.3 Automatization of the HS Code Classification

Several studies regarding the classification of export and import goods using HS Codes have been carried out, resorting to several approaches, including rule-based, Machine Learning (ML), and, more recently, Deep Learning (DL), incentivizing the automatization of the process (Heijde, 2019).

Deep learning is making significant progress regarding the application of different DL techniques in HS Code classification systems, resorting to NLP methods. This progress can be demonstrated in the work of Luppes et al. (2019), Li and Li (2019), Ryzhova and Sochenkov (2019), Che et al. (2018), and Ding et al. (2015). However, despite this tremendous success and potential, DL still has many constraints in industrial and corporate applications. Vajjala et al. (2020) identifies the following motives which illustrate why deep learning is not always the optimal solution: the possible overfitting on small datasets, the difficulty to adapt to other domains, the expensive run times, and the lack of interpretability and controllability of the algorithms.

2.3.1 HS Code Classification using Machine Learning

Regarding the use of ML models, most of the articles reviewed used as input the textual descriptions of the goods and therefore resorted to Natural Language Processing (NLP) techniques. This description is usually positioned next to the HS code in the import form filled by the importer companies, and it is frequently an open, concise, and unstructured sentence. The biggest challenge of an input with said characteristics is correctly extracting keywords, information, or features in the text to obtain an accurate classification model while simultaneously dealing with feature sparseness.

Altaheri and Shaalan (2020) presents an interesting analysis of the performance of various ML models in predicting the HS code based on the customers' input commodity descriptions. The experiments were based on the Dubai Customs dataset and the prediction models were performed in two experiment settings. The first setting evaluated the capability of the model to predict the entire HS Code (six digits). In contrast, the second setting tested the ability to predict the header of the HS Code (first four digits), corresponding to the commodity type (e.g., T-shirts, blouses, trousers).

The preprocessing steps performed to remove any factors which may degrade the performance of the ML model can be summarized as follows: (1) remove duplications, (2) remove punctuation and remove stop words, (3) remove non-English words, (4) remove numbers, (5) lemmatization (i.e., reduce the word to its origin) and (6) lower case. In addition, to bridge the issue of the unbalanced data, a downsampling approach was followed. Finally, a tokenization procedure was applied using the Term Frequency - Invert Document Frequency (TF-IDF) technique. TF-IDF is a simple information retrieval approach for feature extraction that can determine which terms are most distinguishing for that document.

The accuracy, the recall, the F1-score, and the precision were the evaluation metrics tracked. Altaheri and Shaalan (2020) tested multiple Machine Learning models, namely Naive Bayes, K-Nearest Neighbor, Decision Tree, Random Forest, Linear Support Vector Machine, and Adaboost. All of the models were evaluated using the 10-cross fold validation technique. The Linear SVM achieved the highest accuracy of 76,3% and 84,58% for the first and second settings, respectively. It was also observed that the classifier model's performance improved significantly by just considering the code's heading. This was justified by the fact that the four-digit classification reduces the pressure of the classifier, substantially decreasing the number of labels. The model's accuracy was verified to be intimately connected to the quality and veracity of the textual descriptions.

Similar work was done by Harsani et al. (2020), with an experiment that shared the aim of the previous study: the implementation of ML models in the classification of imported goods based on short text descriptions. In this context, the classification was done based on the eight-digit HS system, using an Indonesian trade dataset with respect to footwear and its accessories, focusing on Chapter 64 of the HS Classification System. Three text mining methods were tested out specifically, LibShort Text, Text Categorization, and Topic Modeling. For each of these methods, two ML algorithms were employed: Support Vector Machine (SVM) and K-Nearest Neighbours (KNN).

2.4 Machine Learning Principles

Machine Learning algorithms are currently being incorporated into various programs and systems to support organizations in making better decisions, especially in predictive analysis and pattern recognition.

A Machine Learning process usually consists of a series of steps that form a loop that is iterated until a desirable output is achieved. The process englobes understanding the domain and prior knowledge of the problem, performing data selection, cleaning and pre-processing, learning the models, interpreting the results, and finally consolidating and deploying the discovered knowledge.

Machine Learning algorithms derive their strength from their ability to learn from the available data. There are three main ML models: Supervised Learning/ Predictive Models, Unsupervised Learning/ Descriptive Models and Reinforcement Learning models.

1. Supervised Learning/ Predictive Models

Most Machine Learning algorithms use supervised learning, being the most mature and well-studied type of learning. As explained by Morocho-Cayamcela et al. (2019), supervised learning "comprises looking at several examples of a random vector x and its label value of vector y, then learning to predict y from a completely new x, by estimating p(y|x), or particular properties of that distribution". To rephrase it, in supervised learning, each training example must be fed along with its respective label. The aim is to assign each predicted observation with one of the existent labels (Jung, 2020).

Supervised learning falls mainly into two categories: classification and regression. A classification problem refers to the task of classifying an observation into a discrete set of classes or categories, while regression is the ability to predict the values of a continuous variable for a given example (Vieira et al., 2019).

2. Unsupervised Learning/ Descriptive Models

According to Morocho-Cayamcela et al. (2019), "unsupervised learning implicates observing different instances of a random vector x and aiming to learn the probability distribution p(x), or its properties". Thus, unsupervised learning is an algorithm that learns patterns and trends of similarity from unlabeled data.

3. Reinforcement Learning

It is an example of Machine Learning where the machine is trained to make specific decisions based on the external environment maximize a specific predefined goal. The machine trains itself continually based on the environment it is exposed to and applies its enriched knowledge to specific problems (Vieira et al., 2019).

2.4.1 Classification

In the context of this dissertation, the classification task will be further explored.

Classification is a supervised learning algorithm where a training set with labeled data is used to make predictions. The model which learns from the training data to identify the category or class of the input features is called a classifier. A classifier can be characterized as binary if there are only two possible outcomes (e.g., True or False) or multi-class when more than two categories are available. The classification can additionally be broken down into single-labeled and multi-labeled. The distinction between the two is that in multi-label problems, each sample is assigned to a set of labels or targets, whereas in single-label problems, each observation can only be classified into one category.

There are five main steps in a Supervised Machine Learning process, which can be used to structure any classification problem.

1. Data Collection: the gathering of past data forms the foundation of future learning. The machine's learning prospects improve as the variety, density, and volume of relevant data increases.

2. Data Preparation: The quality of the data used in any analytical procedure is critical. It is necessary to conduct pre-processing steps to fix the dataset issues, such as missing data and outliers. Exploratory analysis is one approach to delving into the intricacies of data in greater depth, hence burgeoning the intellectual value of the data.

4. Model Selection: Model selection is the process of selecting one final Machine Learning model from among a collection of candidates. It is a procedure that can be used to compare models of different types (e.g., Logistic Regression, Decision Trees, KNN) and models of the same type with varying hyperparameters.

It can prove challenging to find the best algorithm for a problem when there is a large number of options, each with a distinct level of complexity. The ultimate goal is to choose a model with a low generalization error, which is the expected value of the misclassification rate when measured on unseen data. This error, however, is computed on an independent test set, not used during model training; therefore, it cannot be used to select the optimal model for the problem. This is why Model Selection is so relevant during the training phase (Murphy, 2012).

There are two main approaches for the model selection phase. The first one is best applied in a data-rich environment where the dataset is divided into three parts: a training set, a validation set, and a test set (Figure 2.2). "The training set is used to fit the models; the validation set is used to estimate prediction error for model selection; the test set is used for assessment of the generalization error of the final chosen model," as Hastie et al. (2017) explained.

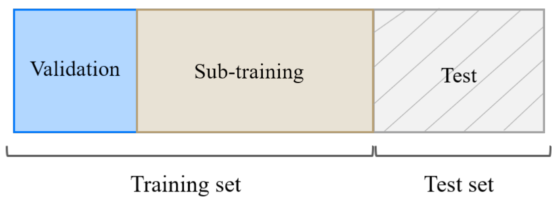


Figure 2.2: Example of a sub-train, validation and test set split (adopted from Hastie et al. (2017))

This approach reveals impractical in many applications given the restricted amount of data available for training and testing. A simple but widely used alternative is cross-validation (Hastie et al., 2017). This technique is usually preferred as it allows the model to train in different training sets, giving a more accurate indication of how it will perform.

The first step in cross-validation is to shuffle the dataset and split it into k folds. For each unique fold, one group is considered the "validation" set, and the remainder k-1 folds are the "sub-training" sets used to fit the model. Then, the average error is computed over all the folds and used as a proxy for the test error (Murphy, 2012). A representation of a 10-cross-validation method is presented in Figure 2.3 for illustrative purposes.

It should be mentioned that model selection does not entail fitting the final model, as all the models will be discarded at the end of this stage. Instead, after the most suitable model is chosen, a new final model will be fitted on all available training data and subsequently evaluated on the preserved testing set that remained unaffected during the model selection phase (Bishop, 2006).

5. Model Training: This step entails determining the best combination of weights and bias to a Machine Learning algorithm to minimize a loss function over the prediction range. Model training aims to build the best mathematical representation of the relationship between data features

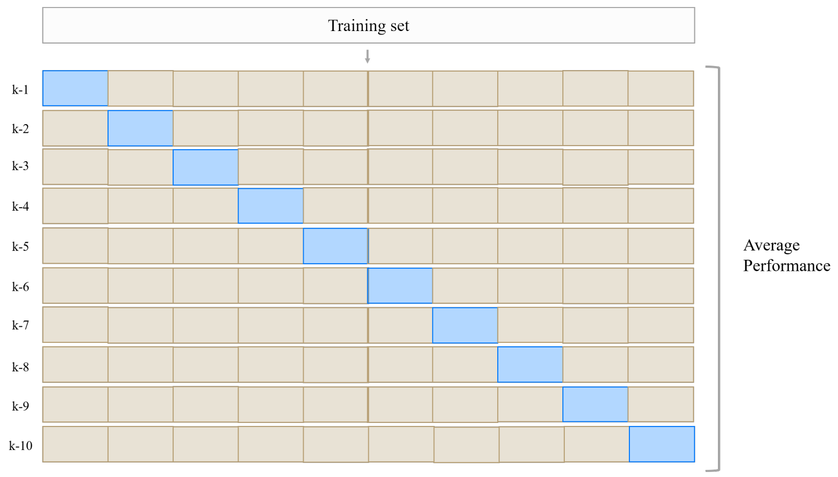


Figure 2.3: 10-cross fold-validation example employed in the training set (adapted from Bishop (2006))

and a target label. The quality of training data and the training algorithm are both critical assets during the model training phase.

6. Model Evaluation: When an algorithm is trained and evaluated on the same set (training data), it is always expected to produce a high accuracy result. For this reason, it is essential to use new data when evaluating the model to prevent the likelihood of overfitting to the training set. This evaluation can be performed using either the testing set or a cross-validation approach.

The three main metrics used to evaluate a classification model are accuracy, precision, and recall.

Accuracy is defined as the percentage of correctly predicted observations for the test data.

$$accuracy = \frac{correct\ predictions}{all\ predictions} \tag{2.1}$$

Precision corresponds to the fraction of relevant examples (true positives) among all the examples classified as relevant (true and false positives). Precision describes the degree of certainty with which a model classifies an event as positive.

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{2.2}$$

Recall is defined as the fraction of relevant examples predicted to belong to a class with respect to all of the examples that truly belong in said class.

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{2.3}$$

The objective is to attain the highest precision and recall possible. **F1-score** is a single metric that can be used to evaluate models.

$$f1\ score = \frac{2 * precision * recall}{precision + recall} \tag{2.4}$$

2.4.2 Naive Bayes Algorithm

Naive Bayes is a very simplistic algorithm for classification, based on the Bayes Theorem (Bayes' rule is represented in 2.5) and in a strong assumption that all variables are conditionally independent given the class (Webb, 2016).

$$P(y|x) = \frac{y * P(x|y)}{P(x)} \tag{2.5}$$

For a given object x, Naive Bayes provides a method for estimating the posterior probability $P(y|x)$ of each class y, resorting to information from sample data. The class y with the highest probability is selected to classify the object x.

Regarding the model's computing efficiency, training time is linear for both the number of training examples and the number of attributes; classification time is linear to the number of attributes but unaffected by the number of training instances. These characteristics make Naive Bayes particularly useful for comparatively large data sets, as they are extremely fast in the training phase compared to other classifiers. Additionally, it requires a small amount of training data to estimate the necessary parameters (Webb, 2016).

2.4.3 K-Nearest Neighbor (KNN)

The K-Nearest Neighbor algorithm is used to predict a sample's class based on the class of its neighbor records (Kuhkan, 2016).

This method is also known as a lazy learning algorithm as it focuses on storing all the instances corresponding to the training data in an n-dimensional space rather than building a generic internal model (Jabbar et al., 2013). Because of this, every time a prediction is required for an unseen record sample, the algorithm has to search through the entire training set for the k-nearest instances to return the most similar class as the prediction, resulting in a very high computational cost. On the other hand, the algorithm's benefits include a simple implementation procedure and a high level of robustness to noisy training data.

2.4.4 Logistic Regression

Logistic regression is a Machine Learning classification algorithm that returns a binary outcome using one or more independent variables. Equation 2.6 illustrates its representation. To predict a binary output value (y), input values (x) are combined linearly using weights or coefficient values (Jung, 2020).

$$y = \frac{e_0^b + b_1 x}{1 + e_0^b + b_1 x} \tag{2.6}$$

Because it can mathematically explain how the predictions were made, it has a very high interpretability potential in illustrating how a set of independent variables affect the outcome of the dependent variable. However, its fundamental drawback is that it only works when the predicted variable is binary, when the data is free of missing values and under the assumption that the predictors are independent of one another (Jung, 2020).

2.4.5 Multinomial Logistic Regression

Multinomial Logistic Regression is an extension of the Logistic Regression that includes built-in support for multi-class classification problems. Given a set of independent variables, the model predicts the probabilities of several possible outcomes of a categorically distributed dependent variable. Logistic Regression is intended for binary classification problems, where the categorical dependent variable is limited to two classes (Wang, 2005). The Multinomial Logistic Regression approach imposes altering the loss function used to train the model to a cross-entropy loss function and shifting the output of the single predicted probability distribution to a multinomial probability distribution (one for each class label) (Krishnapuram et al., 2005).

The multinomial probability distribution (equation 2.7) is the probability distribution that defines multi-class probabilities. Under a Multinomial Logistic Regression model, the probability that x belongs to class i is written as:

$$P(y^i = 1|x,w) = \frac{\exp(w^{i^T} x)}{\sum_{j=1}^{m} \exp(w^{j^T} x)} \tag{2.7}$$

for $i \in (1,....,m)$, where w^i is the weight vector corresponding to class i and the superscript ^{T} denotes vector/matrix transpose.

2.4.6 Support Vector Machine

The Support Vector Machine is a classification algorithm that depicts training data as points in space separated into categories by a margin as wide as possible (Figure 2.4). New points are then added to space by predicting which category they will fall into (Vieira et al., 2019).

The fundamental benefit of SVM is its ability to work with sparse data and generate models with a relatively small amount of training data. The method's only drawback relates to the incapacity to provide probability estimations immediately.

2.4.7 Decision Tree

Decision Trees are widely employed in the data science community, given their capacity to solve complex problems by providing a clear visualization of the problem that allows for a straightforward interpretation and analysis (Amor et al., 2006).

The decision tree algorithm is constructed in the form of a tree structure composed of nodes and leaf nodes (Figure 2.5). It employs if-then rules which are equally exhaustive and mutually

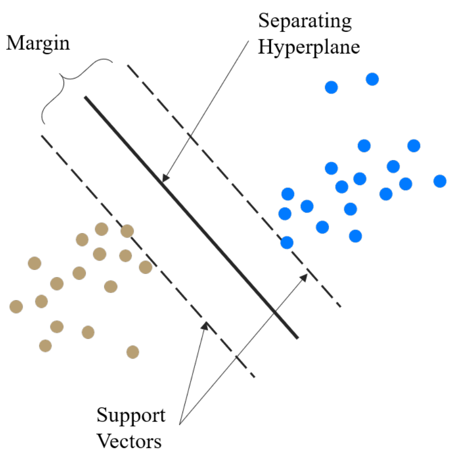


Figure 2.4: Support Vector Machine graphical representation, adapted from Meyer (2020)

exclusive for classification. These rules classify the data by posing a series of questions about the features associated with the items and are learned sequentially using the training data one at a time.

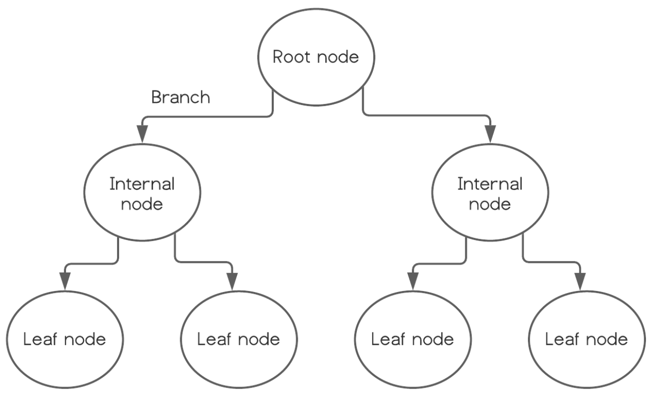


Figure 2.5: Decision Tree Representation

Decision trees are generated with a top-down approach from a root node, from where the data is partitioned into subsets that contain instances with similar values (Yang, 2019). As it grows, the tree can either halt at a leaf node or continue to an internal node. An internal node represents a question, with branches leading to as many child nodes as there are possible answers (Kingsford and Salzberg, 2008)., whereas a leaf node represents a classification or a decision containing the class labels (Song and Lu, 2015). The branches are the segments of the tree that establish the connection between nodes.

2.4.8 Random Forest

Random Forest consists of a large number of individual decision trees working together to form an ensemble. Each tree in the random forest returns a class prediction and the class with the majority of votes determines the model's prediction. Random forest is based on the fundamental concept that a large number of relatively uncorrelated models (trees) acting as a committee will outperform any of the individual constituent models. Because of its high accuracy and ability to handle multiple features with small samples, Random Forests have become a widely used tool. Bagging and Random Selection of Features are the two fundamental concepts behind this technique.

2.5 Hierarchical classification

In the field of data science, there is an inherent dissonance: while the human mind perceives the world around it in hierarchical structures, standard Machine Learning models struggle to represent those relationships, perceiving the data as flat inputs. However, many crucial real-world classification issues, i.e., text categorization, protein function prediction, and music genre classification, are naturally cast as hierarchical classification problems. If these are treated with a flat classification approach, the natural class hierarchy of the data, which might contain highly valuable information, is ignored in the process. Therefore, it is preferable to have a hierarchical approach to deal with taxonomic data.

Before any further analysis on the topic, it is essential to clarify the terminology used in this review, considering that different authors propose different terms for the same concepts, leading to inconsistency across different works.

The main difference between a binary classifier and a multi-class classifier is that the binary classifier can only handle two-class problems, while a multi-class classifier can handle any number of classes. Most approaches to multi-class hierarchical classification could be classified as multi-label (a classifier can assign more than one prediction to a given example). For example, in the hierarchical classification of animals, if a classifier's output has the class "Golden Retriever", it is natural to assume that it also belongs to classes "Mammal" and "Dog", resulting in the classifier's output having three classes. However, because this definition is trivial and applies to all hierarchical approaches, a hierarchical problem will only be considered multi-label if it is possible to assign more than one class to a given example at any hierarchy level (Silla and Freitas, 2009).

Silla and Freitas (2009) proposed a unifying framework for the categorization of the hierarchical algorithm described as a 4-tuple $< \Delta, \Xi, \Omega, \Theta >$, where:

- Δ indicates whether or not the algorithm can predict labels in just one or multiple (more than one) different paths in the hierarchy. SPP (Single Path Prediction) indicates that the algorithm can assign each data instance at most one path of predicted labels. MPP (Multiple Path Prediction) indicates that the algorithm can potentially assign multiple paths of predicted labels to each data instance.

- Ξ is the prediction depth of the algorithm. It can have two values: the hierarchical classification method can be designed to always classify a leaf node (Mandatory Leaf-Node Prediction). Alternatively, the approach can contemplate stopping the classification at any node in any hierarchy level (Non-Mandatory leaf-node prediction).

- Ω relates to the type of hierarchical structure chosen. Usually, the taxonomy is organized into a tree or a DAG (Directed Acyclic Graph) structure (Figure 2.6), but most of the current literature focuses on working with trees as it is an easier problem. For the underlying reason, it will also be the focus of this review.

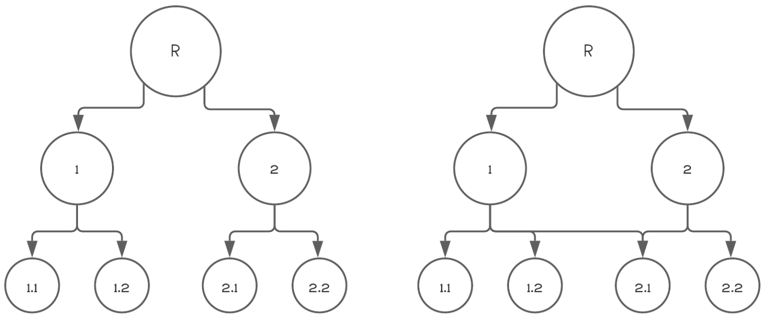


Figure 2.6: An example of a tree structure (left) and a DAG structure (right), adapted from Silla and Freitas (2009)

- Θ relates to the approach in which the hierarchical structure is explored. There are three main methods introduced in the literature: local classifiers, big-bang (or global) classifiers, and flat classifiers.

2.5.1 Flat classification

Flat classification is the most straightforward approach for taxonomic classification, where the class relationships are ignored, and typically only the leaf nodes are predicted. This method operates like a standard classification algorithm during training and testing. However, it gives an indirect solution to hierarchical classification because when a leaf class is assigned to an example, all of its ancestor classes are implicitly assigned to that instance (Silla and Freitas, 2009). Unfortunately, this approach completely disregards the parent-child class relationships, which will most likely reduce the algorithm's performance. When solving a hierarchical classification problem, results reported in several studies were analyzed in Silla and Freitas (2009), and, in the majority of the cases, any hierarchical classification approach (local or global) registered better results than the flat classification approach.

2.5.2 Local Classifiers

The following method is called Local Classifiers, where the predefined data taxonomy is used to create a set of local classifiers by using a circumscribed information perspective (Silla and Freitas,

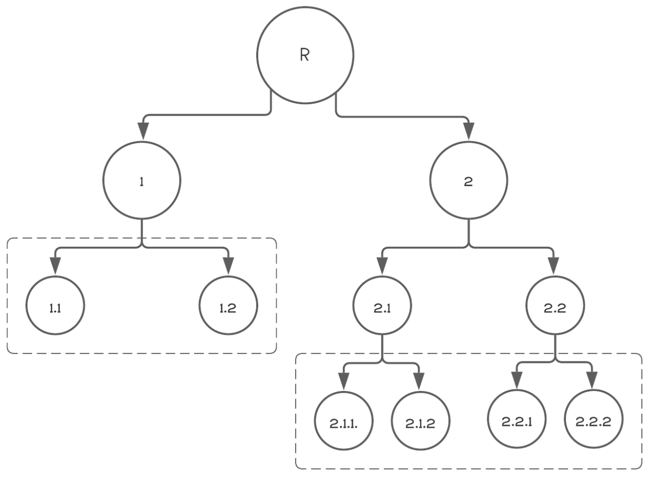


Figure 2.7: Flat classification approach using a flat multi-class classification algorithm to always predict the leaf nodes, adapted from Silla and Freitas (2009)

2009) while retaining simplicity and generality.

This approach also allows multiple datasets, each with different features and different algorithms for each local classifier (Secker et al., 2010). This topic will be further detailed and supported in the section regarding the Local Classifiers per Parent Node.

The local classifiers can be categorized based on how they use local data and build their classifiers around it. More precisely, there are three standard methods for utilizing the local information (Local Classifier per Node approach - LCN, Local Classifier per Parent Node approach - LCPN, and Local Classifier per Level approach - LCL).

Although the three types of local hierarchical classification algorithms covered in the following three subsections differ significantly in their training phases, they all usually use a top-down approach in their testing phases. This top-down technique is usually employed to avoid class member inconsistency.

First, the system predicts the first-level (most generic) class for each new example in the test set. It then utilizes that predicted class to reduce the possibilities of classes predicted at the second level (the only valid candidate second-level classes are the descendants of the anticipated first-level class), and so on, recursively, until the most specific prediction is made.

However, a disadvantage of the top-down class-prediction strategy is that a mistake at a specific class level will be propagated downstream of the hierarchy (error propagation). A blocking technique could be a mechanism for avoiding the proliferation of misclassifications, where an example is transmitted down to the next lower level only if the confidence in the current level's prediction is greater than a threshold. However, this strategy has the cost of presenting the user with less detailed (less valuable) class predictions and should always be evaluated in the specific domain of the problem.

2.5.2.1 Local Classifier per Node approach (LCN)

The local classifier per node method (Figure 2.8 involves training one binary classifier for each node of the class hierarchy (except the root node). The set of positive and negative examples for training binary classifiers can be defined in various ways. In the literature, most works often follow the approach of using as positive training examples all the examples belonging to the current class node and all of its descendant classes. Although this is the most common approach, several other approaches can be used, as Eisner et al. (2005) showed, namely the "Exclusive" policy, the "Less Exclusive" policy, the "Less Inclusive" policy, the "Inclusive" policy, the "Siblings" policy, and the "exclusive siblings" policy.

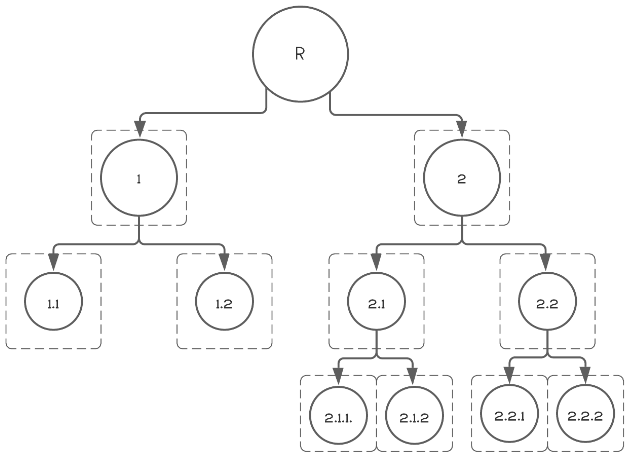


Figure 2.8: Local classifier per node approach (circles represent classes and dashed squares with round corners represent binary classifiers), adapted from Silla and Freitas (2009)

Regardless of how positive and negative instances were specified during the training phase, in the testing stage, the output of each binary classifier will be a prediction indicating whether or not a particular test example belongs to the predicted class of the classifier. However, there is a drawback to this strategy. Using the animals' taxonomy data as an illustrative example, and since the classifiers for each node are trained independently, a possible output can be: class "Dog" = False and class "Golden Retriever" = True. This originates inconsistency in class prediction across different levels during the test phase, which requires the application of inconsistency correction methods such as the top-down approach.

2.5.2.2 Local Classifier per Parent Node approach(LCPN)

The local classifier per parent node (Figure 2.9) strategy is one in which a multiclass classifier is trained per parent node in the class hierarchy to distinguish between its child nodes. It is a very intuitive approach, and since it requires a more controlled number of local classifiers than the previous one, it becomes a leaner and easier method to implement.

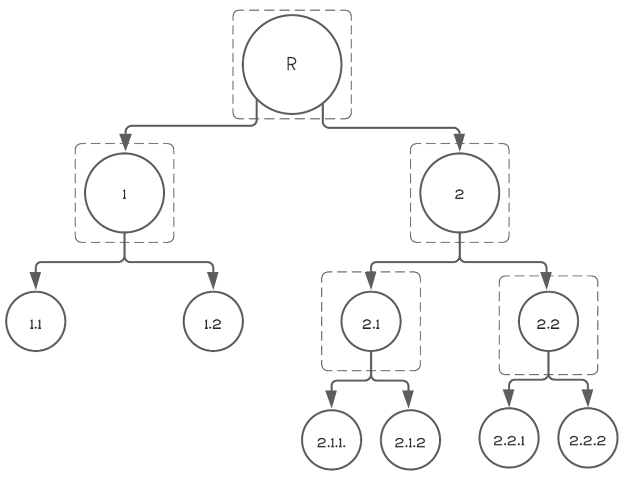


Figure 2.9: Local classifier per parent node (circles represent classes and dashed squares with round corners in parent nodes represent multi-class classifiers that predict their child classes), adapted from Silla and Freitas (2009)

Usually, the same classification algorithm is used throughout the class hierarchy in the local approaches. However, Secker et al. (2007) developed a "selected classifier" approach as an extension of the LCPN technique. The proposed hypothesis stated that utilizing different classification methods at different parent nodes of the class hierarchy would increase the predicted accuracy of the local classifier per parent node approach.

In order to identify which classifier should be used in each node of the class hierarchy, during the training phase, the training set is split into a sub-training and validation set, with examples allocated randomly to each of those datasets. Using the sub-training set, different classifiers are trained and then assessed on the validation set. Finally, the classifier with the highest classification accuracy on the validation set is selected for each parent class node.

Holden and Freitas (2008) introduced an improvement on the selective classifier approach, in which the classifier selection was performed using a swarm intelligence optimization algorithm. The swarm intelligence algorithm does a global search that analyzes the entire tree of classifiers (having a comprehensive view of the training data) at once. In contrast, the original selective classifier approach utilizes a greedy, local search method with a limited local view of the training data when selecting a classifier.

2.5.2.3 Local Classifier per Level approach (LCL)

In the local classifier per level approach, one multiclass classifier is trained for each level of the class hierarchy. Thus, taking, for instance, the example in Figure 2.10, three classifiers, one for each class level, would be trained to predict one or more classes (depending on whether the problem is single-label or multi-label) at its corresponding class level.

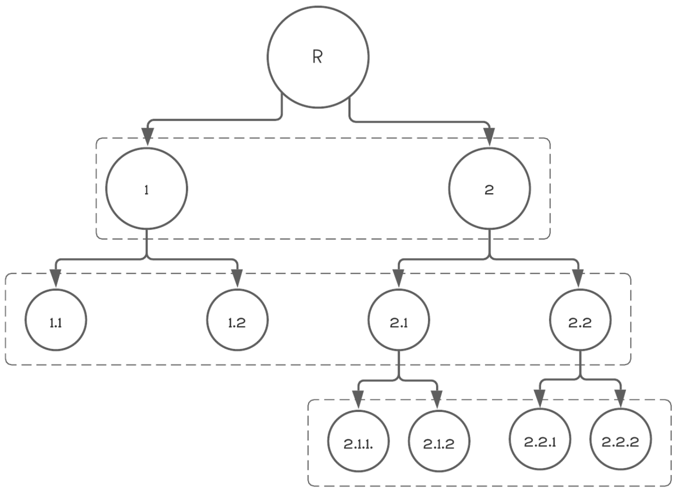


Figure 2.10: Local classifier per level (circles represent classes and each dashed rectangle with round corners encloses the classes predicted by a multi-class classifier), adapted from Silla and Freitas (2009)

The training sets are created in the same way as in the local classifier per parent node technique. The following is one possible (though naive) technique of classifying test samples for classifiers developed using this methodology. Every time a new test sample is provided to the classifier, the output of all classifiers (one per level) is gathered and utilized as the final classification.

The main disadvantage of this class prediction method is that it is prone to inconsistency in class membership. The problem resides in the fact that, by training different classifiers for each taxonomy level, it is possible to obtain class = "Mammal" in the first level, class = "Snake" in the second level, and class = "Golden Retriever" in the final level, resulting in an inconsistent result. Consequently, if this method is utilized, it should be paired with a post-processing procedure that attempts to address the prediction inconsistency, such as the class-prediction top-down approach.

2.5.3 Global Classifiers

The Big-bang or global classifiers correspond to a single, relatively complex model, which considers the entire class hierarchy as a whole during a single run of the classification algorithm (Silla and Freitas, 2009).

The crucial difference between the global and the local classifier lies in the training phase. A global classifier can even use a top-down approach for the testing phase, but the training stage must consider the entire class hierarchy at once. These approaches also lack the modularity for local training of the classifier, which is a key characteristic of the local classifier approach.

Compared to the overall size of all the local models learned by any of the local classifier approaches, learning a single global model for all classes has the advantage that the overall size is often much lower. Furthermore, class membership dependencies across distinct classes (e.g., any example belonging to class "Dog" automatically belongs to class "Mammal") can be taken into

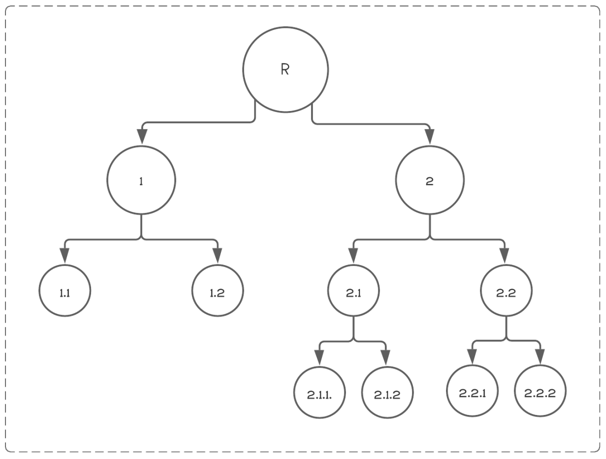


Figure 2.11: Big-Bang classification approach using a classification algorithm that learns a global classification model about the whole class hierarchy, adapted from Silla and Freitas (2009)

account in a natural and even explicit manner (Blockeel et al., 2002). Notwithstanding, the high complexity of this approach is a major drawback.

Global classifiers can go very different directions as it seems to be no specific core characteristic shared by all the different approaches.

2.5.4 Hierarchical Evaluation methods

The subsequent challenge is how to define the evaluation metrics for the hierarchical classification systems. The use of standard performance metrics, designed for flat, one-level classifications, ignores the data's natural taxonomy, which was so carefully preserved during the training stages.

Kiritchenko et al. (2006) explored this problematic, by comparing hierarchical classification approaches with flat classification measures. The difference in the predictive accuracy of the hierarchical metrics over the flat approach in the worst case was 29.39%. This conclusion strengthened the fact that appropriate hierarchical measures should be used in these types of problems.
Kiritchenko et al. (2005) also suggested the use of the metrics of hierarchical precision (hP), hierarchical recall (hR), and hierarchical f1-score (hF). These metrics are modified versions of the standard precision, recall, and f1-score metrics, adjusted for hierarchical categorization, and they can be effectively applied to any hierarchical classification scenario.

$$hP = \frac{\sum_i \mid \hat{P}_i \cap \hat{T}_i \mid}{\sum_i \mid \hat{P}_i \mid} \qquad hR = \frac{\sum_i \mid \hat{P}_i \cap \hat{T}_i \mid}{\sum_i \mid \hat{T}_i \mid} \qquad hF = \frac{2 * hP * hR}{hP + hR} \tag{2.8}$$

- $\hat{P}_i$ is the set consisting of the most specific class(es) predicted for test example i and all its(their) ancestor classes
- $\hat{T}_i$ is the set consisting of the true most specific class(es) of test example i and all its(their) ancestor

classes
- the summations are computed over all the test examples

2.5.5 Hierarchical Classification applications

Hierarchical Text Categorization (HTC) is one of the most explored applications of Hierarchical Classification. The automated classification of electronic documents (emails, articles, web pages) in large-scale taxonomies is becoming increasingly difficult due to the sparsity of the training data. A natural solution is to organize them into hierarchies. For example, in Chakrabarti et al. (1998), the authors present an intriguing example of how hierarchies can improve information retrieval systems. The article gave the example of searching the word "Jaguar", referring to the animal. Without any previous restriction, a vast amount of information about cars was returned. However, limiting the user's search within a hierarchy (for example, searching for "jaguar" in the part of the hierarchy related to animals) would support the disambiguation of polysemous terms. Other authors have proposed different ML models to tackle the HTC problem. Bennett and Nguyen (2009), Gauch et al. (2009), and Qiu et al. (2009) have, respectively, explored an LCN, an LCPN, and a Global Classifier approach to classify large text categorization tasks. Stein et al. (2019) used the Hierarchical Classification (HC) and word embeddings to classify news articles. They also demonstrated that hierarchical performance measures are more suitable for taxonomic problems. Wehrmann et al. (2018) developed a novel approach, using a hybrid method to simultaneously use both local and global classifiers approaches to perform a hierarchical multi-label classification.

Another significant application of HC is protein function prediction in the field of bioinformatics since the classes to be predicted (protein functions) are naturally organized into class hierarchies. Protein function prediction is critical since protein malfunctioning is linked to many diseases, and this type of knowledge can be potentially used to make advancements in medicine. Various authors have explored different protein hierarchies (Clare and King (2003), using an LCL approach; Otero et al. (2010), using a GC approach; and Cerri et al. (2016) tackling a multi-label classification problem).

In organizing and retrieving music information, the genre plays an important role as it is one of the most used concepts to search for music. In this application domain, some of the works that have employed class hierarchies are: Decoro et al. (2007) using an LCN approach; Brecheisen et al. (2006), using an LCPN approach and Ariyaratne and Zhang (2012), which explored an automatic approach to construct a music genre classification tree through subspace cluster analysis.

The aforementioned projects are some of the most common applications of Hierarchical Classification. The HC is not, however, limited to this set of applications. Xiao et al. (2008) created a class hierarchy for the hierarchical classification of emotional speech. A three-level depth hierarchy with an LCPN approach was used to differentiate between six-leaf classes (feeling types): anger, boredom, fear, happiness, sadness, and neutral. In Decoro et al. (2007), the authors employ their Bayesian Network aggregation with K-NN base classifiers in 3D shape classification. The use of hierarchical strategies to solve this problem is motivated by the fact that in 3D shape classification situations, classes are sorted in a hierarchy from most generic to most specialized shapes. Other works that deal with hierarchical image classification are Dimitrovski et al. (2008) and Binder et al. (2010).

To be best of our knowledge, only two papers regarding the hierarchical classification of HS Codes have been published recently. Du et al. (2021) propose an "HS code classification neural network (HScodeNet) by incorporating the hierarchical sequential, and global spatial information of texts, in which a hierarchical sequence learning module is designed to capture the sequential information of commodity description texts". Binh et al. (2021) introduced a deep learning model with a self-attention mechanism alongside hierarchical classifying layers to improve the accuracy of the classification of Vietnamese short text from goods declarations.

2.6 Literature Gap

The majority of the reviewed articles published in the literature resort to NLP and Text Mining techniques to interpret the short text description of the product's attributes and classify it with a six-digit HS Code. These approaches usually follow a flat classification method. As far as the literature review covered, only two papers were published regarding the classification of HS Codes using a hierarchical structure to take advantage of the data's natural taxonomy. However, both of them resort to Deep Learning techniques. In addition, there are very few published articles regarding the implementation of the Local Classifier per Level (LCL) approach in any hierarchical classification application.

From this point, the project intends to take advantage of the topics explored in this chapter to develop an LCL HS Code Hierarchical Classification system. However, prior to the model's development, the next chapter presents a detailed explanation of the problem addressed in this thesis and its relevance for the company.

Chapter 3

Problem Framework and Description

This chapter presents the problem to be tackled. Section 3.1. focuses on the description and critical examination of the AS-IS situation. Section 3.2. consists of a product discovery phase, where the customers' (fashion brands) requirements are thoroughly analyzed, going into detail about their main concerns and specifications. The section also defines the fashion brand and market that will compose the pilot project. Finally, Section 3.3. describes the project's aims where the TO-BE scenario is presented in terms of the new methodological approaches proposed, defined to bridge the gaps of the AS-IS situation.

3.1 Operation AS-IS

To understand how the HS Code Classification model integrates into HUUB's system, it is necessary to analyze the entire process undertaken each time a new product is created.

3.1.1 Product Information Flow

Currently, the brand is the entity responsible for creating HS codes for its products. It can choose to do this service in-house, delegate the function to an outsourcing company recommended by HUUB or assign the task to its supplier. HUUB is independent of the classification process and its respective accuracy in every circumstance, accepting the HS Code into SPOKE's system without any additional validation (Figure 3.1).

After the HS Codes are created and all the product information is gathered, the brand creates the products in SPOKE. This process, conducted through the "Product Creation" microservice, can be done via integration with the online stores or through an import of an excel template. In both of these alternatives, the fields detailed in Figure 3.1 are populated with the product's information. Some of the fields such as the "Product Name", "Reference", and "Model Name" are naturally open fields. Other fields, however, namely "Product Type", "Product Family", "Product Subfamily", "Product Gender", "Product Age Group", "Country of Origin", and "Product Currency", are closed fields, with several limited options the brand can choose from. This restriction is imposed to ensure normalization and standardization of the information to be used internally by HUUB.

Finally, it is worth noting that throughout this dissertation, the term "product" refers to a product model rather than an SKU, which corresponds to the combination of (product model x color x size).

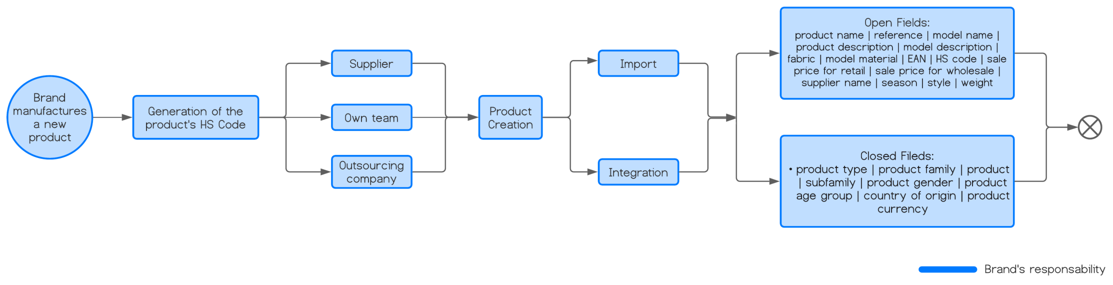

Brand manufactures a new product
Generation of the product's HS Code
Supplier
Own team
Outsourcing company
Product Creation
Import
Integration
Open Fields:
product name | reference | model name | product description | model description | fabric | model material | EAN | HS code | sale price for retail | sale price for wholesale | supplier name | season | style | weight
Closed Fileds:
• product type | product family | product | subfamily | product gender | product age group | country of origin | product currency
Brand's responsability

Figure 3.1: Information flow undertaken by a new product

The HS Codes are of fundamental importance for tax and duties compliance. Currently, HUUB has no visibility regarding this classification process, being pressed to trust a source of information not formally validated. This constitutes the main problem of the AS-IS situation and motivates the changes that must be implemented.

3.2 Product Discovery

A product discovery phase follows the mapping of the AS-IS situation. There are two stages in a product discovery process. The first one entails understanding the clients' needs and expectations by asking the question "What do customers actually want?". The second phase consists of applying these insights to develop critical products that solve a real problem for customers. Product discovery is critical in assisting product teams in determining which features or products to prioritize and build, as well as laying the groundwork for product excellence.

Thus, the purpose of this project discovery process was to assess the degree of interest of HUUB's brands in the DDP service, learn the value-added perception each brand associated with the HS Code Classification service, understand their expectations and requirements and evaluate the urgency of this possible development.

First, in the context of this dissertation, a survey was drawn up and forwarded to all of HUUB's brands. Given the relatively low relevance of the survey in the overall context of the dissertation, only the main conclusions drawn were briefly discussed. The individual questions and corresponding answers were not detailed.

After analyzing the survey, the key takeaways were addressed, which led to a deeper analysis of each brands' motivation behind having DDP, considering its top shipping markets outside the EU. The survey analysis resulted in the selection of a pilot country and a pilot brand for the implementation of the trial project.

Valuable feedback was also collected from several HUUB departments, particularly the Sales Team and the Brand Success Team, which have a direct line of communication with both HUUB's brands and potential new clients.

Finally, a conclusion regarding the relevance and urgency of a DDP development for HUUB was drawn up.

3.2.1 Survey Description

The survey had four primary objectives:

- i be conscious of the knowledge and awareness of HUUB's brands regarding international shipping and DDP;
- ii understand the importance that each brand attributed to the DDP service;
- iii identify the features that brands value most in the DDP service;
- iv learn the value-added perception each brand associated with the HS Code Classification service.

The questionnaire was forwarded to every active portfolio brand in the e-commerce segment, and all four responses were registered. The recipients did not include HUUB's only enterprise brand, as it is currently in a warehouse transition process, and the introduction of a new development idea would be considered disruptive. For confidentiality reasons, the portfolio brands were referred to as B1, B2, B3, and B4. The enterprise brand was named B5.

The questionnaire was available from the 12 of April 2021 to the 19 of April 2021. The historical data used to perform further analysis corresponded to the period between 1 of January 2020 and 3 of February 2021.

The survey significantly focused on understanding the factors behind the brands' motivation regarding DDP to assess whether developing an HS Code classification system would be an added-value service that both current and future brands would value.

3.2.2 Survey Analysis

The data insights indicate that all surveyed brands are aware of the implications of international shipping and do not find it very complex. Furthermore, all brands are familiar with the incoterm "DDP" and show medium to high knowledge in the subject. The countries selected as the most relevant to start implementing DDP align with the brand's most prominent e-commerce markets outside the European Union. When asked about the most crucial advantages of DDP, the avoidance of delays in the delivery of packages and the higher retention of international customers were pointed out as the ones with higher relevance.

As for the impact that Covid-19 had on the e-commerce business, the increase in online sales, the supply chain delays, and the request from partners to start using dropshipping were among the top responses. B1, however, reported that the pandemic had zero impact on online sales.

A search for possible causes of concern regarding DDP was also conducted. The dominant risks in implementing DDP, identified by the brands, were the decrease of online sales due to the increase in the final price presented at checkout and the possible loss of profit on account of incorrect estimates of taxes and duties.

Finally, a section about HS Codes was created to evaluate the brands' response to the possibility of HUUB offering its own integrated solution to classify goods. Three out of the four brands in the survey have their teams generating the products' HS Codes. The remainder (B1) resorts to its supplier. All of the brands either use default HS Codes or are not familiarised with the degree of detail of the codes. Two brands affirmed they would consider outsourcing this service (B2 and B4), and the remaining two said they were not interested in that offer (B1 and B3). The main reason pointed out was that the default code they are currently using is considered accurate enough for customs.

3.2.3 Main survey takeaways

Following the analysis of the questions, a critical review was conducted to extract the survey's key takeaways, reviewed as follows.

All brands understand the advantages and implications of DDP and consider it to be a value-added feature. However, it has been verified that some brands have more urgency in offering this service to their clients. This necessity is closely related to the brand's top shipping countries outside the European Union. Brands that ship a high volume of online orders to non-community countries, where the national threshold for the duties and taxes is lower than the brand's average basket price, are more likely to show interest in DDP.

It was also concluded that the more knowledge and perceived value brands attribute to the DDP service, the more they recognize the importance of having correct, accurate, and personalized HS Codes for their products.

The survey also clarified the target audience for the HS Code Classification tool developed by HUUB. Only brands that perform in-house product classification or resort to outsourcing should be considered, leaving out the brands that employ their suppliers for that job. Suppliers usually have a better comprehension of the products' characteristics and can more easily classify the goods. Therefore HUUB should not position itself to target brands that use their suppliers. However, and given the importance of having correct HS codes when performing DDP, HUUB should always run this service in parallel as a way of guaranteeing compliance.

3.2.4 Data analysis supporting the survey takeways

A deeper analysis of each brands' (including portfolio and enterprise) top shipping markets outside the EU was conducted, resorting to their historical shipments. The purpose was to illustrate through data the pattern extracted from the survey: "Brands that ship a high volume of online orders to non-community countries, where the national threshold for the duties and taxes is lower than the brand's average basket price, are more likely to show interest in DDP".

As shown in Figure 3.2, B2 and B3 would be the brands where DDP interest would be most expected, given the high percentage of shipped orders outside the EU.

However, it is also necessary to consider the thresholds for each country's taxes and duties. Every country has a tax and duty threshold corresponding to the value when a package begins paying for DAT at customs.

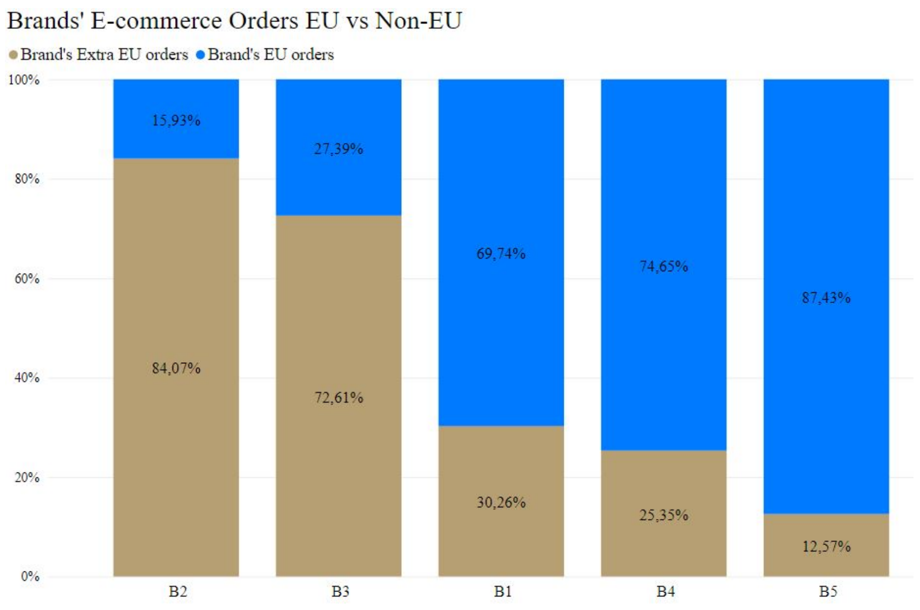


Figure 3.2: Brands' e-commerce volume order - EU vs Non-EU

For each brand, an analysis was conducted for their top markets. These markets only included the extra community countries that comprise more than 1% of the brand's total order volume in the e-commerce segment.

Figure 3.3 exemplifies the analysis conducted for brand B2. For each "brand + country" combination, the average basket price (blue line) was calculated, and the tax (brown line) and duty thresholds (purple line) were mapped. When the blue line is above any of the other two lines, it indicates a high probability of the brand's package being charged with either taxes or duties (or both) at customs.

United Kingdom, Australia, and the United States are the top non-member countries B2 sends orders to. Their relative importance in the brand's total e-commerce order volume is, respectively, 51,28%, 15,97%, and 8,41%. Each country has different taxes and duties thresholds. The United Kingdom, for instance, has a tax threshold value of 0€ and a duty threshold value of approximately 157€.

Interpreting the graph in Figure 3.3, it is clear that the United Kingdom is a market where the DDP will be relevant since the average price of each order is higher than at least one of the two thresholds. The same analysis can be extended for the remaining brands. The results are summarized in Figure A.1 of Appendix A.

A more detailed view can be obtained by considering the actual price of orders instead of the average price, making this fine-grained analysis very relevant. For example, in a scenario where

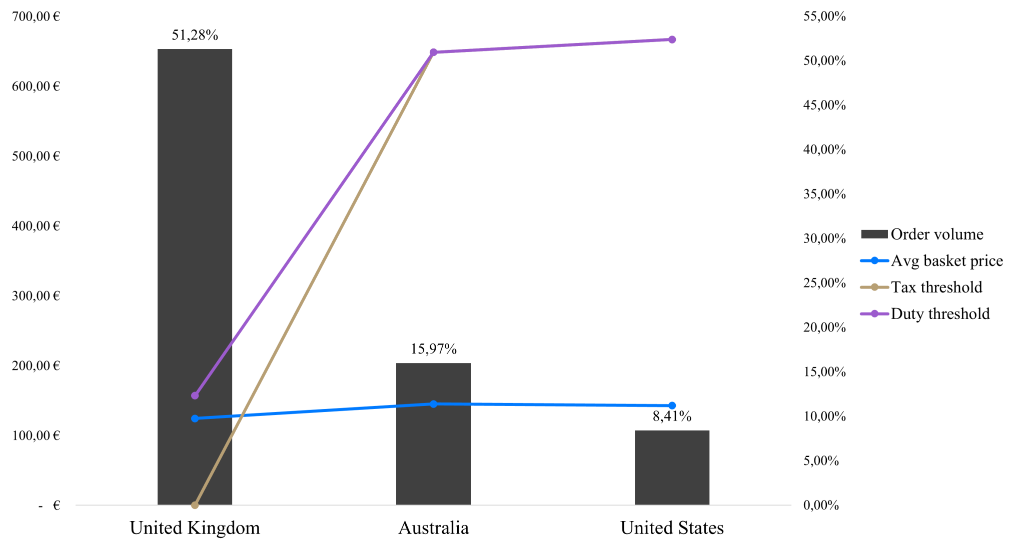


Figure 3.3: Percentage of orders per main market, tax and duty threshold and average basket price of brand B2

a country+brand combination has an average order price below the thresholds, if the price of the orders has high variability, there might still be a considerable proportion of orders paying DAT, which will make DDP relevant. Hence, the percentage of orders above each of the two thresholds was evaluated for all of the top markets of each brand.

B2 results are shown in Figure 3.4. The graphs synthesizing the results from every brand can be found in Figure A.2 of Appendix A.

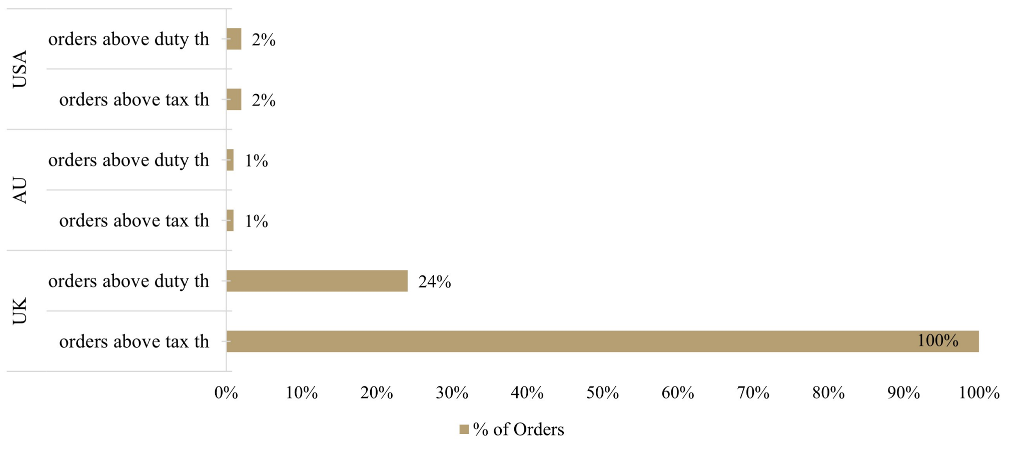


Figure 3.4: Percentage of orders above Tax and Duty Thresholds for B2's Top Markets

All brands have at least one country where DDP would be required. The United Kingdom is the market where this development gains the most relevance, whether because of its low threshold values or the relatively large volume of shipped orders for most brands. Additionally, this market has posed a significant challenge for numerous brands due to the recent trade restrictions imposed by Brexit: because it is no longer a European Union member, all commercial trades between the UK and EU countries are eligible to be levied with duties and taxes.

This outcome solidified the conclusions previously drawn by the survey. The brands that showed more rooted interest in DDP in the survey are the ones that have a higher volume of orders for non-community countries, where the thresholds for taxes and duties are lower than their average basket price.

The UK posses extreme relevance for B2, given the magnitude of that market in the overall context of the brand (around 52% of the total volume of orders). Because of the urgency the DDP development represents to the brand, the combination "B2 + the United Kingdom" was chosen to be the pilot for the project.

3.2.5 Final conclusions

Small and medium-sized enterprises (SMEs) typically do not understand the main advantages and implications of DDP and usually have associated fears and concerns that need to be demystified. HUUB's Sales Team had already broadly detected this pattern, where brands with lower order volumes typically do not have many operational requirements, as they expect to be advised by HUUB. Therefore, it is HUUB's job to advise the brands and explain how and why DDP may be a value-added service to boost their international e-commerce business.

SMEs that are already feeling the pressure of international markets (mainly due to Brexit) have been forced to understand the advantages and implications of DDP and start requesting it. Enterprise brands, including those with a low volume of orders to non-community countries, also view DDP as an essential feature to guarantee scalability and expand to new markets.

The survey and the consequent analysis were critical to ensuring HUUB can align its planning with the brands' expectations and develop a feature as close to the brands' needs as possible. These were also critical for HUUB to access the project's priority, given the limited internal capacity and the constant arising of requirements of new features or services for the platform.

3.3 Operation TO-BE

The product discovery phase clarified that a DDP service would benefit HUUB's brands in multiple spheres. This understanding reinforced the need for a mechanism to generate the HS codes for the products.

The TO-BE situation proposed after implementing this project concerns the task of classifying the newly created products with a six-digit HS Code, as is proposed in Figure 3.5.

The decision for the model to classify the products with only a six-digit code (HS Code) was based on the premise that the HS Codes are universally used. Each country may add suffixes to the original code to suit its tariff and statistical needs, creating more extensive national codes; however, all the countries share the same first six digits.

The classification system has essentially two purposes:

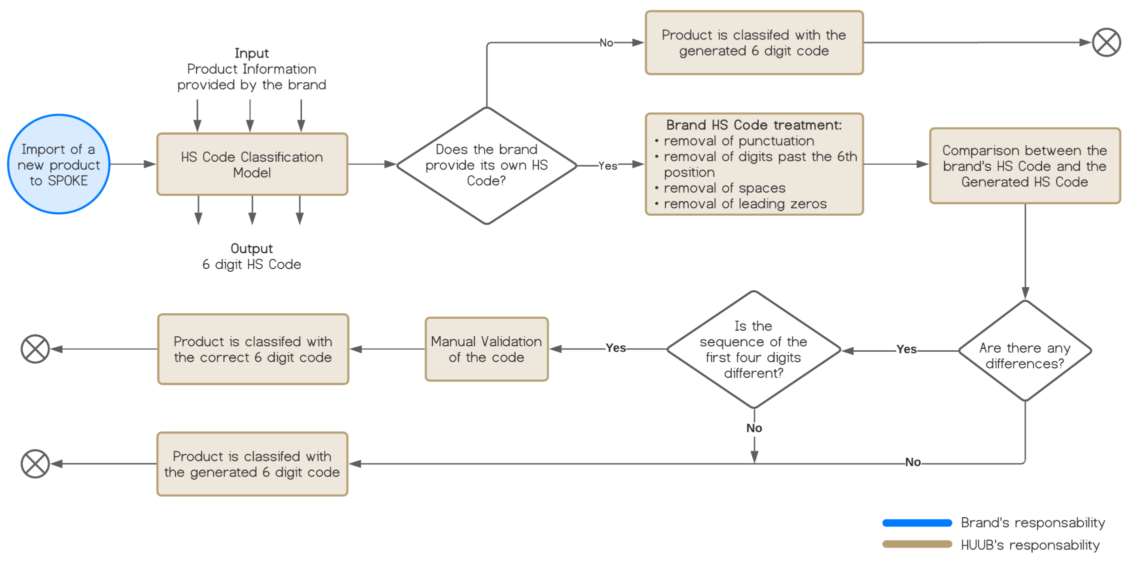

Import of a new product to SPOKE
Input
Product Information provided by the brand
HS Code Classification Model
Output
6 digit HS Code
Does the brand provide its own HS Code?
No
Yes
Product is classifed with the generated 6 digit code
Brand HS Code treatment:
• removal of punctuation
• removal of digits past the 6th position
• removal of spaces
• removal of leading zeros
Comparison between the brand's HS Code and the Generated HS Code
Are there any differences?
Yes
No
Is the sequence of the first four digits different?
Yes
No
Manual Validation of the code
Product is classifed with the correct 6 digit code
Product is classifed with the generated 6 digit code
Brand's responsability
HUUB's responsability

Figure 3.5: Map of the product's information process after the project implementation

- Generation of a six-digit HS Code for the new products created by brands that subscribe to HUUB's classification service;
- Validation of the HS Codes of the new products created by brands that provide their own codes. This authentication is an essential mechanism for HUUB to ensure the compliance of the information provided by external sources.

If the brand requires HUUB to generate its HS Codes, a six-digit code is assigned to each new product created in SPOKE. Alternatively, if the brand provides its own codes a priori, a validation is required.

The brand's code is compared with the internally generated one and approved if both codes match. If there are any inconsistencies between the two, SPOKE will alert the brand that some discrepancies were detected. The internally generated code is suggested and the brand can manually select which of the two codes it considers to be more suitable. This corroboration is vital to maintain classification compliance while enriching the model. Every time a discrepancy is registered, the teams responsible for maintaining and curating the model will determine the source of the truth and add the new correctly classified data to the model. The discrepancies will gain even more relevance when the brand chooses to keep its own HS Code, as this indicates a lack of reliability and trust in HUUB's service. This approach will allow the model to start reporting better accuracy in the classification of currently misclassified codes.

However, if the inconsistencies are identified in the fifth or sixth digits (i.e., the sequence of the first four digits is equivalent), no validation is required, and the product is assigned with the internal code. This inconsistency indicates that the model only failed to classify the product's material, which is not a critical issue for customs as it does not significantly impact the taxes and duties applied. Therefore, the discrepancy is deemed minor and the generated code is used.

Chapter 4

Methodology

In this chapter, the methodology used to approach the HS Code Classification problem is defined. The CRISP-DM (Cross-Industry Process for Data Mining) methodology was used to structure the project.

After stating the desired outputs of the project in Section 3.3, an initial description of the dataset is provided, and the preprocessing steps required to prepare the data for both the Machine Learning and the Rule-based algorithms are detailed. Both of these algorithms are presented with further detail, with the rule-based approach serving as a baseline model for the different ML algorithms tested. Additionally, this section comprehensively explains the hierarchical modeling and evaluation processes developed to tackle the HS Code classification problem. The chapter ends with an overview of the different scenarios to be compared with respect to the prediction task. The deployment of the model is further explored in Section 6.1.

4.1 Data Preprocessing

4.1.1 Dataset characterization

Before diving deeper into the data preprocessing procedures, it is crucial to understand the dataset in use. The dataset was custom built for the context of this project through a SQL query that extracted the desired data. It contained 223.285 instances and six features, including the information of all the products ever created in HUUB up to 01-03-2021.

The selection of the features used in the dataset was made according to an analysis of the HS Code Rule-based system, where the necessary information to classify the code in each of its three hierarchical levels (chapter, heading, subheading) was identified, as shown in Table 4.1. After the preprocessing steps, all the features were classified as categorical and single-labeled (each feature accepts solely one class).

Table 4.1: SPOKE features necessary to each HS Code hierarchical level

HS Code Level	**Necessary SPOKE features**
Chapter	type; sub_family; fabric; material: age_group
Heading	type; sub_family; gender; material: age_group
Subheading	material; gender

The dataset provided encompasses the following features:

material: specification of the product's material. The material of a garment is the substance that goes into making a fabric. For instance, the material cotton is used to create a denim fabric. This is an open field that brands fill in as they see fit, without any predefined standardization. Examples of inserted input for this feature are "100% cotton", "100 CO" and "100% algodón". Each of these entities expresses the same concept (cotton) in varied formats.

gender: specification of the product's gender, having three distinct options: "Male", "Female", or "Unisex".

age_group: specification of the product's age group, ranging from "Baby" and "Baby/Kid", to "Kid", "Teen", "Adult" and "All".

type: states the product's category. It can be either "Clothing", "Accessories", "Underwear Nightwear", "Beachwear", "Footwear", "Swimwear", "Swim accessories", "Homewear" or "Stationery".

sub_family: corresponds to the product's subcategory and has 94 different categories. Some examples of these classes are "Shorts", "Jacket", "Trainers" and "Bath Towels".

fabric: specification of the product's fabric, which is used to make the garments. This is a closed field with 19 different categories. These categories, however, lack a standardization procedure since some of them are overlapping. "Woven", "Woven + Knitted" and "Outer: Knitted, Padding: Woven" are some of the possible values for this feature.

The dependent variable is categorical and corresponds to the HS Code of the product. After analyzing the distribution of chapters (first two digits of the code) present in the dataset, chapters with very few observations outside the normal scope of the categories marketed by HUUB (fashion clothing and accessories) were identified. These chapters were not considered for the analysis as they bring minimal benefit compared to the cost of mapping their rules in the rule-based model. Adding a single new chapter has an exponential cost in terms of the headings and sub-headings subsequently required. The selection process for the chapters to be considered was based on the twofold criteria:

i chapters that are directly related to HUUB's core business (apparel, accessories, and footwear)

ii chapters with more than 500 observations (note: chapters in the same section are always kept; even if a chapter has few observations, it is kept if there are other chapters in the same section that have a relevant number of observations)

Consequently, only chapters 61, 62, 63, 64, 65, 42, and 43 were considered for the analysis. Their frequency of observations is represented in Figure 4.1, and Table 4.2 details which fashion category they entail. This narrowing process reduced the dataset to 199.931 instances.

Table 4.2: Section and Chapter descriptions for the chapters considered in the analysis

Section	**Chapter Description**
XI Textiles and textile articles	**61** Articles of apparel and clothing accessories, knitted or crocheted **62** Articles of apparel and clothing accessories, not knitted or crocheted **63** Other made-up textile articles; sets; worn clothing and worn textile articles; rags
XII Footwear, headgear	**64** Footwear, gaiters and the like; parts of such articles **65** Headgear and parts thereof
VIII Raw hides and skins, leather, furskins; saddlery and harness; travel goods, handbags and similar containers	**42** Articles of leather; saddlery and harness; travel goods, handbags and similar containers; articles of animal gut (other than silkworm gut) **43** Furskins and artificial fur; manufactures thereof

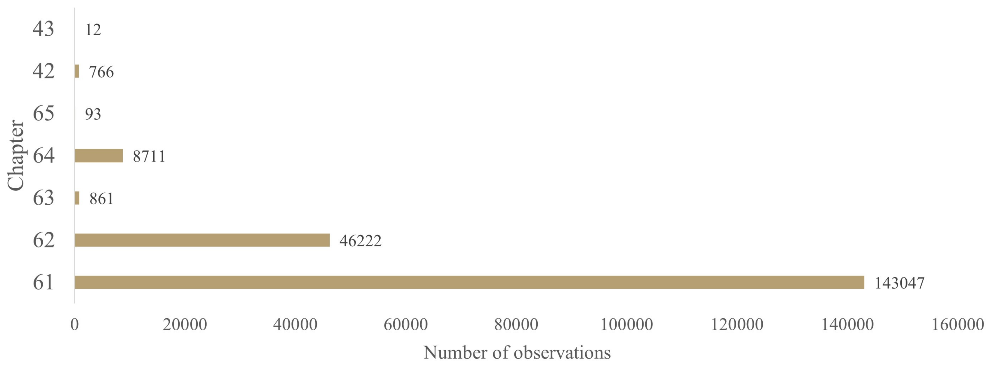


Figure 4.1: Frequency of observations for the chapters considered in the analysis

4.1.2 Data Preparation

A large portion of time in this project was spent preparing the data for both the Machine Learning algorithms as well as the Rule-Based algorithm.

Data cleaning

The dataset only contained approximately 6% of missing values (Figure 4.2 shows the missing data distribution per variable). More than 5% of the total number of observations containing missing values were concentrated in the predictive variable (hscode), making it unfeasible to apply imputation techniques to substitute these values. The remaining 1% was not deemed significant enough to justify the use of any approach. For these reasons, it was decided to eliminate all the observations that contained missing values, resulting in the exclusion of 13250 rows of the dataset, which corresponds approximately to 6% of the original dataset.

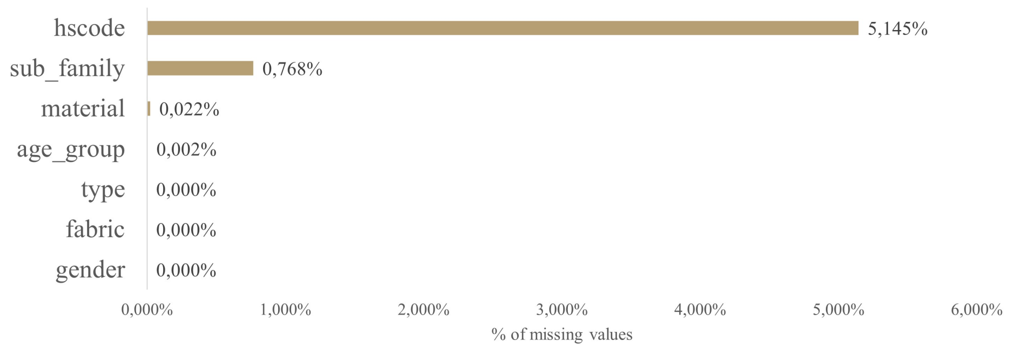


Figure 4.2: Percentage of missing values per variable

Record Sampling

The next step was record sampling, which consists of removing records with erroneous or less representative values to make predictions more accurate.

The feature "material" had 1677 observations with values equal to "0" or "test", which were removed from the dataset since they were non-valuable records. In addition, the feature "fabric" registered 21 records with erroneous information, being also excluded.

In the prediction variable (HS Code), the following errors were identified:

i "00" combination at one of the three levels of the HS Code (e.g., code "610013" has a "00" combination at header level). This possibility does not exist in the real classification system, making those codes incorrect (2792 observations);

ii codes equal to "0". There is no code equivalent to this representation in the real classification system. The "0" acts as a missing value (6797 observations);

iii codes with less than six digits since the objective is to classify products with a six-digit HS Code (11495 observations);

All the observations containing any of the aforementioned types of errors were eliminated, leading to a reduction of the dataset by 21084 rows.

Data formatting and aggregation

Data formatting is the process of transforming the data into a standard format to guarantee the consistency of records and allow for further analysis and comparisons. Data aggregation has the general goal of making each feature homogeneous and ensuring the use of a single value to represent the same concept. In this setting, the features "fabric", "material" and the target variable required some data formatting and aggregation.

Although "fabric" is a closed field, it required data formatting. First, two typographical errors were identified and corrected (kniited $\rightarrow$ knitted, and wowen $\rightarrow$ woven), leaving the feature with 17 different categories.

Each one of these 17 categories was later aggregated into two general categories. In order to classify fashion products, it is only necessary to distinguish between garments with fabric = "knitted" and garments with fabric = "woven". These two families divide the fabrics according to the method that is used to construct them. Knitting is the construction of the elastic and porous fabric created by interlocking yarns and employing needles. Woven fabrics are made by using two or more sets of yarn interlaced at the right angles. Therefore, each record was classified as woven or knitted fabric and aggregated into one of these two general categories. The assignment (shown in Table 4.3) was conducted based on two criteria:

i the record was assigned to the fabric family whose characteristics represented a better match;

ii if the first step was inconclusive, the fabric was attributed to the family that most brands assign it to, according to historical records.

Table 4.3: Fabric classification

Knitted	**Woven**
"Knitted"	"Woven"
"Woven + Knitted"	"Outer: Knitted, Padding: Woven"
"Sweat + Interlock"	"Outer: Knitted, Inner: Woven"
"Jersey"	"Leather"
"Nylon"	"Suede"
"100 % Organic Cotton"	"Velvet"
"Jersey"	"Denim"
"Sweat"	"Woven + Jersey"
"Interlock"	

"Material" is the only feature representing an open field; hence it required the most time to preprocess.

The algorithm will only be interested in the primary material that composes the garment; therefore, the standardization process aimed to extract the name of the dominant material for each of the products. To exemplify, if an observation has the material name "63% leather, 37% cotton", it should be retrieved "leather" as the relevant material for the analysis. Several preprocessing actions were necessary to achieve this intent. Table 4.4 specifies an example for each of the multiple problems identified with the data that required correction. These only illustrate the individual occurrences, but each observation can combine multiple different issues.

Table 4.4: Formatting issues identified in the feature "material"

Problem	**Example**	**Solution**
mixture of lower and upper cases	100% BAMBOO rayon	every string in lower case
no separation between words	100%cotton(organic)	add space between punctuation marks
different values representing the same entity	90% co, 4% el, 6% pl	abbreviation mapping
typographical errors	90% polyamide 10% elasthane	typographical errors correction
words in different languages	100% algodón	word translation

It was, therefore, necessary to take different actions in order to correct every error presented. The feature's data type is a string, and thereby the first step was to convert the string into lower case and add a space before and after every punctuation mark (e.g., 100%(cotton) → 100 % (cotton)). This was done so that the words in the string, specifically the material names, became isolated.

In order to ensure the single existence of a value to represent the same concept, an abbreviation file containing 149 different materials was used to substitute the materials' abbreviations in the string with the complete material name (e.g., co → cotton). Following that, the typographical errors were corrected (e.g., coton → cotton), and the words in foreign languages translated into English (e.g.,algodon → cotton). Table 4.5 shows how the previous issues were corrected after this preprocessing step.

Table 4.5: Correction of formatting issues identified in the feature "material"

Example	**Correction**
100% BAMBOO rayon	100% bamboo rayon
100%cotton(organic)	100% cotton (organic)
90% co, 4% el, 6% pl	90% cotton, 4% elastane, 6% polyester
90% polyamide 10% elasthane	90% polyamid 10% elastane
100% aldodón	100% cotton

Subsequencially it was necessary to extract the name of the primary material. The problem was that multiple observations contain more than one material, and each observation comprises various words other than the material name. The solution was to identify and choose the material name that represents the most significant percentage in the constitution of the garment. For instance, in the record "90% polyamid 10% elastane" the desired output would be "polyamid" since it accounts for 90% of the garment's constitution. This extraction process was performed with recourse to regular expressions. Regular expressions (ReGex) are used to identify whether a pattern exists in a given sequence of characters (string) and locate its position in a corpus of text, helping to manipulate textual data.

Firstly, all spaces were removed, and six possible patterns in the material records were identified and defined in Table 4.6.

Table 4.6: ReGex patterns indentified in the feature "material"

Pattern	Explanation
n%**m1**,n%**m2**,n%**m3**	3 materials separated by a comma
n%**m1**n%**m2**n%**m3**	3 materials not separated by a comma
n%**m1**,n%**m2**	2 materials separated by a comma
n%**m1**n%**m2**	2 materials not separated by a comma
n%**m1**	1 material with percentage value
m1	1 material without percentage value

The list with the 149 materials that was previously used for the abbreviation mapping was a resource to identify material names within the set of words belonging to each string. All the words that were not included in the list were not considered as material names.

For each observation, the pattern to which the string corresponds within those six possibilities is identified. Then, within the corresponding pattern, the names of the materials are identified between the group of words and characters of the string. The material with the higher percentage is then selected. If a description has an inconclusive percentage comparison, the first material is selected as the primary material by default (e.g., 50% cotton 50% silk → cotton). If a description has no percentages, the model assumes the latter to be the primary material (e.g., cotton and leather → leather).

The final output was a new column that attributed a primary material to each of the records.

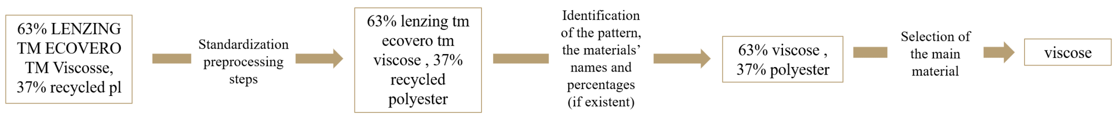

63% LENZING TM ECOVERO TM Viscosse, 37% recycled pl
Standardization preprocessing steps
63% lenzing tm ecovero tm viscose , 37% recycled polyester
Identification of the pattern, the materials' names and percentages (if existent)
63% viscose , 37% polyester
Selection of the main material
viscose

Figure 4.3: Standardization process of the variable "material"

Observations in which no material name has been identified or do not fit the previously described patterns were removed from the dataset (416 removed records).

Finally, the HS Code classification of fashion products, in the rule-based system, distinguishes only four different material families: "Cotton", "Wool", "Man Fibers", and "Others". As a result, each of the different materials was allocated to one of these four categories. This aggregation was based on the materials' properties and used for further analysis (Table 4.7).

Table 4.7: Material Classification in four general categories

Cotton	Wool	Man Fibers	Others
silk	cashmere	viscose	linen
cotton	rabbit fur	polyester	bamboo
terrycloth	wool	elastane	leather
flannel	melton	polyamid	silver
flannelette	merino	acrilic	cupro
poplin	polyolefin	velvet	suede
twill	nylon	tencel	(...)
jersey	viscose	mesh	
		fleece	

The final phase was to analyze the target variable. In order to ensure its standardization, all spaces between numbers were eliminated (e.g., 61 13 01 → 61130), the dots separating the hierarchical levels were removed (e.g., 63.04.30 → 630430), and digits beyond the sixth position were excluded (e.g., 63043010 → 630430).

Both the Rule-based and the Machine Learning approaches were submitted to all of the preprocessing steps listed above.

4.2 Rule Based Model

The baseline model for this dissertation is a Rule-based approach that intends to mimic the traditional hierarchical classification structure for the Harmonized Coding System.

The objective was to use the categories in the HUUB dataset (explained in further detail in Section 4.1.1) to predict the products' classification.

Based on the rules defined by the World Customs Organization (WCO), a translation was performed between the closed categories used by HUUB and the words and terms used by WCO in the official rules. The category "trenchcoat", belonging to the variable "sub_family", was chosen to serve as an example. Although the word "trenchcoat" is not explicitly defined in the HS Codes official rules, in the model, this category was associated with the word "coats", which is already defined in the rule system. Fortunately, multiple categories in the dataset were already in conformity with the words and expressions defined in the rules, so few manual translations or adaptations were necessary.

The algorithm was constructed in the following way. First, the chapter was defined. Next, for each of the chapters, the heading was classified, and, consequently, for each of the combinations "chapter+heading", the subheading was determined. For instance, a clothing item classified with chapter "61" can only be classified with headings belonging to that chapter. However, and because each child node shares the same nomenclature (e.g., chapter "61" and chapter "62" both have headings with the name "01", "02", "03"), if a code is read isolated, it can hold multiple meanings. For example, a heading equal to "02" can have multiple meanings, depending on the chapter preceding it. Likewise, following the same logic, a subheading equal to "01" can signify different things, according to the previous "chapter+heading" classification.

The model structure, which can be visualized in Figure 4.4, clearly illustrates the incremental cost and effort to add an extra chapter or even a heading to the rule-based classification system.

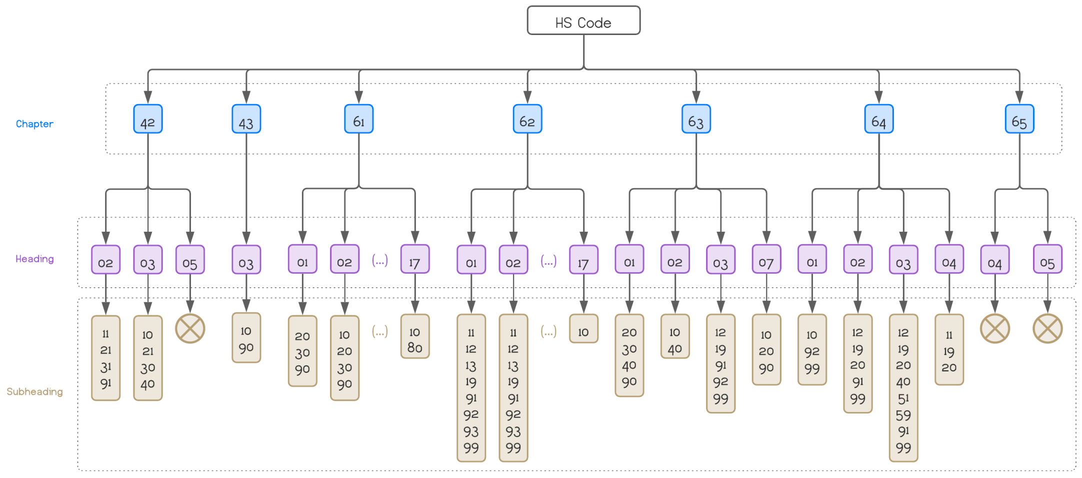


Figure 4.4: Rule-based Structure

4.3 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data to discover patterns, spot anomalies, test hypotheses, and check assumptions with the help of summary statistics and graphical representations. It is a way to understand the data before any modeling is put in place and gather the most insights.

4.3.1 Class Frequency

Initially, the six-digit HS Code class frequency was analyzed and summarized in the first graph of Figure 4.5. From the 353 possible HS Codes classes, there were distinguished 126 with only one observation. This poses a problem for a Machine Learning model since these observations can only be assigned to either the testing or the training dataset. If the observation is attributed to the training set, the model is unable to assess how good it is in predicting said class. By contrast, if the observation is included in the testing set, the model will be unfamiliar with that observation and fail to predict its HS Code class.

Moreover, classes with very few observations (in this project's context, classes with less than 100 observations were considered poorly populated) do not have enough critical knowledge to train the model accurately. Classes with few observations would also be detrimental if an under-sampling technique were to be applied since all classes would have the same number of observations as the smallest one. In the dataset, 110 classes had less than 100 observations.

In both cases, the classes were eliminated, resulting in a target variable with 117 different classes (the second graph of Figure 4.5 illustrates the new class distribution). However, it is worth noting that removing these classes may decrease the robustness of the model and its future predictive ability, having difficulties in the generalization of new data.

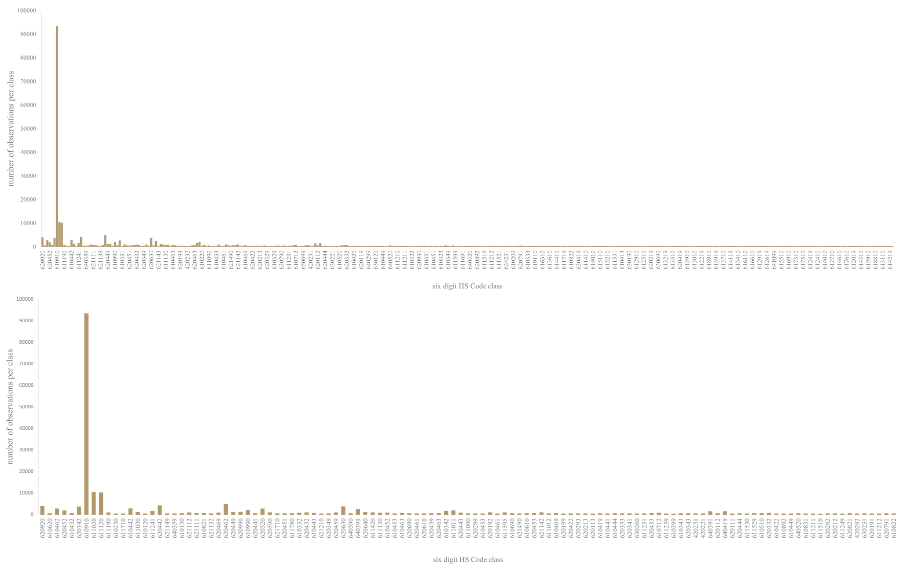


Figure 4.5: The first graph illustrates the initial class distribution of the dataset whereas the second graph represents the final class distribution

4.3.2 Data Imbalance

An imbalanced classification problem is one in which the distribution of examples across known classes is skewed or biased. The abundance of examples from the majority class (or classes) can undermine the minority class. The generality of the classification Machine Learning algorithms is built and demonstrated on problems that assume an equal distribution of classes. This means that a naive dataset handling may only focus on learning the characteristics of the overflowing classes, disregarding the examples from the minority class(es).

Analyzing Figure 4.5, it is clear that the dataset has a natural asymmetric input. The majority class is the "610910" HS code, with 93077 observations, which corresponds to approximately 50% of the total observations. Suppose a dummy model was developed, which classified every observation with the "610910" HS Code. The model would return an accuracy score of 50%, which could be satisfactory; however, the precision would be unacceptable since the model would not have the capability to predict the minority class(es).

Furthermore, this dataset distribution is not representative of the current reality in HUUB. The peak volume in class "610910" was caused due to the historical data of a particular brand, which is no longer in HUUB's portfolio. The brand was running a make-to-order business, creating a new SKU each time a product was sold, with the only available product being cotton T-shirts (correspondent to the code "610910"). This is one of the most prominent justifications for the dataset imbalance. None of the past nor present HUUB's brands have a similar product typology distribution. Therefore, the asymmetry of the dataset corresponds to an unrealistic portrayal of current reality.

Resampling techniques can be applied to balance the data. It is worth noting that the resampling techniques are exclusively applied to the training set. If this is done before the split of data in training and test sets, the quality of the model can be compromised due to data leakage, causing overfitting and poor generalization.

The resampling can be achieved by increasing the frequency of the minority class(classes) - oversampling - or by reducing the majority class frequency through undersampling techniques.

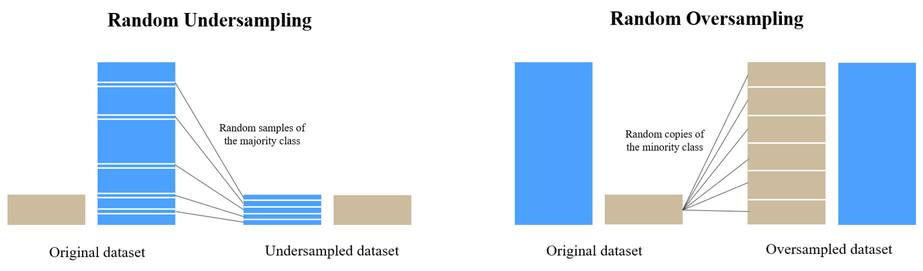


Figure 4.6: Random undersampling and random oversampling techniques

The most simple sampling techniques were chosen, namely, random undersampling and random oversampling. The random undersampling includes selecting samples from the majority class(es) at random and removing them from the training dataset. It is possible to perform an undersampling technique only to the majority class or apply it to all the classes except for the minority class. The main disadvantage of random undersampling is that it can omit potentially valuable data for the induction process. In contrast, random oversampling randomly duplicates minority class samples in the training dataset, which can lead to overfitting in some models.

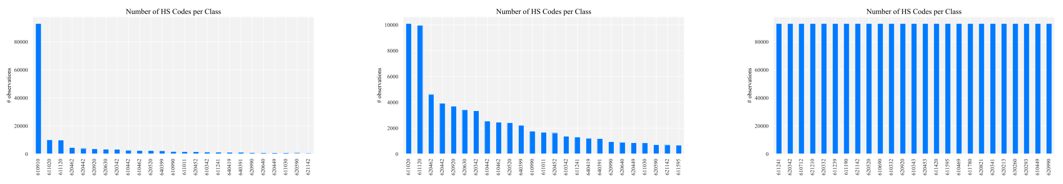
The initial dataset (with 183736 observations) was split into a training set (128615 observations) and a testing set (55121 observations), with Section 4.5.1 going over the process in greater detail. To empirically test which was the most appropriate technique to apply, three different training datasets were used. The holdout set (test set) remained unchanged for the three different scenarios.

"Imbalanced" training set
The original dataset with no modifications to its constitution, with 128615 record samples.

"Random undersampling of the majority class" training set
In this training set, the majority's class observations are removed to match the class with the least number of samples, with the rest of the classes remaining untouched. The technique results in a "semi-imbalanced" training set, with a considerably lower difference in frequency between the majority class and the remainder, when compared to the initial dataset distribution. This training set has 63531 record samples. In this case, the training and the testing set have a very similar proportion, which is not ideal since it may cause underfitting. Given that a different initial split (e.g., 80/20) would undermine the future scenario comparations, the underfitting risk was acknowledged, but this technique was accepted as a viable solution.

"Random Undersampling followed by Random Oversampling" training set
A random oversampling of all the classes except the majority class would result in a dataset with over ten million observations. While most machines lack the memory capacity to complete a task of that magnitude, the models' training time would also be unendurable. The solution was first to perform a random undersampling to the majority class, followed by a random oversampling technique, in which every class was augmented to match the number of observations of the majority class. As a result, the training dataset ended with 827658 observations.

A predictive class distribution overlook for each of the three training datasets for the first 25 classes can be found in Figure 4.7.



(a) (b) (c)

Figure 4.7: (a) "Imbalanced" training set (b)"Random undersampling of the majority class" training set (c) "Random Undersampling followed by Random Oversampling" training set

It is also worth mentioning the effects of resampling strategies during the model selection process, covered in further depth in Section 4.5.4. After the dataset is split into k folds, the resampling method employed is only applied to the training folds (k-1 folds), leaving the test fold with the original class distribution. This ensures that the model selection phase uses untempered test sets to assess the performance of the ML algorithms.

4.4 Feature transformation

Data encoding and feature scaling are the last data preprocessing steps required for the ML models. The previous steps were detailed in Section 4.1.2, given that they are shared by both the Rule-based and the Machine Learning models.

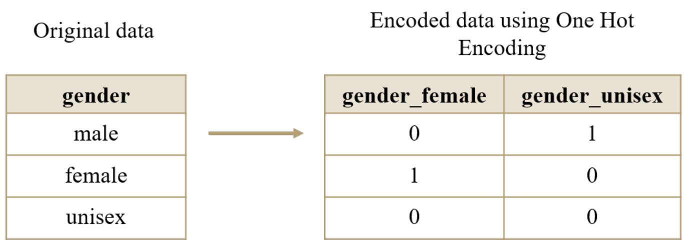
Data encoding

Numerical data involves features that are only composed of numbers, such as integers or floating-point values. Categorical data, on the other hand, consists of features containing label values that can be divided into two different types of variables: nominal and ordinal. A nominal variable does not have an inherent order to its categories, while an ordinal variable does. Almost every Machine Learning model requires all input variables to be numeric, which means if there are any categorical variables, these must be encoded before being used with ML algorithms.

All the features in the dataset are nominal categorical variables. In this case, a One-hot encoding or a dummy variable encoding can be applied. The first method was chosen for this project. The One-hot encoding system generates one binary variable for each category, returning a vector for each unique value of the categorical column. The only drawback is that One-hot encoding demands the categorical variables to be numerical labels, which requires a first pre-processing step of integer-encoding. However, after this transformation, One-hot encoding can be applied.

One advantage of the One-hot encoding method is that it allows to fit the object on the training set and reuse it to transform the test set, thus solving the problem of having different features in the training and the testing sets.

The problem with this representation is that it induces multicollinearity into the dataset. Multicollinearity occurs when two or more independent variables in the dataset are correlated with each other. For example, if one feature is composed of three categories (A, C and C), [1,0,0] represents category "A", [0,1,0] represents category "B", and [0,0,1] represents category "C". However, there is no need for all three variables to be encoded. Category C, for instance, can be represented by the absence of the other two, e.g., [0,0,0]. To overcome this multicollinearity issue, one of the binary variables is dropped for every dataset feature. This representation is depicted in Figure 4.8.


Original data

gender
male
female
unisex

Encoded data using One Hot Encoding

gender_female	gender_unisex
0	1
1	0
0	0

Figure 4.8: One hot encoding with a "drop first" approach for the feature "gender"

Feature Scaling

Feature scaling refers to the process of placing the values of the independent variables in the same range or same scale so that no variable is dominated by the other. Considering that the entire dataset comprises categorical dummy variables that range between 0 and 1, no scaling preprocessing step was required.

4.5 Hierarchical Classification Model

Initially, a conventional flat classification approach was tested out, using only one classifier to predict the entire six-digit HS Code. However, it was rapidly discarded due to the low accuracy results (less than 50% of accuracy). More importantly, from a business perspective, this approach added virtually no value. The customs attribute the most relevance to the first four digits of the HS Code, with a particular emphasis on the first two. Therefore, it would be beneficial to have a model that could return multiple code levels, depending on the degree of granularity required. In addition, the HS Code Classification problem has a natural tree hierarchical structure that can be easily visualized in Figure 4.9. Each of the subheadings belongs to a specific heading, which, in turn, also relates to a particular chapter. Given its taxonomic characteristics, a hierarchical classification structure was chosen to take advantage of the natural information about the data hierarchy.

Using the algorithm characterization framework proposed by Silla and Freitas (2009), <Δ, Ξ, Ω, Θ>, the HS Code Classification problem was defined in the following manner:

$\Delta \rightarrow$ **Single Path Prediction (SPP)**
The algorithm can only assign each data instance with one predicted label (single-label problem). For instance, one record cannot be classified with chapter 61 and chapter 62 at the same time.

$\Xi \rightarrow$ **Mandatory Leaf Node Predictions (MLNP)**
The algorithm is designed to always classify a leaf node.

$\Omega \rightarrow$ **Tree**
The taxonomy is organized into a tree structure.

$\Theta \rightarrow$ **Local Classifier per Level Structure (LCL)**
The local classifier approach was chosen given its simplicity and generality capacities while also being able to capture the predefined data taxonomy with considerable granularity. Within the local classifiers approaches, the Local Classifier per Level (LCL) was considered the most appropriate because of its small number of required classifiers and its suitable fit for the problem structure.

Therefore the HS Code Product Classification problem can be summarised by the 4-tuple <SPP, MLNP, Tree, LCL>.

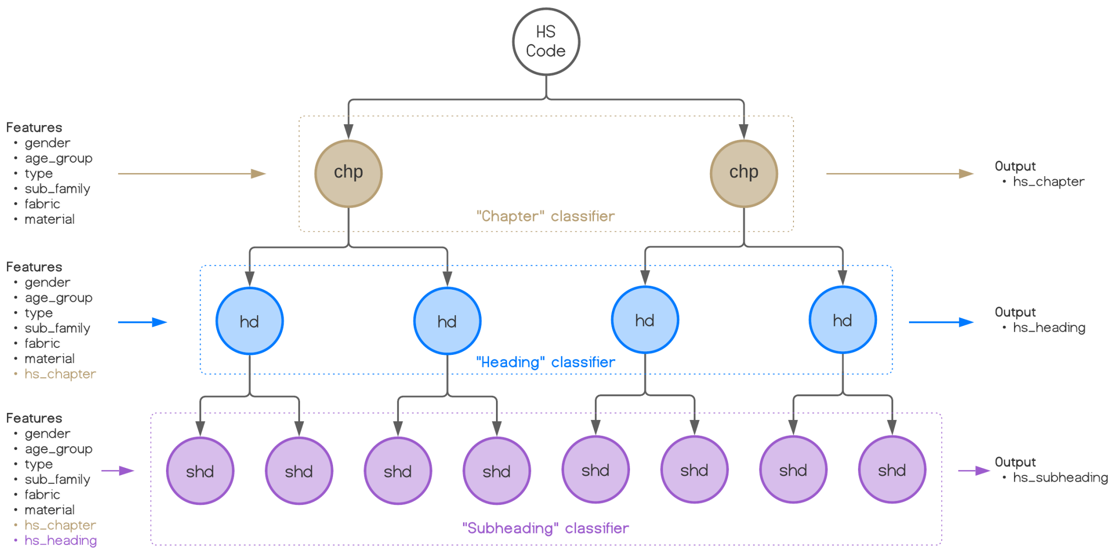


Figure 4.9: HS Code classification hierarchical structure

4.5.1 Model Structure

Three independent classifiers, one for each HS Code level (Chapter, Heading, and Subheading), were developed. The "Chapter Classifier" takes as input the standard features of the dataset, intending to predict the two digits corresponding to the Chapter. Following, the "Heading Classifier" uses as input the dataset features as well as the two digits of the Chapter in order to predict the heading code. Finally, the "Subheading Classifier" considers as input the dataset features, the chapter code, and the heading code to predict the two digits that constitute the subheading. Figure 4.9 illustrates the described process.

The final output of the model will be a six-digit predicted code, concatenating the predictions of the two-digit codes of each of the three classifiers.

In Machine Learning, the goal is to achieve a model that generalizes well on new unseen data. The dataset is split into a training set, where the model learns good values for all the weights and the bias from labeled examples and a test set that evaluates the model's performance.

For this project, 70% of the total data was used for training, and 30% was reserved for the testing phase of the model. This partitioning was stratified, ensuring that the predicted class (six-digit HS Code) distribution was identical in both the training and testing sets. This step gains particular relevance in the imbalanced training set.

4.5.2 Model Training

For the model training, two separate approaches were designed.

The first approach entailed using real output data from the previous classifier to train the subsequent one. For the first classifier ("Chapter"), only the features containing the brand's information were used in the training phase. The "Heading" classifier's output, which is immediately below, depends on what was defined in the previous level. In an attempt to reflect this dependency, the "Heading" classifier, besides the standard features, also took as input for the training phase the actual two-digit chapter code of that observation. Following the same course of thought, the "Subheading" classifier received as training input the actual chapter and heading codes, along with the standard features.

The use of real data of the predictive variable as input for the succeeding models is justified by the fact that, in the training phase, the intention is to avoid the introduction of prediction errors made in antecedent models. The three models are treated as three separate problems, each with the goal of predicting its own sub-code with the highest level of accuracy.

Consider the following scenario to demonstrate this reasoning. The "Chapter" classifier reports an accuracy of 94% in predicting the first two digits of the HS code. The use of this output as training input to the "Heading" classifier would introduce a 6% error to the second model. This training error would be fooling the model, transmitting a false relationship or pattern between the predicted variable (the 3rd and 4th digits, correspondent to the heading) and one of the input variables (the 1st and 2nd digits, correspondent to the chapter). This propagation cycle perpetuates to the final classifier. The training sequence for this approach is summarized in Figure 4.10 for a better understanding.

The second approach followed a traditional method, applying the predicted data from the previous model as the input data for the subsequent model. In this context, the output of the first classifier (the 1st and 2nd digits predicted) is used as training input for the second classifier. In the same way, the output of the first (the 1st and 2nd digits predicted) and second (the 3rd and 4th digits predicted) classifiers is utilized for training the third model, which aims to predict the 5th and 6th digits. This sequence can be found in Figure 4.11. The problem with this approach is the aforementioned error propagation in the training phase.

As a final note, it is important to refer that the predicted output for each of these models is calculated based on the training set, as this phase has the purpose of training the models, not evaluating them. For reference purposes, the first approach will be labeled the "Real Data" approach, and the second one will be referred to as the "Predicted Data" approach.

4.5.3 Model Evaluation

Both of the above-mentioned approaches share the same evaluation scheme, which can be found in either Figure 4.10 or Figure 4.11. The sole difference between the two procedures resides in the training phase: the "Real Data" approach predicts the six-digit HS Code using the models trained with the actual data, whereas the "Predicted Data" approach predicts the code using the models trained with predicted information from previous models.

The evaluation phase intends to assess the model behavior when placed in an unknown environment that tries to mimic the production settings. In order to guarantee this assumption, during this stage, the models were only fed with the SPOKE features, not receiving any information regarding the HS codes, which is the variable the models aim to predict.

A pipeline for the model prediction was created where the first classifier, using the SPOKE features, predicts the chapter. Next, this predicted code is inputted in the subsequent model, which predicts the heading. Finally, both the chapter and the heading codes are inserted in the final classifier to predict the subheading. The output from each of the three classifiers is concatenated, obtaining the six-digit predicted HS code.

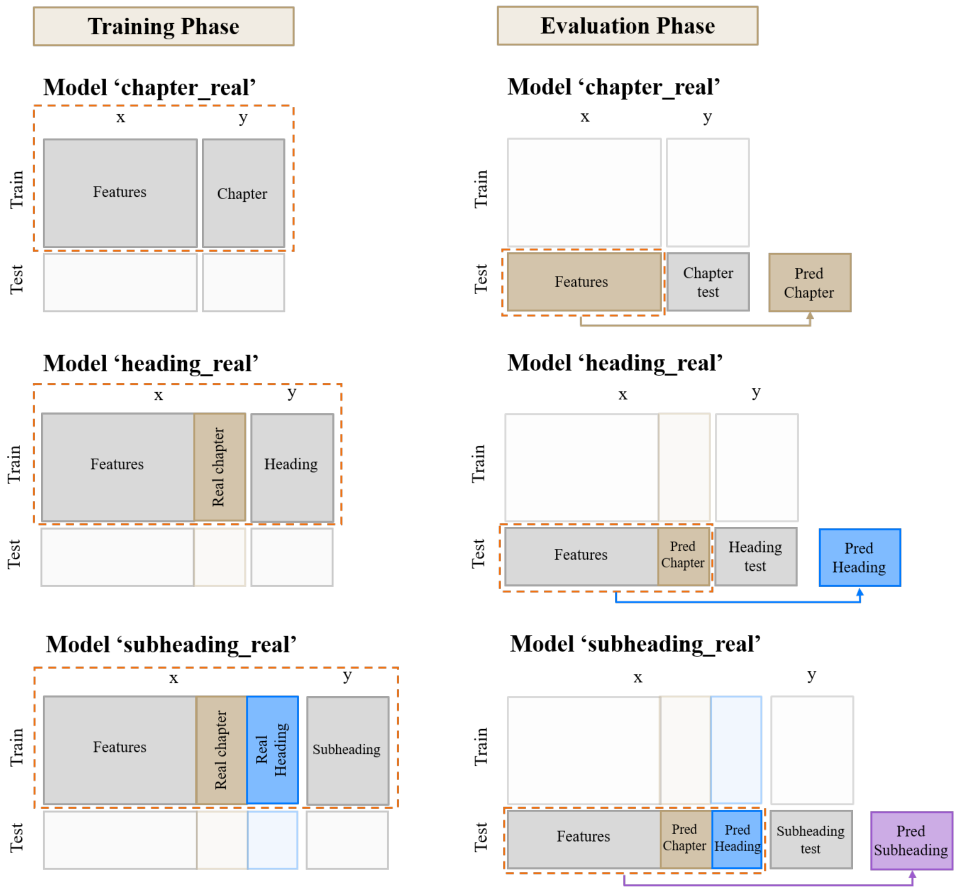


Figure 4.10: Training and evaluation scheme of the "Real Data" approach

4.5.3.1 Evaluation Metrics

Aside from accuracy, the hierarchical precision, recall, and f1-score (Equations 4.1) will be utilized to evaluate the final model's performance.

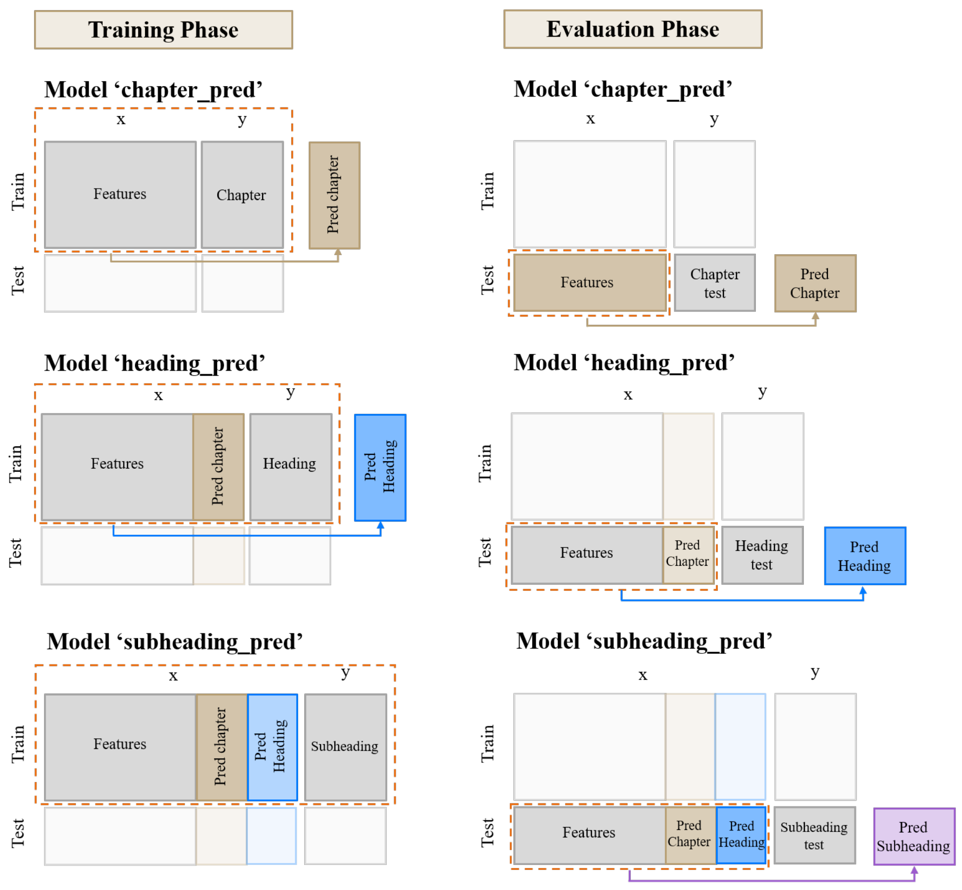


Figure 4.11: Training and evaluation scheme of the "Predicted Data" approach

$$hP = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{P}_i|} \qquad hR = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{T}_i|} \qquad hF = \frac{2 * hP * hR}{hP + hR} \qquad accuracy = \frac{\#correct\ predictions}{\#predictions} \tag{4.1}$$

where
$\hat{P}_i$ is the set consisting of the most specific class(es) predicted for test example i and all its(their) ancestor classes and $\hat{T}_i$ is the set consisting of the true most specific class(es) of test example i and all its(their) ancestor classes.

The hierarchical metrics were chosen so that the hierarchical structure of the problem could be captured during the evaluation phase, which would not happen with flat performance measures.

In order to apply these metrics correctly, it is necessary to understand the concrete specifications of the HS Code problem.

For each six-digit HS Code predicted, once an incorrectly classified two-digit code is obtained, it is presumed that whatever succeeds those numbers is also incorrect. This assumption is made since a two-digit code with the same terminology can mean completely different things, depending on the antecedent classifications. To give an example, assume the actual six-HS Code is "610910", but the model classified it as "620910". Although the model got the heading (09) and subheading (10) correct, it wrongly classified the chapter (62 instead of 61). In this case, the entire code is invalid since the heading "09" belonging to chapter "61" refers to "T-shirts and singlets", whereas the heading "09" from chapter "62" concerns "Baby Garments". Similarly, code "10" from heading "6109" denotes cotton garments, even though subheading "10" does not even exist in the "6209" heading. Concluding, the code is only considered correct until the first inconsistency is found. From that moment on, everything that follows will be inherently incorrect.

The hierarchical precision, recall, and f1-score metrics are calculated per leaf node class, treating the labels (both the predicted and true) as binary. This means each predicted class (each six-HS Code) will have its corresponding precision, recall and f1-score. In Table 4.8, the class "610910" is used to illustrate the logic behind the hierarchical measures. The rationale is repeated for all observations in the dataset and then, the formulas described in 4.1 are applied to calculate the final metrics for the class "610910". The exact process is repeated for each of the remaining 116 classes. The final evaluation measures of the model correspond to an average of these metrics across all leaf node classes.

Table 4.8: Hierarchical measures explained, using test observations of class "610910" as an example, where x1 = $Pi \cap Ti$, x2 = $(Pi \cap Ti)/Pi$ and x3 = $(Pi \cap Ti)/Ti$

obs	Predicted Class	Actual Class	Pi	Ti	x1	x2	x3
1	610910	610910	3[61:1, 09:1, 10:1]	3[61:1, 09:1, 10:1]	3	1	1
2	640910	640910	0[61:0, 09:0, 10:0]	0[61:0, 09:0, 10:0]	0	N/A	N/A
3	420230	610420	0[61:0, 09:0, 10:0]	1[61:1, 09:0, 10:0]	0	N/A	0
4	610590	610910	1[61:1, 09:0, 10:0]	3[61:1, 09:1, 10:1]	1	1	1/3
5	610590	611020	1[61:1, 09:0, 10:0]	1[61:1, 09:0, 10:0]	1	1	1
6	610120	610990	1[61:1, 09:0, 10:0]	2[61:1, 09:1, 10:0]	1	1	1/2
7	610990	610990	2[61:1, 09:1, 10:0]	2[61:1, 09:1, 10:0]	2	1	1
8	610910	610990	3[61:1, 09:1, 10:1]	2[61:1, 09:1, 10:0]	2	2/3	1
9	620411	610910	0[61:0, 09:0, 10:0]	3[61:1, 09:1, 10:1]	0	N/A	0
...	...	...	...	...	...	...	...

Regarding the accuracy, three different levels were evaluated. The accuracy is calculated by concatenating the two-digit combinations predicted by the different classifiers and comparing them with the actual code. In this case, the three computed accuracies were the following: the "two-digit HS Code accuracy" (how well the model predicts the chapter), the "four-digit HS Code accuracy" (how well the model predicts the heading), and the "six-digit HS Code accuracy" (how well the model predicts the subheading, or, in other words, the complete HS Code).

4.5.4 Model Selection

Before the actual training and evaluation phases, it is necessary to choose which model best fits the problem - this phase is called model selection and consists of choosing one final model among many candidates for a predictive modeling problem. Model selection can be used to compare different types of models, referred to as algorithm selection (e.g., Decision Trees, Logistic Regression, Naive Bayes,...), as well as models of the same type with several model hyperparameters, known as hyperparameter optimization. Hyperparameters are the parameters of the learning method itself which have to be specified a priori, i.e., before model fitting. The model selection procedure is always performed on the training set.

For a Local Classifier problem, each classifier may have a different model and specific input features. Since each level tries to predict a different dimension, the corresponding classifiers will have specific requirements that will respond uniquely to each ML model and input features. As a result, for each of the three classifiers (Chapter, Heading, and Subheading), every ML model was tested for every possible set of features, and the most suitable combination was chosen as the final algorithm.

The constitution of each dataset (combination of different features) tested for each one of the three classifiers is shown in Appendix B. For each of the three taxonomic levels, the features containing the indispensable information for the model prediction of the two-digit code were identified. These "must-have" features were kept across all dataset variations. These variations correspond to all the different possible combinations of the remaining features. For instance, the classifier "Chapter" requires the features "type", "sub_family", "fabric", "material" and "age_group". This understanding comes from domain-specific knowledge regarding the rule construction of HS codes. These "mandatory" features are present in all variations of the datasets for the first level. The "gender" feature (the remaining feature), not being absolutely necessary for the prediction, needs to be assessed in terms of model performance. As follows, the first dataset created for the classifier "Chapter" has as input features [type, sub_family, fabric, material, age_group], and the second dataset has the features [product_type, sub_family, fabric, std_material, age_group, **gender**]. Both of these datasets were tested, and the one with the best performance was chosen. The process was repeated for the remaining two hierarchical levels.

Cross Validation and Parameter Tuning

For the model selection phase, the cross-validation technique was chosen over a simple sub-training/validation split. Cross-validation is used to have a more confident estimation of the skill of a Machine Learning model to make predictions on unseen data. This technique is usually preferred over a sub-training/validation split as it allows the model to train in different sub-training sets, giving a more accurate indication of how the model will perform. A sub-training/validation split is dependent on how the data was split for that arrangement, being less representative of the reality.

Both algorithm selection and hyperparameter optimization are important aspects of the model selection stage. Algorithm selection is used to evaluate the performance of a set of Machine Learning algorithms and hyperparameter tuning is used to find the best set of hyperparameters for that ML algorithm. The k-fold cross-validation procedure is used in both of these procedures. However, when the same cross-validation procedure and dataset are used to both tune and select a model, the evaluation of the model's performance is likely to be optimistically biased, returning a poor estimation of errors in the training set due to information leakage (overfitting).

Nested Cross-Validation incorporates both cross-validation and hyperparameter tuning. It is used to assess the performance of a Machine Learning algorithm while estimating the underlying model's generalization error and its hyperparameter search. This process (detailed in Figure 4.12), as it was already mentioned, is exclusively performed on the training set and can be explained in the following way:

Step 1 - Outer CV

The selection of the Machine Learning algorithm is based on its performance on the outer loop of cross-validation. A k equal to 10 outer folds was selected for this context. To split the data into 10-folds such that both the validation and the sub-training sets have an equal number of instances of the target class label, a stratified K fold procedure was implemented. This procedure ensures that one class is not over-represented in the validation or sub-training sets, which can happen when the dataset is imbalanced.

Step 2 - Inner CV

Five inner folds (k'=5) were defined for the internal cross-validation. The inner CV is applied to the 9 folds (k-1), correspondent to the sub-training sets, from the outer CV. The set of parameters are optimized using a random grid search procedure and are then used to configure the model. The best model returned from the grid search is then evaluated using the last fold (validation fold) from the outer CV. This method is repeated k'=5 times, and the final CV score is computed by taking the mean of all k (k=10) scores.

It is common to use k=10 for the outer loop and a smaller k' for the inner loop, such as k'=5. Each iteration of the outer cross-validation procedure reports the estimated performance of the best performing model (using the 5-fold cross-validation) and the hyperparameters found to perform the best, as well as the accuracy on the holdout dataset. A final mean classification accuracy of each of the ten iterations is then reported.

The outcome of the Nested Cross-Validation is the average performance measures of the most suitable model and the set of hyperparameters that perform best for that model.

The test set remained unaffected to ensure the model continued unknown to the data used in a later stage to report the performance of the final model.

As a closing note, it is important to refer that this phase does not intend to fit the final model, as all models are discarded at the end. Instead, after the most suitable model is chosen, a new final model will be fitted on all available training data and subsequently evaluated on the preserved testing set, as described in Sections 4.5.2 and 4.5.3, respectively.

For the Machine Learning algorithms that did not experience any hyperparameter tunning, a simple stratified 10-cross fold validation was employed to assess their performance.

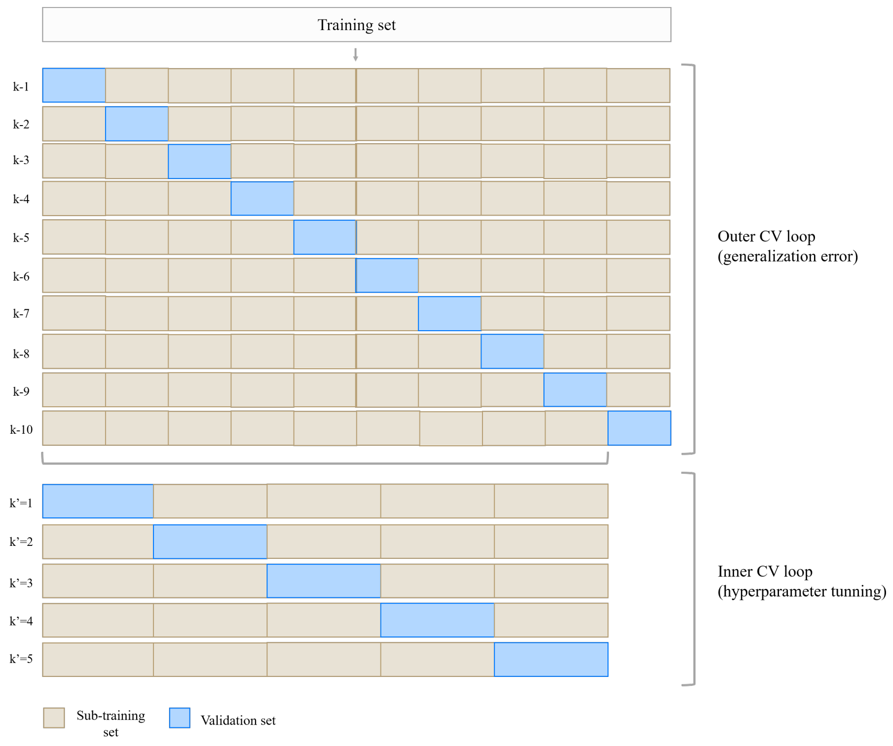


Figure 4.12: Nested Cross Fold Validation applied to the training set

As for the hyperparameters used in each model, Decision Tree algorithms were allowed to vary their maximum depth, the criterion used to split the nodes, the minimum samples in the leaf nodes, and the minimum samples split. For the Random Forest algorithm, the number of trees, the maximum depth, the minimum samples leaf, and the minimum samples split were the hyperparameters varied. The remaining algorithms were not subjected to any hyperparameter tuning. A brief description of these hyperparameters is provided in Table 4.9.

Table 4.9: Hyperparameter description

Hyperparameter	**Description**
Minimum samples leaf	minimum number of samples required to be at a leaf node (external node)
Minimum samples split	minimum number of samples required to split an internal node
Maximum depth	maximum allowed tree size
Number of trees	number of trees built to produce the Random Forest
Criterion	function to measure the quality of a split

For each independent classifier, all the possible ML algorithms dataset combinations were tested. The Machine Learning algorithms tested were Naive Bayes Gaussian, Naive Bayes Bernoulli, Multinomial Logistics Regression, Decision Tree, Random Forest, K-nearest Neighbour (KNN), and Support Vector Machine (SVM). This variety of ML models were chosen since its processing capacity was expected to be reasonable enough to conduct multiple tests.

Finally, to choose the best-suited model for each classifier, different factors were considered, as they are many competing concerns when performing model selection beyond model performance.

The trade-off between model accuracy and model interpretability is important to assess. The more complex the model, the better it can capture the patterns in the data. However, as the model complexity increments, it becomes increasingly difficult to understand all the variations and determine how much weight each factor had on the output value (Figure 4.13). Depending on the primary aim of the analysis, one of the two factors, performance or interpretability, may be privileged.

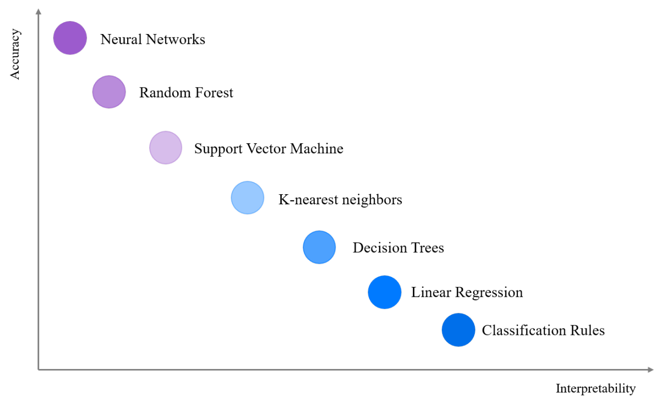


Figure 4.13: Trade-off between interpretability and accuracy of some relevant ML models (adapted from Morocho-Cayamcela et al. (2019))

Bearing this in mind, the choice of the best fitting model in the selection phase was based on the following criteria, in descending order of significance:

1. model performance (flat performance measures)
2. running time
3. model interpretability

Model interpretability was considered the least relevant factor in the decision-making process; its importance, however, cannot be disregarded. For sensitive ethical Machine Learning algorithms, such as the credit-scoring algorithm, model interpretability gains extreme relevance. In such circumstances, it is crucial to be able to explain why a model performs the way it does. This enables data scientists to determine, for example, whether the model is biased due to race, gender, or sexual orientation.

The model interpretability is not critical in the context of this project, but it can be used to explain to the brands why a given HS Code was generated for a specific product. Furthermore, and more importantly, it also enables the assessment of the model's prediction quality. The easier it is to understand the decision undertaken by the algorithm, the easier it will be to detect the systematic classification errors the model might be making.

4.6 Scenario Definition

Different scenarios were created in order to determine the optimal combination of factors that lead to the best-fitted model for the classification of the HS Codes.

The main goal was to compare the two training approaches proposed and validate if the hypothesis of using real input data from previous classifiers to progressively train the succeeding classifiers results in higher final performance. X was chosen to represent the possible training methodologies and can take the value of "Real Data" or "Predicted Data". Within each training approach, it is necessary to determine which factors achieve the best-performing model.

For each training approach, three different training datasets were tested. Y was chosen to represent the different training datasets options, which include "Imbalanced", "Random undersampling of the majority class", and "Random Undersampling followed by Random Oversampling", as defined in Section 4.3.2.

For each training set + training approach combination (x,y), different Machine Learning models were evaluated, varying the number of features used for each one of the three classifiers (Chapter, Heading, and Subheading). The feature selection process was explained in Section 4.5.4 and detailed in Appendix B. Finally, for each classifier included in each (x,y) scenario, the best performing (model+dataset) combination is chosen. Figure 4.14 represents the standard typology applied to all of the different scenarios tested for each (x,y) combination.

In total, 72 scenarios were tested, with the results discussed in the next chapter.

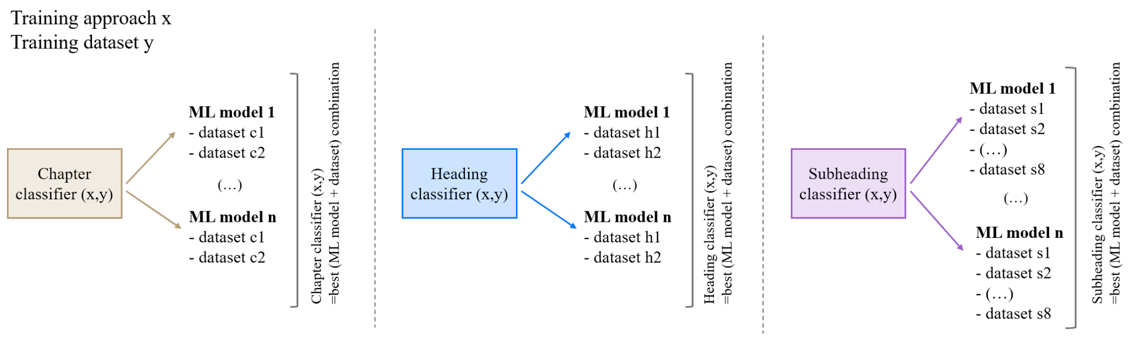


Figure 4.14: Standard typology applied to all of the different scenarios tested for each (x,y) combination

4.7 Programming Tools

The project's rationale required two different programming languages, each with a distinct and specific aim. There was a substantial focus on searching for data in HUUB's database management system, PostgreSQL, to collect and store the appropriate and relevant information to be used in the classification algorithms.

Regarding the models' development, Python 3.9 was the elected programming language. Python is one of the most widely used programming languages in Machine Learning and data science due to its easy syntax, simplicity, and extensive library of ready-to-use packages. The reasoning behind this choice was also to ensure consistency with the rest of HUUB's algorithms.

The code was programmed with the support of several libraries provided in Python that proved fruitful when developing the model. Pandas was the main library employed during the project, used to perform data analysis and manipulation. Pandas is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools, enabling built-in functions that increase the coding efficiency. NumPy was the second most commonly utilized library when handling various N-dimensional array objects, advanced functions, and linear algebra. Both of these libraries were used throughout the entire project.

Concerning the pre-processing steps of the dataset, three libraries were employed: Re, which performs regular expression matching operations; pyspellchecker, which supports spelling corrections in multiple languages; and googletrans, for translation of non-English words.

The scikit-learn package, which provides efficient implementations of a large variety of common algorithms, was used to construct the Machine Learning models. This library was chosen because of its simple, consistent, and streamlined API, as well as its comprehensive online documentation. The imbalanced-learn library was also explored for its range of dataset resampling approaches.

All experiments were computed using an Intel® Core™ i7-7500U CPU @ 2.70GHz 2.90 GHz processor with 8GB of RAM.

Chapter 5

Results and discussion

The results obtained from applying the algorithms and methodology defined in the previous chapters are shown in this chapter where multiple scenarios are evaluated to determine the most suitable model for classifying a product's HS Code.

5.1 Scenario Analysis

All the performance results of the Machine Learning algorithms in all 72 scenarios are shown in Appendix C, as well as the training and predictive computational times needed for these algorithms.

5.1.1 Rule Based Approach

The Rule-base model served as the baseline for the subsequent Machine Learning models. This model registered a remarkable capacity (87,64% of accuracy) in predicting the first two digits of the code (Chapter), but it struggled to classify the complete code (53,42% of accuracy) correctly. The results are shown in Table 5.1.

Table 5.1: Rule-based approach results

Hierarchical Level	**Accuracy**
Chapter (2-digit accuracy)	87.64%
Heading (4-digit accuracy)	74.72%
Subheading (6-digit accuracy)	53.42%

There are several drawbacks to this strategy. First of all, it is not scalable since it requires an update in the model's rules every time a new product is added, belonging to a Chapter, Heading, or Subheading that has not yet been mapped. Aside from being a time-consuming and exhaustive task, it is likely that certain instances will be overlooked due to human error. In addition, some products can be classified into several different categories, and it is unclear which classification is the most appropriate, making it challenging to map the rules.

To conclude, the Rule-based model is an exceedingly high-maintenance model that could never be used in a production setting. The objective of the ML models developed was to outperform this simple approach.

5.1.2 "Real Data" Training Approach

The "Real Data" approach used real output data from the previous classifiers to train the subsequent ones and was employed in three different training dataset distributions.

During the model selection phase, for each independent classifier in each training dataset, the best-performing model was selected. As an example, Figure 5.1 summarizes the ML models' performance for the classifier "Heading" in the "Random Undersampling of the majority class" training set. In this case, the Decision Tree algorithm was chosen (the reasoning behind this decision is further explored in Section 5.2)

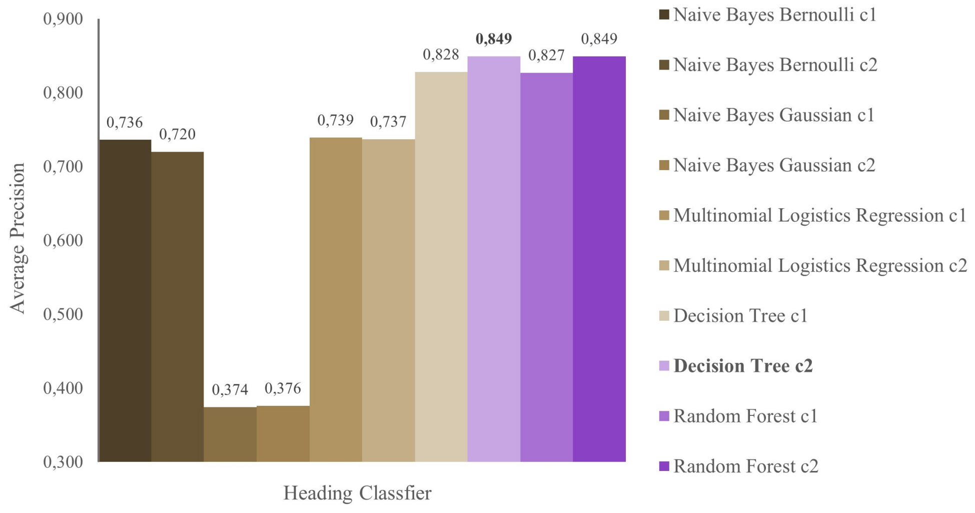


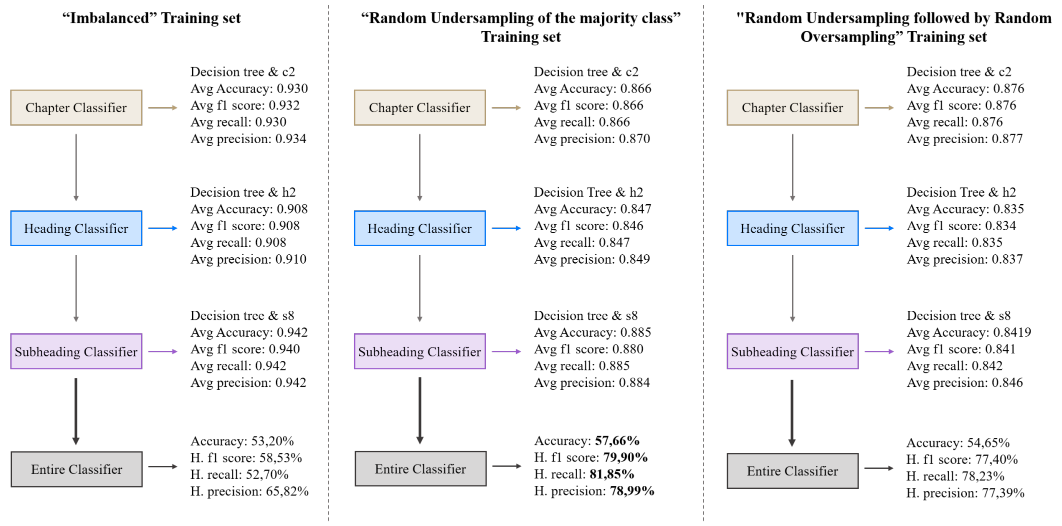
Figure 5.1: Precision results for the different (ML model + dataset) combinations tested for the Heading Classifier of the "Random Undersampling of the majority class" training set

The K-Nearest Neighbours (KNN) and the Support Vector Machine (SVM) algorithms were not tested due to their extraordinarily high computational costs. KNN has a relatively fast training time, but the prediction time is extremely high since it is a lazy algorithm. This occurs due to the lack of generalization capacity of the algorithm; instead, it scans the historical database each time a prediction is needed. SVM, on the other hand, struggles with large data sets and dimensionality. Because the approach is not incremental and requires the complete dataset to be in RAM all at once, it is not computationally efficient.

Regarding the datasets chosen, the algorithms always privileged the ones with the most features.

For the model selection phase, all the Machine Learning algorithms were evaluated using flat performance metrics, and the results can be consulted extensively in Appendix C. The three taxonomic classifiers registered highly encouraging independent performances.

After selecting the ML model for each classifier, the overall model performance was assessed, resorting to hierarchical measures to capture the taxonomic relationships of the HS Code structure. The results are summarized in Table 5.2 and illustrated in Figure 5.2.



"Imbalanced" Training set
Chapter Classifier
Decision tree & c2 Avg Accuracy: 0.930 Avg f1 score: 0.932 Avg recall: 0.930 Avg precision: 0.934
Heading Classifier
Decision tree & h2 Avg Accuracy: 0.908 Avg f1 score: 0.908 Avg recall: 0.908 Avg precision: 0.910
Subheading Classifier
Decision tree & s8 Avg Accuracy: 0.942 Avg f1 score: 0.940 Avg recall: 0.942 Avg precision: 0.942
Entire Classifier
Accuracy: 53,20% H. f1 score: 58,53% H. recall: 52,70% H. precision: 65,82%
"Random Undersampling of the majority class" Training set
Chapter Classifier
Decision tree & c2 Avg Accuracy: 0.866 Avg f1 score: 0.866 Avg recall: 0.866 Avg precision: 0.870
Heading Classifier
Decision Tree & h2 Avg Accuracy: 0.847 Avg f1 score: 0.846 Avg recall: 0.847 Avg precision: 0.849
Subheading Classifier
Decision tree & s8 Avg Accuracy: 0.885 Avg f1 score: 0.880 Avg recall: 0.885 Avg precision: 0.884
Entire Classifier
Accuracy: 57,66% H. f1 score: 79,90% H. recall: 81,85% H. precision: 78,99%
"Random Undersampling followed by Random Oversampling" Training set
Chapter Classifier
Decision tree & c2 Avg Accuracy: 0.876 Avg f1 score: 0.876 Avg recall: 0.876 Avg precision: 0.877
Heading Classifier
Decision Tree & h2 Avg Accuracy: 0.835 Avg f1 score: 0.834 Avg recall: 0.835 Avg precision: 0.837
Subheading Classifier
Decision tree & s8 Avg Accuracy: 0.8419 Avg f1 score: 0.841 Avg recall: 0.842 Avg precision: 0.846
Entire Classifier
Accuracy: 54,65% H. f1 score: 77,40% H. recall: 78,23% H. precision: 77,39%

Figure 5.2: Results of the different scenarios for the "Real Data" approach

The "Random undersampling of the majority class" training set model was the one with the best-fitted performance. It recorded a six-digit accuracy of 58,86%, a four-digit accuracy (heading) of 68,71%, and a two-digit accuracy of 88,50%. Since the most critical information for customs extends to the fourth digit (heading), an accuracy of almost 70% is highly acceptable. The hierarchical metrics registered a hierarchical precision of 79,69%, a hierarchical recall of 82,67%, and an f1-score of 80,82%. The precision measure, which takes into account the performance of minority classes, was given particular attention.

Table 5.2: Performance measures for the "Real Data" training approach, where T1 = "Imbalanced" Training set, T2 = "Random Undersampling of the majority class" Training set and T3 = "Random Undersampling followed by Random Oversampling" Training set

Performance Measures	**T1**	**T2**	**T3**
6-digit accuracy	0,531612	0,576604	0,546507
4-digit accuracy	0,579344	0,667713	0,625442
2-digit accuracy	0,779413	0,879320	0,867999
H. precision	0,658170	0,789857	0,773924
H. recall	0,526950	0,818450	0,782323
H. f1-score	0,585295	0,798956	0,774021

5.1.3 "Predicted Data" Training Approach

The "Predicted Data" approach used the predicted output data from the previous classifiers to train the subsequent ones and was employed in three different training dataset distributions.

Similar to what was done in the previous approach, during the model selection phase, for each independent classifier in each training dataset, the best-performing model was selected. As an example, Figure 5.3 summarizes the ML models' performance for the classifier "Heading" in the "Random Undersampling of the majority class"training set. Once again, the Decision Tree algorithm was chosen. This algorithm was also selected for every classifier, having a similar behavior to the one registered in the "Real data" training approach.

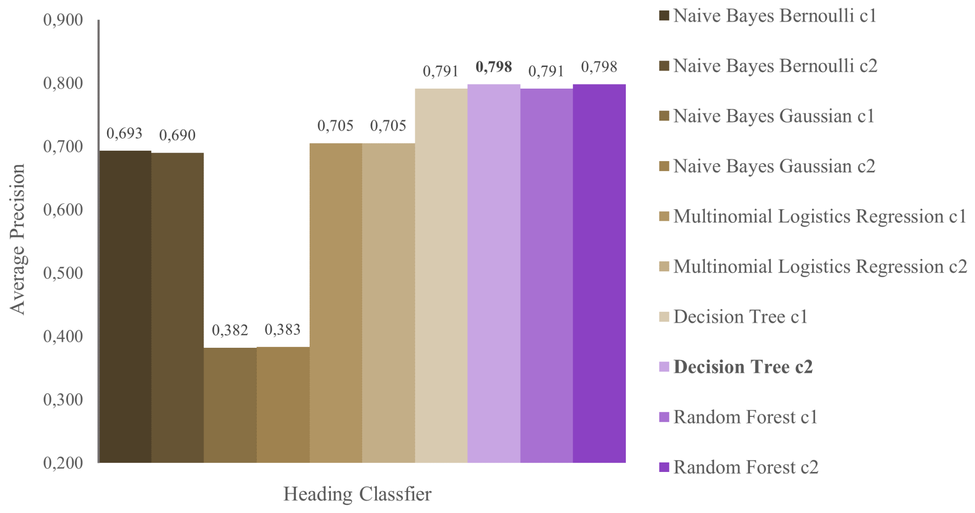


Figure 5.3: Precision results for the different (ML model + dataset) combinations tested for the Heading Classifier of the "Random Undersampling of the majority class" training set

Regarding the datasets chosen, the algorithms always privileged the ones with the most features.

After selecting the ML model for each classifier, the overall model performance was assessed, resorting to hierarchical measures to capture the taxonomic relationships of the HS Code structure. The results are summarized in Table 5.3 and illustrated in Figure 5.4.

The "Random undersampling of the majority class" training set model was, once again, the one with the best-fitted performance. It recorded a six-digit accuracy of 45,23%, a four-digit accuracy (heading) of 65,40%, and a two-digit accuracy of 87,98%. Regarding the hierarchical metrics, it registered a hierarchical precision of 78,43%, a hierarchical recall of 80,68%, and an f1-score of 79,27%.

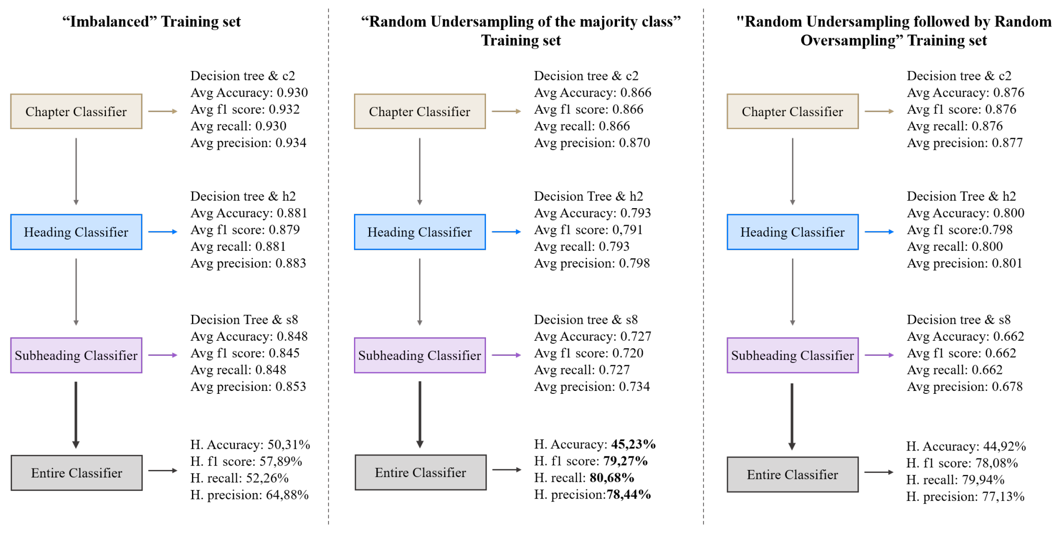


Figure 5.4: Results of the different scenarios for the "Predicted Data" approach

Table 5.3: Performance measures for the "Predicted Data" training approach, where T1 = “Imbalanced” Training set, T2 = “Random Undersampling of the majority class” Training set and T3 = "Random Undersampling followed by Random Oversampling” Training set

Performance Measures	**T1**	**T2**	**T3**
6-digit accuracy	0,503111	0,452260	0,449203
4-digit accuracy	0,578654	0,654034	0,641425
2-digit accuracy	0,779413	0,879846	0,871301
H. precision	0,648775	0,784374	0,771321
H. recall	0,522602	0,806849	0,799411
H. f1-score	0,578893	0,792735	0,780826

5.2 Training Approaches Comparison

Relative to the Rule-based model, the "Real Data" approach exhibited an 10,2% increase in the six-digit accuracy. The "Predicted Data" strategy, on the other hand, returned an 15,33% lower six-digit accuracy when compared to the baseline model. However, considering all of the limitations listed in 5.1.1 for this manual model, as well as the closeness of the results, it is acceptable to infer that both ML approaches are suitable for replacing the Rule-based model in the classification of the HS Codes, producing a more maintainable and scalable model.

In both training approaches, the undersampling training set was the one that produced the best outcomes. The "Imbalanced" training set’s lower results could be owing to the significant data imbalance. In this case, there is a probability that the model was biased towards the majority class, registering good accuracy results. However, it performed poorly when classifying minority classes, resulting in a lower precision. On the other hand, the lower performance of the training set "Undersampling of the majority class followed by oversampling" can be explained by the overfitting of the model during the training phase, resulting in a lack of generalization in the test phase.

The Decision Tree algorithm was the classifier that consistently registered the best performance metrics across all scenarios. Despite being computationally expensive and highly sensitive to changes in the training set, it possesses qualities that make it the ideal candidate for the job. Firstly, it is a very intuitive and interpretable model that closely mimics the human decision-making process. It also performs feature selection automatically, ensuring that irrelevant features have no bearing on the outcome. Quality is also unaffected by the presence of interdependent features (multicollinearity). The Random Forest algorithm also recorded very similar results; however, because of its limited control over how the model performs and higher prediction times, the Decision Tree algorithm was repeatedly privileged, being the selected algorithm for every single classifier.

Comparing the two training approaches, the "Real data" approach stood out, with an increase of 30,13% in the final accuracy. Hierarchical precision, recall, and f1-score were also higher (0,70%, 1,44%, and 0,79% increases, respectively). This outcome can be explained with the help of the diagram in Figure 5.5.

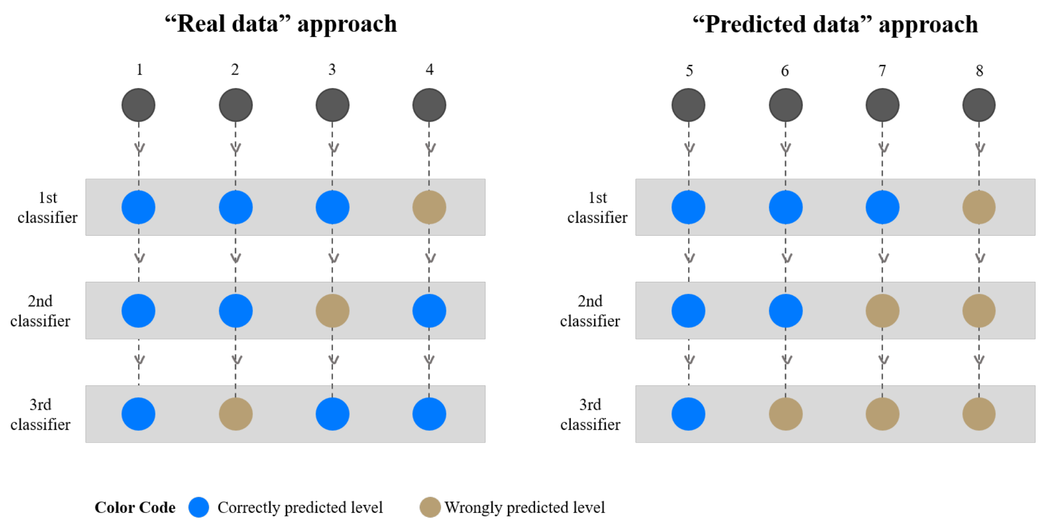


Figure 5.5: Diagram exemplifying the training phase of the two tested approaches

The "Real Data" training approach trains the model with real data, which prevents the propagation of errors from one classifier to another during the model's learning phase. The example "3" and "4" of Figure 5.5 illustrate a scenario where the "Real data" approach gains relevance. During the training phase, even if the previous classifier got a wrong prediction, because the subsequent classifier learns with real input data and is entirely independent of the previous one, there is still a possibility for the subsequent classifier to get the prediction correct. This approach prevents compromising the learning of future classifiers because of prediction errors made beforehand.

The same phenomenon is not verified in the "Predicted Data" approach. Once a beige dot is detected (wrong level prediction), all the further classifiers, which use that wrong input data, will most likely be wrong as well. The errors made by the previous classifier are propagated to the following one and the model learns a inaccurate relationship between the various variables.

The winning approach presented a two-digit accuracy of 87,93%, a four-digit accuracy of 66,77%, and a six-digit accuracy of 57,66%. These results are considered good enough since, from a business perspective, the first level (first two digits) posses the most significant influence on customs. The latter level (last two digits), which corresponds to the six-digit accuracy, has minimal influence on customs compliance since it only relates to the product's material.

Chapter 6

Conclusions and Future Work

The foremost purpose of the work presented in this dissertation, conducted in the Product Team, was to develop a solution that could enable HUUB to classify the HS Codes of fashion products, resorting to Machine Learning models. As the online consumption increases each year, with the Covid-19 pandemic working as an accelerator factor, fashion brands are becoming progressively more exigent with the service level HUUB's is offering them. Delivered Duty Paid (DDP) is becoming a frequently demanded service that HUUB does not currently provide.

At the beginning of the project, HUUB only offered DDP to the B2B segment and its current operation lacked the scale to adapt to the B2C order volume. The first step of the project was to understand how the DDP process worked, what were the risks involved, and what did brands valued the most about it. During this discovery process, the HS Code was identified as a critical element in the project's risk assessment. This is justified because the information contained in its six digits is used by the customs to determine how much taxes and duties each package will be levied. The main problem was that this product classification was beyond HUUB's control, posing a greater risk to the company's operations. Hence, this project intended to build a model that leveraged the products' information stored in HUUB's database to generate the six-digit HS code, given HUUB full ownership over the DDP process. The deployment of the model would constitute the first step in the Delivered Duty Paid (DDP) service's implementation.

The chosen methodology consisted of the complete development of various Machine Learning models, using different training approaches to select the best-fitted model for the classification of the HS Codes.

The project started by framing the problem at hand, namely thoroughly comprehending the HS Code hierarchical structure. After having a clear understanding of the problem, the data required for the classification task was identified and gathered. A more in-depth analysis of the features contained in the dataset was carried out, which revealed the presence of open fields that lacked standardization. Hence, the first part of the project was spent preprocessing the data.

A Rule-based approach was developed, intended to mimic the traditional and manual hierarchical classification structure. This approach served solely as a baseline model for the comparison with posterior ML models.

The analysis of the dataset revealed a natural asymmetry in the data distribution regarding the predictive variable. The majority class "610910" accounted for more than 50% of the data volume, which originated an extremely skewed data distribution. In addition, this imbalanced distribution was not representative of HUUB's current reality. Resampling techniques were applied to balance the data, namely a random undersampling approach and a random undersampling followed by oversampling approach. The three dataset distributions (including the original distribution) were tested.

Once the data preprocessing phase finished, the following step was to develop a classification model that could capture the taxonomy of the HS Code problem. To that extent, a Hierarchical Classification model was built, using a Tree Local Classifier per Level (LCL) structure. This structure involved three independent classifiers, one for each hierarchical level (Chapter, Heading, and Subheading).

For the model training, two different training approaches were designed. The first approach ("Real data" approach) entailed using real output data from the previous classifier to train the subsequent one. The prospect of avoiding error propagation was the propulsor behind its development. The second approach ("Predicted Data" approach) followed a traditional method, applying the predicted data from the previous classifier as the input data for the subsequent one.

Different scenarios were designed, considering the variations in the training approaches and the dataset distributions, along with the experiment of multiple Machine Learning models. One of the model's advantages is its ability to integrate with any machine learning model due to the generic structure built, not requiring adaptions or adjustments depending on the algorithm employed. This project focused on using the Naive Bayes, the Multinomial Logistic Regression, the Decision Tree, and the Random Forest algorithms.

These scenarios were then compared, ending up with a choice over the model to be implemented. The choice was made taking into account not only the model's accuracy in the prediction task, but also other performance metrics such as the hierarchical precision, recall, and f1-score, as well as the time needed to train and provide a prediction. The experiments were conducted using unseen observations to mimic the actual production environment. In the end, the "Real data" approach with the undersampling training set distribution using only Decision Trees classifiers was selected to be implemented as it provided the best performance results.

This approach presented a two-digit accuracy of 87,93%, a four-digit accuracy of 66,77%, and a six-digit accuracy of 57,66%. These results are considered good enough since, from a business perspective, the first level (first two digits) posses the most significant influence on customs. The latter level (last two digits), which corresponds to the six-digit accuracy, has minimal influence on customs compliance since it only relates to the product's material.

One of the project's primary goals was to demonstrate that Machine Learning models may be a viable solution for the HS Code categorization task, which is a very manual procedure. This was accomplished as the best ML model developed registered a 10,2% accuracy increase compared to the baseline model. Another intent was to prove if the "Real Data" approach could be a feasible solution, considering the novel methodology. The results showed that the best model of the "Real

Data" approach had 30,13% better accuracy when compared to the "Predicted Data" approach, resultant from the diminishing in the error propagation between classifiers. These results validated the "Real Data" training approach.

The presented methodology brought up multiple advantages. Firstly, this project allowed HUUB to gain a deeper understanding of the data used in this process.The data cleaning and preprocessing steps were automated, removing the need to manually transform data to a suitable format when classifying new products. In addition, this project focused a great amount of time processing the open field "material", allowing its use in further models developed by HUUB.

Secondly, and given that the HS Codes contain codified information about the products' attributes, the output of this classification model can be utilized as input to new future models that require information contained in its six digits.

Finally, the main contribution of this project was the development of a novel HS Code methodology that will allow HUUB to gain ownership over a service that it is currently being outsourced. Moreover, by deploying the model, HUUB will be one step closer to implementing the DDP service.

6.1 Future Work and Improvement Opportunities

The following natural step is to deploy the classification model. In order to do so, the microservice "Product Service", which handles the products' creation in SPOKE, is connected to the "HS Classifier" microservice, which manages the classification of the products' HS codes. This linkage is established through KAFKA, a service aggregator platform and portrayed in Figure 6.1. Once a product is created, its information is communicated to the "HS Classifier" microservice, where the prediction for its six-digit code is made and subsequently communicated back to the "Product Service" microservice to update the product information. This communication is done asynchronously and independently from the rest of the processes, ensuring that the standard flow of product creation is not disrupted. After this linkage is established, a product will automatically be assigned with a new or validated HS Code when it is created on SPOKE.

Despite the promising results, the model is naturally subject to improvement in order for this project to achieve its full potential.

The Boosting Machine Learning algorithms, which were not considered in the scope of this dissertation, could be further explored. The boosting algorithms seek to improve the prediction power by training a sequence of weak models, each compensating for the flaws of its predecessors and have the potential to provide more accurate results.

A "blocking by confidence" strategy could also be used to enhance the model's reliability. A blocking technique could be a mechanism for avoiding the proliferation of misclassifications, where an example is transmitted down to the next level only if the confidence in the current level's prediction is greater than a threshold. However, this strategy has the cost of presenting the user

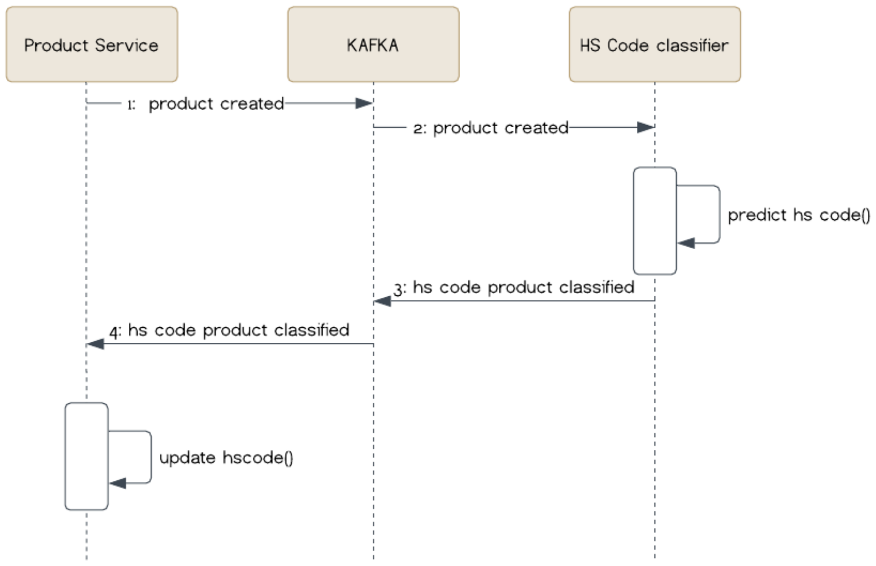


Figure 6.1: Deployment of the "HS Code Classifier" microservice

with less detailed HS codes. The endorsement of this technique requires an in-depth analysis on the impact of a more generic code on customs clearance, as well as research on a proper threshold to preserve a trade-off between the reliability and the specialization capacity of the model. This strategy would result in various adjustments to the TO-BE operation's design (Figure 3.5).

The limited representation of the range of HS Codes is one of the main disadvantages of the proposed work. The model developed only covered 117 different HS codes, which corresponds to approximately 30% of the total range of HS Codes of fashion products and accessories and 2% of the total number of possible HS Codes. The model will be unfamiliar with any new product outside the scope of the 117 different HS Codes. In addition, the removal of less representative classes, justified in Section 4.3.1, may decrease the robustness of the model and its future predictive ability, having difficulties in the generalization of the new data. For example, if HUUB were to enter new business model segments (e.g., start exploring sports gear wear) or acquire a new fashion brand with a different product typology, there is a high probability that the model would not be able to predict the HS Codes correctly. To overcome this, a data collection pipeline must be created in order to constantly provide the model with unseen product typology. Notwithstanding, the project's main objective was to prove the concept of the HS Code Classification with Machine Learning models, considering its limitations.

The methodology described in this dissertation provides a solid foundation for HUUB to begin offering an HS Code classification service to its brands. The model, however, must adapt to the company's ever-evolving business vision.

One of the planned developments that could impact the usability of this model is the prospect of HUUB not imposing a product categorization structure on its brands.

Product categorization (or product hierarchy) is the management of products by grouping the items with related characteristics and attributes into categories and subcategories. Currently, HUUB requires brands to adapt to its internal product categorization structure: as described in the AS-IS situation in Section 3.1, when a new product is created in SPOKE, the brand must classify it into multiple categories (e.g., product type, product family, and product subfamily), each with a limited number of alternatives established by HUUB. The new company's vision will allow brands to have much more freedom and customization power, as it will enable them to create their products according to their own internal structure, without any interference from HUUB. Although this solution enhances the brands' value proposition, it requires internal developments and adjustments.

Because it was built using these internal categories, the model developed in the dissertation is an excellent illustration of an element that will need to be adapted. Other models currently being used by HUUB will also need to be modified. One viable solution focuses on the creation of a translation module placed before the models that employs middleware layers to translate each of the brand categories. This module could resort to Text Mining and Natural Langue Processing (NLP) techniques to interpret the words before classifying them into one of HUUB's internal categories. An interesting python library to explore for this purpose is spaCy.

An even more far-reaching approach would be for no brand to be obliged to categorize its products, merely providing a short text description with detailed information about the product. In the future, it would also be interesting to explore the application of a Computer Vision model that uses images of the products to retrieve its characteristics instead of resorting to textual descriptions.

These potential new developments need to be further discussed and explored in order to be aligned and sustain the future growth of HUUB, contributing to continuously improving its value proposition.

Bibliography

Altaheri, F. and Shaalan, K. (2020). *Exploring machine learning models to predict harmonized system code*, volume 381 LNBIP. Springer International Publishing.

Amor, N. B., Benferhat, S., and Elouedi, Z. (2006). Qualitative classification with possibilistic decision trees. In In Bouchon-Meunier, B., Coletti, G., and Yager, R. R., editors, *Modern Information Processing*, pages 159–169. Elsevier Science, Amsterdam.

Ariyaratne, H. B. and Zhang, D. (2012). A novel automatic hierachical approach to music genre classification. *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2012*, pages 564–569.

Bennett, P. N. and Nguyen, N. (2009). Refined experts: Improving classification in large taxonomies. *Proceedings - 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009*, pages 11–18.

Bergami, R. (2016). International Delivery Risks: The Case of Delivered Duty Paid in Australia. *Acta Universitatis Bohemiae Meridionalis*, 19(1):1–9.

Binder, A., Kawanabe, M., and Brefeld, U. (2010). Efficient classification of images with taxonomies. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5996 LNCS, pages 351–362.

Binh, N. T., Nguyen, H. A., Linh, P. N., Giang, N. L., and Thang, T. N. (2021). Attentive RNN for HS Code Hierarchy Classification on Vietnamese Goods Declaration. pages 298–304. Springer, Singapore.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer, Singapore, Cambridge.

Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., and Struyf, J. (2002). Hierarchical multi-classification. In: Proceedings of the First SIGKDD Workshop on MultiRelational Data Mining. pages 21–35.

Brecheisen, S., Kriegel, H.-P., Kunath, P., and Pryakhin, A. (2006). HIERARCHICAL GENRE CLASSIFICATION FOR LARGE MUSIC COLLECTIONS. In *In: Proc. of the IEEE 7th Int. Conf. on Multimedia Expo*, pages 1385–1388. In: Proc. of the IEEE 7th Int. Conf. on Multimedia Expo.

Cerri, R., Barros, R. C., de Carvalho, A. C., and Jin, Y. (2016). Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics*, 17(1):1–24.

Chakrabarti, S., Dom, B., Agrawal, R., and Raghavan, P. (1998). Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal*, 7(3):163–178.

Che, J., Xing, Y., and Zhang, L. (2018). A comprehensive solution for deep-learning based cargo inspection to discriminate goods in containers. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1287–1294.

Chen, C. (2020). Which Retailers Have Been Hit Hardest by the Covid-19 Crash? *Business of Fashion*.

Clare, A. and King, R. D. (2003). Predicting gene function in Saccharomyces cerevisiae. In *Bioinformatics*, volume 19. Bioinformatics.

Davis, J. and Vogt, J. (2021). Incoterms® 2020 and the missed opportunities for the next version. *International Journal of Logistics Research and Applications*, pages 1–24.

Decoro, C., Barutcuoglu, Z., and Fiebrink, R. (2007). BAYESIAN AGGREGATION FOR HIERARCHICAL GENRE CLASSIFICATION. Technical report, In: Proc. of the 8th Int. Conf. on Music Information Retrieval, Vienna, Austria.

Dimitrovski, I., Kocev, D., Loskovska, S., and Džeroski, S. (2008). Hierarchical Annotation of Medical Images. In *Proc. of the 11th Int. Multiconference Information Society*, pages 174–177.

Ding, L., Fan, Z. Z., and Chen, D. L. (2015). Auto-categorization of HS code using background net approach. *Procedia Computer Science*, 60(1):1462–1471.

Du, S., Wu, Z., Wan, H., and Lin, Y. (2021). HScodeNet: Combining Hierarchical Sequential and Global Spatial Information of Text for Commodity HS Code Classification. pages 676–689. Springer, Cham.

Duarte, P., Costa e Silva, S., and Ferreira, M. B. (2018). How convenient is it? Delivering online shopping convenience to enhance customer satisfaction and encourage e-WOM. *Journal of Retailing and Consumer Services*, 44(May):161–169.

Durdağ, C. and Gül, Delipinar, E. (2021). The past, today and future of incoterms in international delivery: A review on the innovations in logistics. Technical Report 4.

Eisner, R., Poulin, B., Szafron, D., Lu, P., and Greiner, R. (2005). Improving protein function prediction using the hierarchical structure of the gene ontology. *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology, CIBCB '05*, 2005(December).

Gauch, S., Chandramouli, A., and Ranganathan, S. (2009). Training a hierarchical classifier using inter document relationships. *Journal of the American Society for Information Science and Technology*, 60(1):47–58.

Gessner, G. H. and Snodgrass, C. R. (2015). Designing e-commerce cross-border distribution networks for small and medium-size enterprises incorporating Canadian and U.S. trade incentive programs. *Research in Transportation Business and Management*, 16:84–94.

Harsani, P., Suhendra, A., Wulandari, L., and Wibowo, W. C. (2020). A study using machine learning with Ngram model in harmonized system classification. *Prihastuti Harsani, Adang Suhendra, Lily Wulandari, Wahyu Catur Wibowo*, 12(6 Special Issue):145–153.

Hastie, T., Tibshirani, R., and Friedman, J. (2017). *The elements of Statistical Learning*.

Heijde, J. (2019). *Automated classification tool for e-commerce products and their HS code*. PhD thesis, Eindhoven University of Technology.

Holden, N. and Freitas, A. (2008). Improving the performance of hierarchical classification with swarm intelligence | Proceedings of the 6th European conference on Evolutionary computation, machine learning and data mining in bioinformatics. *Springer*, 4973(Lecture Notes in Computer Science):48–60.

Jabbar, M. A., Deekshatulu, B., and Chandra, P. (2013). Classification of Heart Disease Using K-Nearest Neighbor and Genetic Algorithm. *Procedia Technology*, 10:85–94.

Jung, A. (2020). Machine Learning: Basic Principles. Technical report, Aalto University, Finnland.

Kansara, V. A. (2020). Could the Pandemic Make Retail Better? | BoF Professional, This Week in Fashion | BoF.

Kappler, H. (2011). Reversing the trend: Low cost and low risk methods for assuring proper duty payments. *World Customs Journal*, 5(2):109–122.

Kim, R. Y. (2020). The Impact of COVID-19 on Consumers: Preparing for Digital Sales. *IEEE Engineering Management Review*, 48(3):212–218.

Kingsford, C. and Salzberg, S. L. (2008). What are decision trees? Technical report.

Kiritchenko, S., Matwin, S., and Famili, A. F. (2005). Functional annotation of genes using hierarchical text categorization. *Proceedings of the ACL workshop on linking biological literature, ontologies and databases: mining biological semantics*.

Krishnapuram, B., Carin, L., Figueiredo, M. A., and Hartemink, A. J. (2005). Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6):957–968.

Kuhkan, M. (2016). A Method to Improve the Accuracy of K-Nearest Neighbor Algorithm. *International Journal of Computer Engineering and Information Technology*, 8(6):90–95.

Li, G. and Li, N. (2019). Customs classification for cross-border e-commerce based on text-image adaptive convolutional neural network. *Electronic Commerce Research*, 19(4):779–800.

Luppes, J., Ikonomakis, M., Kotsiantis, S. B., Tampakas, V., , , Sebastian, D., Nugraha, K. A., Yovellia Londo, G. L., Kartawijaya, D. H., Ivariyani, H. T., Yohanes Sigit, P. W., Muhammad Rafi, A. P., Ariyandi, D., Rodrı́guez, P., Bautista, M. A., Gonzàlez, J., Escalera, S., Putra, I. F., Purwarianti, A., C, P. P. D. M. S., Kotsiantis, S. B., Zaharakis, I. D., Pintelas, P. E., Kapil, A. R., Edizel, B., Piktus, A., Bojanowski, P., Ferreira, R., Grave, E., Silvestri, F., Wongso, R., Luwinda, F. A., Trisnajaya, B. C., Rusli, O., Rudy, Potdar, K., S., T., D., C., Altaheri, F., Shaalan, K., Liu, C., Wang, X., Berrahou, S. L., Buche, P., Juliette, D. B., Roche, M., Utomo, M. R. A., and Sibaroni, Y. (2019). Classifying Short Text for the Harmonized System with Convolutional Neural Networks. *arXiv*, 26(August 2014):291–303.

Meyer, D. (2020). Support Vector Machines. Technical report, FH Technikum Wien, Austria.

Morocho-Cayamcela, M. E., Lee, H., and Lim, W. (2019). Machine learning for 5G/B5G mobile and wireless communications: Potential, limitations, and future directions. *IEEE Access*, 7:137184–137206.

Murphy, K. P. (2012). *Machine learning - A probabilistic Perspective*. The MIT Press, Cambridge, Massachusetts.

Office of the Auditor General of Canada (2010). 2010 Fall Report of the Auditor General of Canada. `https://www.oag-bvg.gc.ca/internet/English/parl_oag_201010_08_e_34291.html`.

Otero, F. E., Freitas, A. A., and Johnson, C. G. (2010). A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing*, 2(3):165–181.

Qiu, X., Gao, W., and Huang, X. (2009). Hierarchical multi-class text categorization with global margin maximization. *ACL-IJCNLP 2009 - Joint Conf. of the 47th Annual Meeting of the Association for Computational Linguistics and 4th Int. Joint Conf. on Natural Language Processing of the AFNLP, Proceedings of the Conf.*, (June 2014):165–168.

Reuters (2020). Brands See an Uptick in Online Sales During the Covid-19 Crisis. *Business of Fashion*.

Rita, P., Oliveira, T., and Farisa, A. (2019). The impact of e-service quality and customer satisfaction on customer behavior in online shopping. *Heliyon*, 5(10).

Rohm, A. J. and Swaminathan, V. (2004). A typology of online shoppers based on shopping motivations. *Journal of Business Research*, 57(7):748–757.

Ryzhova, A. and Sochenkov, I. (2019). Deep learning for customs classification of goods based on their textual descriptions analysis. *Proceedings - 2019 Ivannikov Ispras Open Conference, ISPRAS 2019*, pages 55–59.

Sabanoglu, T. (2021). • Global retail e-commerce market size 2014-2023.

Secker, A., Davies, M. N., Freitas, A. A., Clark, E. B., Timmis, J., and Flower, D. R. (2010). Hierarchical classification of G-Protein-Coupled Receptors with data-driven selection of attributes and classifiers. *International Journal of Data Mining and Bioinformatics*, 4(2):191–210.

Secker, A. D., Davies, M. N., Freitas, A. A., Timmis, J., Mendao, M., and Flower, D. R. (2007). An experimental comparison of classification algorithms for hierarchical prediction of protein function. *Expert Update (Magazine of the British Computer Society's Specialist Group on AI)*, 9(3):17–22.

Sheth, J. (2020). Impact of Covid-19 on consumer behavior: Will the old habits return or die? *Journal of Business Research*, 117(January):280–283.

Silla, C. N. and Freitas, A. A. (2009). A Survey of Hierarchical Classification Across Different Application Domains. *Intelligent Systems Reference Library*, 1(1):623–637.

Song, Y. Y. and Lu, Y. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2):130–135.

Stein, R. A., Jaques, P. A., and Valiati, J. F. (2019). An analysis of hierarchical text classification using word embeddings. *Information Sciences*, 471:216–232.

Vajjala, S., Majumder, B., Gupta, A., and Surana, H. (2020). *Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems*.

Vieira, S., Lopez Pinaya, W. H., and Mechelli, A. (2019). *Introduction to machine learning*.

Wang, Y. (2005). A multinomial logistic regression modeling approach for anomaly intrusion detection. *Computers Security*, 24:662–674.

Watanabe, T. and Omori, Y. (2020). Online Consumption During the COVID - 19 Crisis : Evidence from Japan. *JSPS Grant-in-Aid for Scientific Research (S)*, 023(023):39.

Webb, G. I. (2016). Encyclopedia of Machine Learning and Data Mining. *Encyclopedia of Machine Learning and Data Mining*, (January 2016).

Weerth, C. (2008). Basic Principles of Customs Classifications under the Harmonized System. *Global Trade and Customs Journal*, 3(2):61–67.

Wehrmann, J., Cerri, R., and Barros, R. C. (2018). Wehrmann18a.Pdf. *Proceedings of the 35th International Conference on Machine Learning*, 80:5075–5084.

World Customs Organization (2013). What is the Harmonized System (HS)? `http://www.wcoomd.org/en/topics/nomenclature/overview/what-is-the-harmonized-system.aspx`. Accessed on 3 march 2021.

Xiao, Z., Dellandrea, E., Dou, W., and Chen, L. (2008). Automatic Hierarchical Classification of Emotional Speech. pages 291–296. Institute of Electrical and Electronics Engineers (IEEE).

Yang, X. S. (2019). *Introduction to algorithms for data mining and machine learning*. Elsevier.

Youn, S. Y., Lee, J. E., and Ha-Brookshire, J. (2021). Fashion Consumers' Channel Switching Behavior During the COVID-19: Protection Motivation Theory in the Extended Planned Behavior Framework. *Clothing and Textiles Research Journal*.

Yu, D. Z., Cheong, T., and Sun, D. (2017). Impact of supply chain power and drop-shipping on a manufacturer's optimal distribution channel strategy. *European Journal of Operational Research*, 259(2):554–563.

Appendix A

Product Discovery Analysis

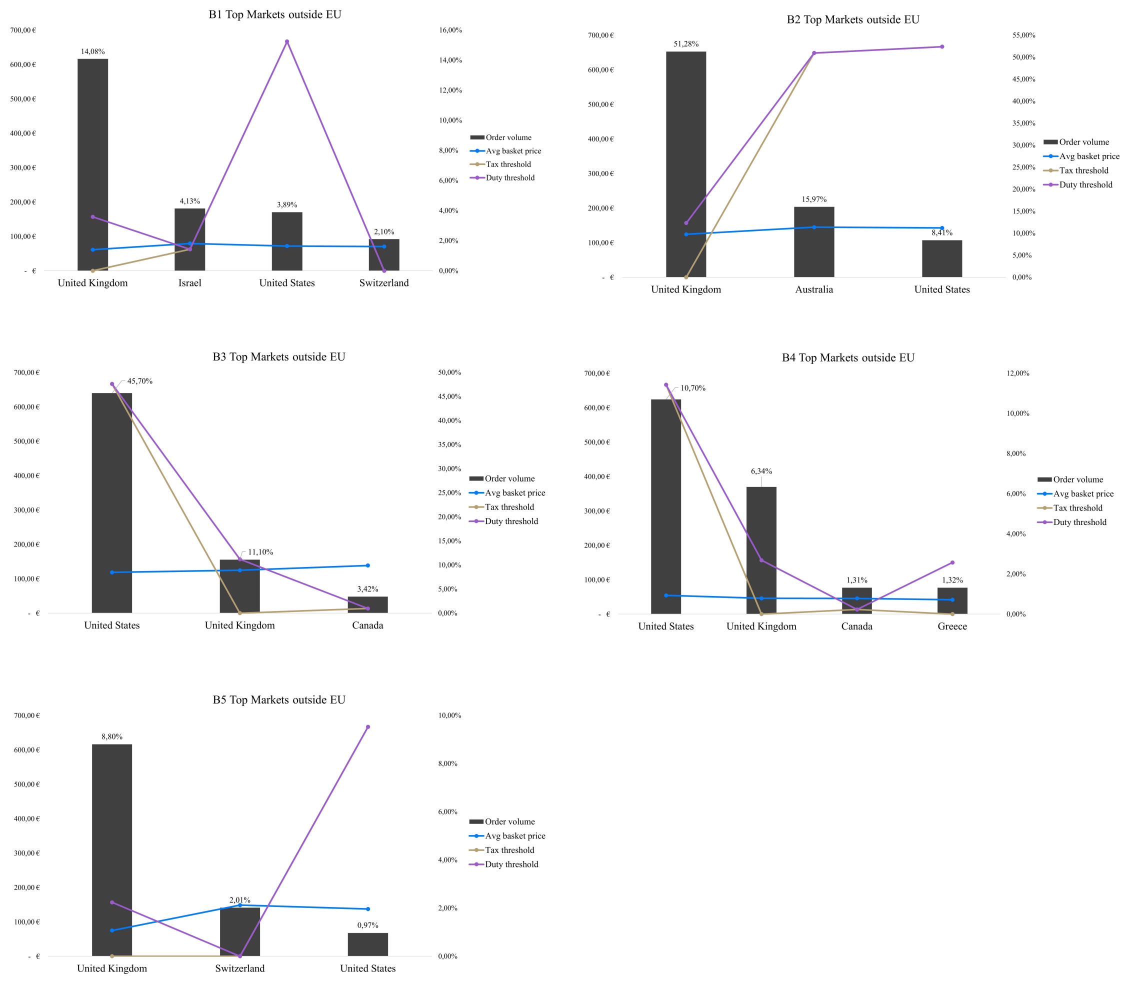


Figure A.1: Percentage of orders per main market, tax and duty threshold and average basket price for HUUB's brands

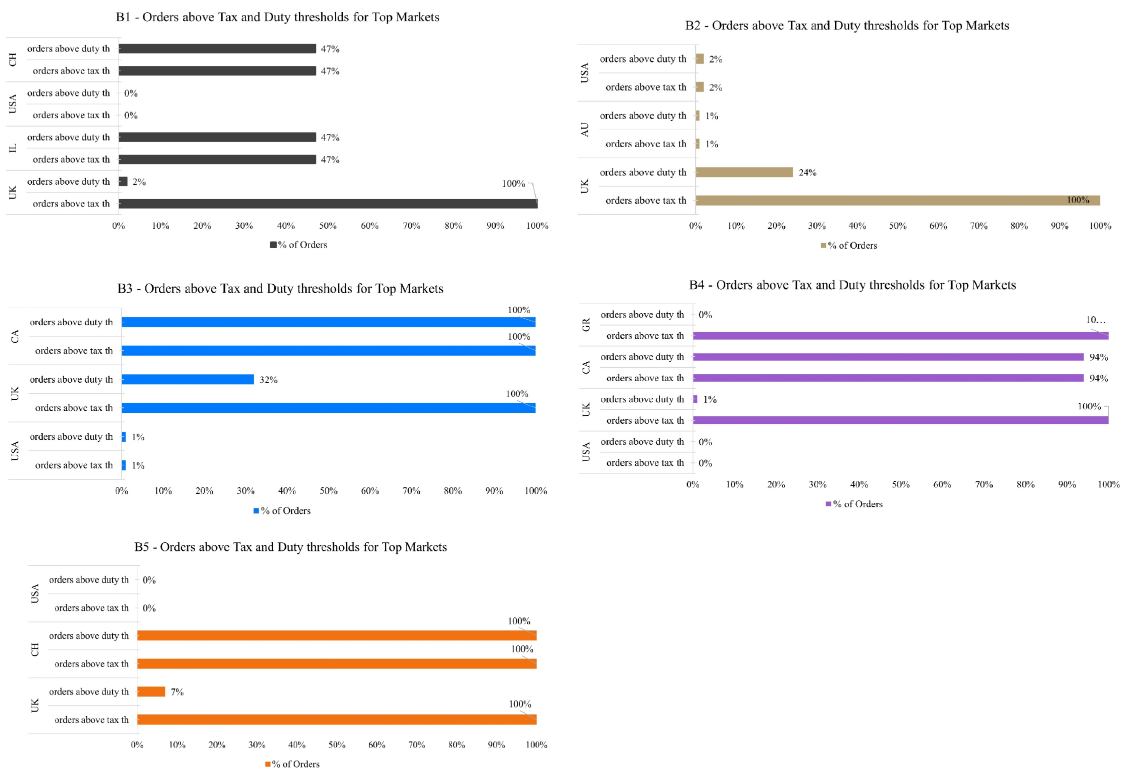


Figure A.2: Percentage of orders above Tax and Duty thresholds for the Top Markets of HUUB's five brands

Appendix B

Dataset constitution

Table B.1: Constitution of every dataset (combination of different features) tested for each one of the three classifiers

CHAPTER CLASSIFIER		
Dataset	**Features**	**Dep. Variable**
c1	type, sub_family, fabric, material, age_group	hs_chapter
c2	type, sub_family, fabric, material, age_group, gender	hs_chapter
HEADING CLASSIFIER		
Dataset	**Features**	**Dep. Variable**
h1	type, sub_family, age_group, gender, hs_chapter, material	hs_heading
h2	type, sub_family, age_group, gender, hs_chapter, material, fabric	hs_heading
SUBHEADING CLASSIFIER		
Dataset	**Features**	**Dep. Variable**
s1	sub_family, gender, material, hs_chapter, hs_heading	hs_subheading
s2	sub_family, gender, material, fabric, hs_chapter, hs_heading	hs_subheading
s3	sub_family, gender, material, age_group, hs_chapter, hs_heading	hs_subheading
s4	sub_family, gender, material, type, hs_chapter, hs_heading	hs_subheading
s5	sub_family, gender, material, fabric, age_group, hs_chapter, hs_heading	hs_subheading
s6	sub_family, gender, material, fabric, type, hs_chapter, hs_heading	hs_subheading
s7	sub_family, gender, material, type, age_group, hs_chapter, hs_heading	hs_subheading
s8	sub_family, gender, material, fabric, age_group, type, hs_chapter, hs_heading	hs_subheading

Appendix C

Results

Table C.1: Model Selection Results for the Chapter Classifier, using the "Real Data" Approach and the "Imbalanced" Training set

CHAPTER CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
c1	0.883419	0.889363	0.883419	0.909006	0.9122s	0.2649s
c2	0.862598	0.870893	0.862598	0.900368	0.8780s	0.2495s
Naive Bayes Gaussian						
c1	0.851168	0.840157	0.851168	0.841812	0.6437s	0.3383s
c2	0.852576	0.842489	0.852576	0.843607	0.5863s	0.3065s
Multinomial Logistics Regression						
c1	0.923897	0.925116	0.923897	0.927423	10.8523s	0.4411s
c2	0.923143	0.924525	0.923143	0.927282	9.8860s	0.3388s
Decision Tree						
c1	0.928640	0.929641	0.928640	0.931474	148.1396s	0.6652
c2	**0.930498**	**0.931690**	**0.930498**	**0.934124**	**138.0479s**	**0.6989s**
Random Forest						
c1	0.928414	0.929427	0.928414	0.931281	184.8735s	1.1996s
c2	0.930335	0.931531	0.930335	0.933975	86.0617s	0.6228s

Table C.2: Model Selection Results for the Heading Classifier, using the "Real Data" Approach and the "Imbalanced" Training set

HEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
h1	0.819189	0.824552	0.819189	0.842449	1.0607s	0.2876s
h2	0.801462	0.812267	0.801462	0.839235	2.0872s	0.5396s
Naive Bayes Gaussian						
h1	0.519348	0.547700	0.519348	0.818577	0.7690s	0.5419s
h2	0.520188	0.549197	0.520188	0.821053	1.5065s	1.1801s
Multinomial Logistics Regression						
h1	0.868950	0.867007	0.868950	0.868916	17.8238s	0.3592s
h2	0.870373	0.868545	0.870373	0.870244	26.6246s	0.5863s
Decision Tree						
h1	0.895463	0.894087	0.895463	0.89517594.5512	94.5512s	0.5132s
h2	**0.908308**	**0.907951**	**0.908308**	**0.909504**	**108.4439s**	**0.5803s**
Random Forest						
h1	0.895214	0.893741	0.895214	0.894915	84.9413s	0.6111s
h2	0.907958	0.907321	0.907958	0.909113	140.0418s	1.1646s

Table C.3: Model Selection Results for the Subheading Classifier, using the "Real Data" Approach and the "Imbalanced" Training set

SUBHEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
s1	0.833589	0.832580	0.833589	0.844027	1.4975s	0.3506s
s2	0.830066	0.830760	0.830066	0.845174	1.2785s	0.2662s
s3	0.846666	0.848496	0.846666	0.859082	1.3247s	0.3055s
s4	0.836574	0.832865	0.836574	0.840000	1.9439s	0.3934s
s5	0.838642	0.841278	0.838642	0.852153	1.2042s	0.2636s
s6	0.833472	0.829615	0.833472	0.837865	1.1830s	0.2385s
s7	0.842639	0.841126	0.842639	0.847937	1.3533s	0.3041s
s8	0.839614	0.838821	0.839614	0.845114	1.4321s	0.2952s
Naive Bayes Gaussian						
s1	0.616701	0.600987	0.616701	0.781786	1.4663s	1.2783s
s2	0.616996	0.601514	0.616996	0.782235	0.8465s	0.7583s
s3	0.616444	0.602190	0.616444	0.784175	0.8820s	0.8204s
s4	0.617136	0.600723	0.617136	0.781818	1.3348s	1.1323s
s5	0.613902	0.600864	0.613902	0.783868	0.8454s	0.8052s
s6	0.617129	0.601101	0.617129	0.781431	0.8817s	0.8098s
s7	0.616864	0.601936	0.616864	0.784251	0.8902s	0.9127s
s8	0.616903	0.601991	0.616903	0.783904	0.9398s	0.9044s
Multinomial Logistics Regression						
s1	0.906714	0.901154	0.906714	0.903628	32.0426s	0.3097s
s2	0.909451	0.904628	0.909451	0.907435	30.0180s	0.3028s
s3	0.912918	0.909151	0.912918	0.910364	30.0345s	0.3123s
s4	0.907538	0.901880	0.907538	0.903943	36.8663 s	0.4273s
s5	0.914885	0.911115	0.914885	0.912677	31.1199s	0.3303s
s6	0.910251	0.905238	0.910251	0.907950	30.2684s	0.3188s
s7	0.914893	0.911276	0.914893	0.912382	31.6543s	0.3498s
s8	0.916129	0.912276	0.916129	0.913773	35.4001s	0.3497s
Decision Tree						
s1	0.925918	0.922915	0.925918	0.926371	105.0901s	0.5227s
s2	0.931726	0.928631	0.931726	0.932521	101.9350s	0.4830s
s3	0.936042	0.933187	0.936042	0.935542	97.3386s	0.5032s
s4	0.926797	0.923812	0.926797	0.927404	118.4395s	0.5829s
s5	0.941640	0.939441	0.941640	0.941318	106.0036s	0.5089s
s6	0.932613	0.929619	0.932613	0.933473	102.7272s	0.5250s
s7	0.936897	0.934075	0.936897	0.936380	105.0036s	0.5018s
s8	**0.942231**	**0.940069**	**0.942231**	**0.941885**	**104.4316s**	**0.5648s**
Random Forest						
s1	0.925320	0.922230	0.925320	0.926055	92.4553s	0.6132s
s2	0.931470	0.928376	0.931470	0.932177	93.8773s	0.6142s
s3	0.935591	0.932738	0.935591	0.935250	86.4305s	0.5703s
s4	0.926338	0.923119	0.926338	0.927287	109.2299s	0.6234s
s5	0.941375	0.939070	0.941375	0.941164	87.9163s	0.6132s
s6	0.932154	0.929119	0.932154	0.932932	102.5980s	0.6140s
s7	0.936485	0.933654	0.936485	0.936089	88.9695s	0.6322s
s8	0.942145	0.939921	0.942145	0.941836	89.0147s	0.6148s

Table C.4: Model Selection Results for the Chapter Classifier, using the "Real Data" Approach and the "Random undersampling of the majority class" Training set

CHAPTER CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
c1	0.797170	0.789170	0.797170	0.817014	0.3753s	0.1045s
c2	0.797013	0.790096	0.797013	0.817286	0.4795s	0.1266s
Naive Bayes Gaussian						
c1	0.679371	0.651765	0.679371	0.702066	0.2736s	0.1297s
c2	0.689382	0.663484	0.689382	0.704170	0.5863s	0.1194s
Multinomial Logistics Regression						
c1	0.799405	0.793827	0.799405	0.815328	5.5502s	0.1836s
c2	0.802286	0.796734	0.802285	0.817852	4.1061s	0.1247s
Decision Tree						
c1	0.862177	0.862292	0.862178	0.864774	62.6295s	0.2091s
c2	**0.865546**	**0.865635**	**0.865546**	**0.869589**	**52.1716s**	**0.1983s**
Random Forest						
c1	0.861438	0.861556	0.861438	0.864320	40.5740s	0.2543s
c2	0.865373	0.865462	0.865373	0.869644	40.818240.8182s	0.3294s

Table C.5: Model Selection Results for the Heading Classifier, using the "Real Data" Approach and the "Random undersampling of the majority class" Training set

HEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
h1	0.703171	0.683569	0.703171	0.736357	0.4633s	0.1008s
h2	0.680631	0.661913	0.680631	0.720249	0.5555s	0.1251s
Naive Bayes Gaussian						
h1	0.136800	0.100894	0.136800	0.374448	0.4423s	0.2923s
h2	0.137587	0.102320	0.137587	0.376174	0.6519s	0.4148s
Multinomial Logistics Regression						
h1	0.713953	0.701799	0.713953	0.738541	8.7256s	0.1488s
h2	0.715448	0.703002	0.715448	0.737157	9.3929s	0.1728s
Decision Tree						
h1	0.826951	0.825109	0.826951	0.828359	50.1949s	0.2379s
h2	**0.846972**	**0.845528**	**0.846972**	**0.848649**	**47.8183s**	**0.2327s**
Random Forest						
h1	0.825691	0.823731	0.825691	0.827028	36.1356s	0.6111s
h2	0.907958	0.907321	0.907958	0.909113	140.0418s	0.2807s

Table C.6: Model Selection Results for the Subheading Classifier, using the "Real Data" Approach and the "Random undersampling of the majority class" Training set

SUBHEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
s1	0.632778	0.608848	0.632778	0.665908	0.5449s	0.0992s
s2	0.631440	0.606473	0.631440	0.664768	0.5531s	0.1100s
s3	0.655429	0.633851	0.655429	0.680984	0.5813s	0.1387s
s4	0.638586	0.609505	0.638586	0.652430	0.5425s	0.0944s
s5	0.652643	0.630870	0.652643	0.677150	0.5243s	0.0940s
s6	0.629756	0.598405	0.629756	0.648503	0.6147s	0.1068s
s7	0.648503	0.623009	0.648503	0.655463	0.6868s	0.1592s
s8	0.651478	0.626363	0.651478	0.666511	0.5424s	0.1090s
Naive Bayes Gaussian						
s1	0.198218	0.157458	0.198218	0.532555	0.4166s	0.3578s
s2	0.198769	0.158194	0.198769	0.532303	0.5570s	0.5935s
s3	0.196062	0.160082	0.196062	0.576109	0.4689s	0.4462s
s4	0.195354	0.153735	0.195354	0.530336	0.4306s	0.4195s
s5	0.196093	0.160109	0.196093	0.576197	0.3666s	0.3457s
s6	0.196030	0.154209	0.196030	0.529639	0.4295s	0.4038s
s7	0.198706	0.162527	0.198706	0.585384	0.3889s	0.3753s
s8	0.200107	0.163638	0.200107	0.583794	0.4316s	0.4182s
Multinomial Logistics Regression						
s1	0.773749	0.749631	0.773749	0.763327	14.1114s	0.1204s
s2	0.776457	0.753242	0.776457	0.767566	19.0721s	0.1830s
s3	0.779526	0.757655	0.779526	0.774735	17.5750s	0.1968s
s4	0.775009	0.750852	0.775009	0.762839	15.9664s	0.1431s
s5	0.782659	0.761501	0.782659	0.778365	14.6434s	0.1255s
s6	0.776567	0.752991	0.776567	0.769380	16.3876s	0.1511s
s7	0.782265	0.761122	0.782265	0.776834	15.3398s	0.1351s
s8	0.783461	0.763263	0.783461	0.781693	16.3989s	0.1675s
Decision Tree						
s1	0.852261	0.846155	0.852261	0.853033	57.9935s	0.2299s
s2	0.863830	0.857605	0.863830	0.865678	69.1447s	0.3495s
s3	0.872173	0.866098	0.872173	0.871078	61.7820s	0.2393s
s4	0.854229	0.848102	0.854229	0.855208	55.0848s	0.2348s
s5	0.883207	0.878533	0.883207	0.882613	57.9001s	0.1968s
s6	0.865452	0.859409	0.865452	0.867385	49.0409s	0.2232s
s7	0.873920	0.867911	0.873920	0.872845	50.1173s	0.2273s
s8	**0.884639**	0**0.880033**	**0.884639**	**0.884002**	**53.1448s**	**0.2127s**
Random Forest						
s1	0.851427	0.845259	0.851427	0.852248	35.3310s	0.2564s
s2	0.863405	0.857225	0.863405	0.865021	46.9167s	0.3717s
s3	0.872298	0.866430	0.872299	0.871175	37.3824s	0.2673s
s4	0.853725	0.847557	0.853725	0.854882	47.1713s	0.3649s
s5	0.882262	0.877413	0.882262	0.881555	36.8904s	0.2635s
s6	0.865121	0.859058	0.865121	0.866695	37.1854s	0.2906s
s7	0.873369	0.867344	0.873369	0.872023	39.4938s	0.2959s
s8	0.942145	0.939921	0.942145	0.941836	89.0147s	0.6148s

Table C.7: Model Selection Results for the Chapter Classifier, using the "Real Data" Approach and the "Random Undersampling followed by Random Oversampling" Training set

CHAPTER CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
c1	0.795170	0.786265	0.795170	0.823841	7.5674s	2.0627s
c2	0.787349	0.777116	0.787349	0.819341	4.5350s	1.3724s
Naive Bayes Gaussian						
c1	0.572642	0.504501	0.572642	0.594082	4.7646s	2.5473s
c2	0.572026	0.503967	0.572026	0.593457	3.6963s	2.0063s
Multinomial Logistics Regression						
c1	0.797005	0.792016	0.797006	0.820149	59.0580s	1.8893s
c2	0.798422	0.794117	0.798422	0.817882	68.1550s	2.0521s
Decision Tree						
c1	0.867559	0.867487	0.867559	0.869676	1016.1726s	2.8528s
c2	**0.875604**	**0.875562**	**0.875604**	**0.877461**	**1002.5719s**	**2.7570s**
Random Forest						
c1	0.867552	0.867475	0.867552	0.869738	818.1356s	2.8633s
c2	0.875563	0.875515	0.875563	0.877514	1032.3425s	3.6138s

Table C.8: Model Selection Results for the Heading Classifier, using the "Real Data" Approach and the "Random Undersampling followed by Random Oversampling" Training set

HEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
h1	0.630917	0.593414	0.630917	0.652397	3.9885s	1.2730s
h2	0.613790	0.581704	0.613790	0.655304	5.0652s	1.5579s
Naive Bayes Gaussian						
h1	0.297183	0.238386	0.297183	0.506136	2.9928s	2.5903s
h2	0.303474	0.242206	0.303474	0.501115	3.2983s	2.9817s
Multinomial Logistics Regression						
h1	0.651862	0.630366	0.651862	0.664172	90.2048s	1.8228s
h2	0.654436	0.654435	0.654436	0.670404	103.8801s	1.9571s
Decision Tree						
h1	0.814882	0.812995	0.814882	0.815165	783.4968s	3.2521s
h2	**0.835102**	**0.833996**	**0.835102**	**0.836701**	**960.1509s**	**3.0942s**
Random Forest						
h1	0.814990	0.813070	0.814990	0.816239	791.7550s	3.5849s
h2	0.835045	0.833918	0.835045	0.836583	837.8175s	3.4105s

Table C.9: Model Selection Results for the Subheading Classifier, using the "Real Data" Approach and the "Random Undersampling followed by Random Oversampling" Training set

SUBHEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
s1	0.454253	0.427761	0.454253	0.495659	5.8232s	1.4191s
s2	0.451998	0.423101	0.451998	0.481521	4.9595s	1.3412s
s3	0.482803	0.453726	0.482803	0.512610	5.0711s	1.3554s
s4	0.450605	0.420502	0.450605	0.486098	6.0356s	1.6844s
s5	0.477422	0.447613	0.477422	0.509255	5.1966s	1.2837s
s6	0.448899	0.418167	0.448899	0.485434	8.9853s	1.9588s
s7	0.479811	0.448755	0.479811	0.508861	5.8008s	1.3150s
s8	0.468357	0.437407	0.468357	0.500854	5.6717s	1.3546s
Naive Bayes Gaussian						
s1	0.353052	0.288100	0.353052	0.450497	3.3446s	4.8025s
s2	0.354874	0.289689	0.354874	0.449427	3.475724s	4.7542s
s3	0.360808	0.298758	0.360808	0.466318	3.5827s	5.2517s
s4	0.356985	0.291749	0.356985	0.452938	3.6959s	5.0752s
s5	0.362319	0.299917	0.362319	0.469301	3.4823s	5.1070s
s6	0.358851	0.293370	0.358851	0.454028	5.2461s	6.7537s
s7	0.363693	0.301419	0.363693	0.464321	3.7436s	5.4367s
s8	0.364355	0.301624	0.364355	0.465181	3.6571s	5.6009s
Multinomial Logistics Regression						
s1	0.633302	0.616306	0.633302	0.651018	162.6264s	1.7427s
s2	0.648326	0.632734	0.648326	0.668965	169.6353s	1.9475s
s3	0.658663	0.643670	0.658663	0.677963	174.0335s	1.8798s
s4	0.646905	0.630044	0.646905	0.670295	172.8511s	1.8069s
s5	0.670677	0.657157	0.670677	0.690551	171.0375s	1.7811s
s6	0.652534	0.637933	0.652534	0.678080	196.8583s	2.0278s
s7	0.668958	0.654378	0.668958	0.688889	176.1697s	1.9171s
s8	0.675060	0.663304	0.675060	0.701844	173.0962s	1.6522s
Decision Tree						
s1	0.767884	0.767283	0.767884	0.784605	839.0230s	2.8513s
s2	0.789243	0.788948	0.789243	0.799388	894.9593s	3.0384s
s3	0.821165	0.820785	0.821165	0.828721	1033.9487s	3.4730s
s4	0.772462	0.772145	0.772462	0.788293	1065.4043s	2.9910s
s5	0.839422	0.837447	0.839422	0.843605	1033.3801s	3.1321s
s6	0.792819	0.792359	0.792819	0.801750	1394.2804s	4.2915s
s7	0.824593	0.824066	0.824593	0.831117	1364.0490s	3.7467s
s8	**0.841918**	0**0.840538**	**0.841918**	**0.846033**	**1162.1129s**	**3.0422s**
Random Forest						
s1	0.767965	0.767278	0.767965	0.784599	791.2474s	3.2425s
s2	0.789158	0.788905	0.789158	0.799237	933.8458s	3.4487s
s3	0.821026	0.820667	0.821026	0.828572	876.2528s	3.2480s
s4	0.772436	0.772188	0.772436	0.788171	874.71523s	3.5672s
s5	0.839349	0.837411	0.839349	0.843507	940.8261s	3.9575s
s6	0.792690	0.792297	0.792690	0.801526	1057.5191s	3.9595s
s7	0.824434	0.823937	0.824434	0.831107	1170.9336s	4.3870s
s8	0.841911	0.840651	0.841911	0.846116	890.8271s	3.1953s

Table C.10: Model Selection Results for the Chapter Classifier, using the "Predicted Data" Approach and the "Imbalanced" Training set

CHAPTER CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
c1	0.883419	0.889363	0.883419	0.909006	0.9122s	0.2649s
c2	0.862598	0.870893	0.862598	0.900368	0.8780s	0.2495s
Naive Bayes Gaussian						
c1	0.851168	0.840157	0.851168	0.841812	0.6437s	0.3383s
c2	0.852576	0.842489	0.852576	0.843607	0.5863s	0.3065s
Multinomial Logistics Regression						
c1	0.923897	0.925116	0.923897	0.927423	10.8523s	0.4411s
c2	0.923143	0.924525	0.923143	0.927282	9.8860s	0.3388s
Decision Tree						
c1	0.928640	0.929641	0.928640	0.931474	148.1396s	0.6652
c2	**0.930498**	**0.931690**	**0.930498**	**0.934124**	**138.0479s**	**0.6989s**
Random Forest						
c1	0.928414	0.929427	0.928414	0.931281	184.8735s	1.1996s
c2	0.930335	0.931531	0.930335	0.933975	86.0617s	0.6228s

Table C.11: Model Selection Results for the Heading Classifier, using the "Predicted Data" Approach and the "Imbalanced" Training set

HEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
h1	0.802045	0.811421	0.802045	0.836748	1.3358s	0.3578s
h2	0.788213	0.801114	0.788213	0.832531	1.0462s	0.2781s
Naive Bayes Gaussian						
h1	0.517972	0.545955	0.517972	0.818871	0.7688s	0.5199s
h2	0.518299	0.546573	0.518299	0.820249	0.7588s	0.5500s
Multinomial Logistics Regression						
h1	0.848797	0.847047	0.848797	0.850357	17.2897s	0.3289s
h2	0.848361	0.846679	0.848361	0.850104	19.464s	0.3850s
Decision Tree						
h1	0.876220	0.874428	0.876220	0.8766745	103.9121s	0.5787s
h2	**0.880690**	**0.879392**	**0.880690**	**0.882591**	**101.8395s**	**0.6882s**
Random Forest						
h1	0.875932	0.874150	0.875932	0.876325	88.3179s	0.6081s
h2	0.880652	0.879323	0.880652	0.882616	92.2402s	0.6049s

Table C.12: Model Selection Results for the Subheading Classifier, using the "Predicted Data" Approach and the "Imbalanced" Training set

SUBHEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
s1	0.779683	0.776496	0.779683	0.793756	1.2850s	0.25181s
s2	0.774497	0.771029	0.774497	0.785660	1.1645s	0.2431s
s3	0.794052	0.794527	0.794052	0.810576	1.2037s	0.2352s
s4	0.783913	0.778044	0.783913	0.793584	1.2347s	0.2288s
s5	0.792131	0.792320	0.792131	0.806315	1.1702s	0.2334s
s6	0.774101	0.769014	0.774101	0.780689	1.1814s	0.2334s
s7	0.789472	0.788695	0.789472	0.801404	1.3162s	0.3034s
s8	0.788617	0.786430	0.788617	0.796658	1.4206s	0.3325s
Naive Bayes Gaussian						
s1	0.057808	0.046414	0.057808	0.777479	0.8590s	0.7539s
s2	0.058609	0.047196	0.058609	0.778157	0.8444s	0.7434s
s3	0.505097	0.524654	0.505097	0.773300	0.8428s	0.7530s
s4	0.058236	0.048310	0.058236	0.775474	0.8566s	0.7828s
s5	0.505680	0.525458	0.505680	0.772864	0.8493s	0.7760s
s6	0.059199	0.049406	0.059199	0.776304	0.9649s	0.8530s
s7	0.507002	0.528520	0.507002	0.777539	1.3640s	1.2658s
s8	0.507336	0.529175	0.507336	0.778053	0.9764s	0.9081s
Multinomial Logistics Regression						
s1	0.824849	0.819792	0.824849	0.832736	34.0927s	0.3311s
s2	0.826785	0.821479	0.826785	0.835329	28.9158s	0.3048s
s3	0.832181	0.827624	0.832181	0.834845	28.9507s	0.2988s
s4	0.826397	0.821350	0.826397	0.834201	29.6832s	0.3016s
s5	0.832881	0.828160	0.832881	0.835077	29.2533s	0.3057s
s6	0.827602	0.822154	0.827602	0.835248	30.5307s	0.3420s
s7	0.834133	0.829308	0.834133	0.837198	41.491s	0.4640s
s8	0.834382	0.829333	0.834381	0.838209	32.1988s	0.2980s
Decision Tree						
s1	0.839288	0.836609	0.839288	0.847474	106.7761s	0.5461s
s2	0.840571	0.837994	0.840571	0.848814	102.9417s	0.4850s
s3	0.845765	0.843265	0.845765	0.851149	88.4398s	0.4829s
s4	0.839964	0.837370	0.839964	0.848348	86.5324s	0.4845s
s5	0.847133	0.844803	0.847133	0.852688	93.9565s	0.5141s
s6	0.841270	0.838813	0.841270	0.849472	91.3150s	0.4891s
s7	0.846503	0.844124	0.846503	0.851989	156.4672s	0.7388s
s8	**0.847708**	**0.845455**	**0.847708**	**0.853155**	**116.6975s**	**0.5940s**
Random Forest						
s1	0.839093	0.836445	0.839093	0.847161	87.6435s	0.6281s
s2	0.840299	0.837689	0.840299	0.848307	100.9633s	0.5570s
s3	0.845982	0.843306	0.845982	0.851346	84.4140s	0.5833s
s4	0.839622	0.837012	0.839622	0.847836	107.9236s	0.5494s
s5	0.847009	0.844430	0.847009	0.852387	88.9847s	0.6158s
s6	0.8408734	0.838273	0.840874	0.849024	92.4124s	0.5582s
s7	0.846565	0.844050	0.846565	0.852091	109.4449s	0.7079s
s8	0.847732	0.845345	0.847732	0.853440	85.9739s	0.5876s

Table C.13: Model Selection Results for the Chapter Classifier, using the "Real Data" Approach and the "Random undersampling of the majority class" Training set

CHAPTER CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
c1	0.797170	0.789170	0.797170	0.817014	0.3753s	0.1045s
c2	0.797013	0.790096	0.797013	0.817286	0.4795s	0.1266s
Naive Bayes Gaussian						
c1	0.679371	0.651765	0.679371	0.702066	0.2736s	0.1297s
c2	0.689382	0.663484	0.689382	0.704170	0.5863s	0.1194s
Multinomial Logistics Regression						
c1	0.799405	0.793827	0.799405	0.815328	5.5502s	0.1836s
c2	0.802286	0.796734	0.802285	0.817852	4.1061s	0.1247s
Decision Tree						
c1	0.862177	0.862292	0.862178	0.864774	62.6295s	0.2091s
c2	**0.865546**	**0.865635**	**0.865546**	**0.869589**	**52.1716s**	**0.1983s**
Random Forest						
c1	0.861438	0.861556	0.861438	0.864320	40.5740s	0.2543s
c2	0.865373	0.865462	0.865373	0.869644	40.818240.8182s	0.3294s

Table C.14: Model Selection Results for the Heading Classifier, using the "Real Data" Approach and the "Random undersampling of the majority class" Training set

HEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
h1	0.666007	0.651753	0.666007	0.692514	0.4381s	0.1056s
h2	0.645954	0.634280	0.645954	0.689537	0.5142s	0.1117s
Naive Bayes Gaussian						
h1	0.135367	0.096442	0.135367	0.382186	0.3660s	0.2354s
h2	0.135997	0.097524	0.135997	0.382555	0.3791s	0.2176s
Multinomial Logistics Regression						
h1	0.685903	0.677789	0.685903	0.705288	7.6144s	0.1254s
h2	0.683998	0.676963	0.683998	0.704978	7.8544s	0.1422s
Decision Tree						
h1	0.787962	0.785764	0.787962	0.791488	48.9217s	0.2585s
h2	**0.793392**	**0.791399**	**0.793392**	**0.797968**	**51.3821s**	**0.2472s**
Random Forest						
h1	0.787222	0.784992	0.787222	0.791116	42.7600s	0.2944s
h2	0.793471	0.791582	0.793471	0.798141	40.4937s	0.2772s

Table C.15: Model Selection Results for the Subheading Classifier, using the "Real Data" Approach and the "Random undersampling of the majority class" Training set

SUBHEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
s1	0.565331	0.538570	0.565331	0.571513	0.5631s	0.1015s
s2	0.559743	0.531865	0.559743	0.567024	0.5657s	0.1036s
s3	0.591492	0.570574	0.591492	0.597345	0.5419s	0.0818s
s4	0.561365	0.532286	0.561365	0.563639	0.5439s	0.1006s
s5	0.588800	0.569455	0.588801	0.600357	0.6492s	0.1294s
s6	0.560829	0.528267	0.560829	0.564792	0.5624s	0.0992s
s7	0.589320	0.564661	0.589320	0.583113	0.5679s	0.1019s
s8	0.591886	0.566987	0.591886	0.591318	0.7264s	0.1669s
Naive Bayes Gaussian						
s1	0.113126	0.083842	0.113126	0.370847	0.4084s	0.3650s
s2	0.115424	0.085799	0.115424	0.372899	0.4073s	0.3662s
s3	0.111914	0.083697	0.111914	0.385207	0.3881s	0.3620s
s4	0.124443	0.099082	0.124443	0.398287	0.4036s	0.3825s
s5	0.112780	0.084725	0.112780	0.384616	0.3822s	0.3649s
s6	0.125199	0.099586	0.125199	0.398481	0.4184s	0.3917s
s7	0.123877	0.099090	0.123877	0.418876	0.4611s	0.4361s
s8	0.124097	0.100184	0.124097	0.419817	0.5267s	0.4999s
Multinomial Logistics Regression						
s1	0.659050	0.631048	0.659050	0.646249	15.5541s	0.1330s
s2	0.660986	0.632893	0.660987	0.646609	21.5768s	0.2081s
s3	0.666920	0.639402	0.666920	0.650766	15.9355s	0.1557s
s4	0.659963	0.632251	0.659963	0.644057	16.6612s	0.1599s
s5	0.667440	0.639701	0.667440	0.648679	16.0851s	0.1572s
s6	0.662151	0.633885	0.662151	0.647064	17.4768s	0.1947s
s7	0.668857	0.641443	0.668857	0.649396	16.2679s	0.1595s
s8	0.668841	0.641563	0.668841	0.649367	16.9068s	0.1696s
Decision Tree						
s1	0.711007	0.703599	0.711007	0.724202	57.1603s	0.2425s
s2	0.713715	0.706323	0.713715	0.727370	68.2126s	0.2504s
s3	0.722938	0.715564	0.722938	0.730610	57.7328s	0.2764s
s4	0.712408	0.705111	0.712408	0.726168	52.4185s	0.2358s
s5	0.725599	0.718365	0.725599	0.733232	51.9935s	0.2375s
s6	0.714974	0.707717	0.714974	0.728701	51.2743s	0.2583s
s7	0.724276	0.717104	0.724276	0.732062	55.3729s	0.2343s
s8	**0.726952**	0**0.719867**	**0.726952**	**0.734387**	**61.0417s**	**0.2509s**
Random Forest						
s1	0.711070	0.702976	0.711070	0.725151	37.0362s	0.2900s
s2	0.713117	0.705589	0.713117	0.726488	43.4467s	0.2766s
s3	0.723284	0.715786	0.723284	0.731262	43.3296s	0.3111s
s4	0.712314	0.704677	0.712314	0.725704	41.7202s	0.2986s
s5	0.725662	0.718566	0.725662	0.733410	38.4193s	0.2776s
s6	0.714486	0.707268	0.714486	0.727887	41.2229s	0.2836s
s7	0.724544	0.717130	0.724544	0.732561	44.2143s	0.2806s
s8	0.726512	0.719099	0.726512	0.734155	43.8653s	0.2928s

Table C.16: Model Selection Results for the Chapter Classifier, using the "Predicted Data" Approach and the "Random Undersampling followed by Random Oversampling" Training set

CHAPTER CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
c1	0.795170	0.786265	0.795170	0.823841	7.5674s	2.0627s
c2	0.787349	0.777116	0.787349	0.819341	4.5350s	1.3724s
Naive Bayes Gaussian						
c1	0.572642	0.504501	0.572642	0.594082	4.7646s	2.5473s
c2	0.572026	0.503967	0.572026	0.593457	3.6963s	2.0063s
Multinomial Logistics Regression						
c1	0.797005	0.792016	0.797006	0.820149	59.0580s	1.8893s
c2	0.798422	0.794117	0.798422	0.817882	68.1550s	2.0521s
Decision Tree						
c1	0.867559	0.867487	0.867559	0.869676	1016.1726s	2.8528s
c2	**0.875604**	**0.875562**	**0.875604**	**0.877461**	**1002.5719s**	**2.7570s**
Random Forest						
c1	0.867552	0.867475	0.867552	0.869738	818.1356s	2.8633s
c2	0.875563	0.875515	0.875563	0.877514	1032.3425s	3.6138s

Table C.17: Model Selection Results for the Heading Classifier, using the "Predicted Data" Approach and the "Random Undersampling followed by Random Oversampling" Training set

HEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
h1	0.617461	0.578394	0.617461	0.648679	7.9526s	2.4392s
h2	0.597799	0.566230	0.597799	0.648002	4.2677s	1.3182s
Naive Bayes Gaussian						
h1	0.288574	0.229014	0.288575	0.517535	5.64018s	4.0894s
h2	0.290088	0.229642	0.290088	0.508989	3.0632s	2.7142s
Multinomial Logistics Regression						
h1	0.637857	0.613126	0.637857	0.653273	135.6004s	3.1595s
h2	0.639922	0.617165	0.639923	0.655928	86.8677s	1.6180s
Decision Tree						
h1	0.793384	0.791245	0.793384	0.794695	1110.4688s	5.0884s
h2	**0.800659**	**0.798463**	**0.800659**	**0.801833**	**1206.1018s**	**4.7357s**
Random Forest						
h1	0.79338	0.791274	0.793383	0.794541	894.4696s	5.1358s
h2	0.800604	0.798388	0.800604	0.801747	1175.0370s	4.1991s

Table C.18: Model Selection Results for the Subheading Classifier, using the "Predicted Data" Approach and the "Random Undersampling followed by Random Oversampling" Training set

SUBHEADING CLASSIFIER						
Dataset	Avg Accuracy	Avg f1-score	Avg Recall	Avg Precision	Training time	Predictive Time
Naive Bayes Bernoulli						
s1	0.390118	0.359263	0.390118	0.402201	5.5528s	1.4535s
s2	0.382829	0.352047	0.382829	0.394776	6.0952s	1.6071s
s3	0.415562	0.384476	0.415562	0.413198	5.3410s	1.2825s
s4	0.392195	0.358955	0.392195	0.406258	8.1656s	2.0402s
s5	0.412295	0.382261	0.412295	0.419014	5.8050s	1.3973s
s6	0.409243	0.374397	0.409243	0.402498	5.4846s	1.3303s
s7	0.409008	0.374868	0.409008	0.403952	8.1488s	1.9030s
s8	0.397975	0.366513	0.397975	0.405634	5.4619s	1.4076s
Naive Bayes Gaussian						
s1	0.242860	0.181328	0.242860	0.328961	3.6199s	5.2261s
s2	0.244758	0.184628	0.244758	0.333383	4.2425s	6.0332s
s3	0.251151	0.194734	0.251151	0.348389	3.8243s	5.3232s
s4	0.244683	0.182997	0.244683	0.359620	5.3664s	7.3460s
s5	0.243219	0.192377	0.243219	0.347878	3.4217s	5.0231s
s6	0.249910	0.194476	0.249910	0.361437	3.5442s	5.4089s
s7	0.248191	0.194049	0.248191	0.364662	3.8708s	5.5803s
s8	0.240352	0.191326	0.240352	0.363155	3.6716s	5.3189s
Multinomial Logistics Regression						
s1	0.507764	0.492303	0.507764	0.517587	177.7101s	2.2567s
s2	0.512996	0.494797	0.512996	0.525963	188.7495s	1.8439s
s3	0.527363	0.510277	0.527363	0.542174	173.7772s	1.7813s
s4	0.510108	0.493905	0.510108	0.520987	195.9713s	2.1574s
s5	0.541363	0.526679	0.541363	0.557090	165.9176s	1.7602s
s6	0.537366	0.520964	0.537366	0.552854	168.8481s	1.7460s
s7	0.535522	0.518363	0.535522	0.550982	169.1796s	1.7453s
s8	0.544444	0.529583	0.544443	0.560831	169.3929s	1.8759s
Decision Tree						
s1	0.619148	0.615363	0.619148	0.637029	1225.5477s	3.4161s
s2	0.631652	0.629615	0.631652	0.650359	1453.5391s	3.8658s
s3	0.653841	0.654100	0.653841	0.669627	1119.5453s	3.3577s
s4	0.625085	0.622598	0.625085	0.645702	1154.3785s	3.1558s
s5	0.658335	0.659059	0.658335	0.675164	1141.1581s	2.9293s
s6	0.658185	0.658874	0.658185	0.675201	1023.8651s	3.5623s
s7	0.656855	0.656824	0.656855	0.671477	1131.2238s	2.8761s
s8	**0.661651**	0**0.662440**	**0.661651**	**0.677882**	**1185.4316s**	**2.8261s**
Random Forest						
s1	0.619181	0.615439	0.619181	0.638035	904.4443s	3.6933s
s2	0.631723	0.629758	0.631723	0.651101	1186.0800s	3.8523s
s3	0.653762	0.654209	0.653762	0.670221	978.2431s	3.7473s
s4	0.625084	0.622875	0.625084	0.647607	912.5916s	3.5776s
s5	0.657200	0.657238	0.657200	0.671839	1197.5283s	2.9490s
s6	0.634318	0.632458	0.634318	0.654326	1081.0777s	3.4688s
s7	0.656768	0.656830	0.656768	0.671914	933.1844s	3.3887s
s8	0.661529	0.6623089	0.661529	0.677957	1032.5960s	4.0639s