NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ALICE RODRIGUES DIAS

BSc in Electrical and Computer Engineering

DATA MINING PROJECT TO IMPROVE SOFT SKILLS ACQUISITION AND ENGINEERING EDUCATION

EVALUATION OF CTCT COURSE RESULTS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
November, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

DATA MINING PROJECT TO IMPROVE SOFT SKILLS ACQUISITION AND ENGINEERING EDUCATION

EVALUATION OF CTCT COURSE RESULTS

ALICE RODRIGUES DIAS

BSc in Electrical and Computer Engineering

Adviser: Fernando Luís Lourenço Ferreira
Invited Assistant Professor, NOVA School of Science and Technology - Nova University of Lisbon

Co-adviser: Nelson Fernando Chibeles Pereira Martins
Associate Professor, NOVA School of Science and Technology - Nova University of Lisbon

Examination Committee

Chair: Rui Alexandre Nunes Neves da Silva
Associate Professor, NOVA School of Science and Technology - Nova University of Lisbon

Rapporteur: Rui Manuel Leitão Tavares
Assistant Professor, NOVA School of Science and Technology - Nova University of Lisbon

Adviser: Fernando Luís Lourenço Ferreira
Invited Assistant Professor, NOVA School of Science and Technology - Nova University of Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
November, 2024

Data Mining Project to improve Soft Skills Acquisition and Engineering Education

Evaluation of CTCT Course Results

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.0.3) [1].

To my boyfriend, Daniel, who bore the weight of this project with me and, at times, even more than I did. To my mother, who (almost) never gave up on me, and whose unwavering support kept me going. To my little dog, Mortred, whose companionship gave me the strength to persevere. And to my late grandmother, Preciosa, and my nanny, Catarina—though they left many years ago, they will always remain in my heart.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Fernando, and co-advisor, Nelson, for their invaluable guidance, support, data, time, and the opportunities they provided throughout this journey. I am also sincerely thankful to all the teachers at NOVA School of Science and Technology who have supported me in countless ways—from offering reassurance during challenging web application tests to providing diverse and insightful advice. I received valuable guidance regarding web application development, data analysis, and even life advice—especially on managing perfectionism, frustrations, and navigating the complexities of this entire experience.

I extend my sincere appreciation to NOVA School of Science and Technology for supporting my participation as a virtual author at the EDULEARN24 conference and for being an exceptional institution—a sentiment I can confidently express after attending several other engineering schools before finding my place here. While NOVA may sometimes face resource limitations, these challenges are more than offset by the dedication of its faculty and staff, whose constant commitment ensures the school's smooth functioning.

I also owe my deepest thanks to my parents for their unwavering financial and emotional support throughout my endless pursuit of knowledge, especially my mother. Beyond the essential financial help, my mother has always been there for me—assisting with domestic tasks, listening to my frustrations, and offering comfort whenever I needed it. Over the past 18 years, her continuous support in these small but significant ways has made this journey possible.

Finally, I must truly thank my boyfriend, Daniel. Without him, I would not have made it this far. Most of my degree was completed while living with him, and it's safe to say that his patience and steady support were essential in helping me manage the stress before every project deadline. He has listened to me talk about this dissertation—and the CTCT course—nearly every day for the past year and a half. Daniel, I couldn't have done this without you, and I can't thank you enough for your understanding, patience, and encouragement.

”

“You cannot teach a man anything; you can only help him discover it in himself.”

— Galileo, Somewhere in a book or speach
(Astronomer, physicist and engineer)

Abstract

This dissertation explores the potential of educational data mining (EDM) to enhance the engineering curriculum, specifically within the Transversal Skills in Science and Technology mandatory program for first-year students at NOVA School of Science and Technology. It employs machine learning (ML) algorithms, including classification and regression, to analyze student performance data. Various research concerns were addressed, such as the impact of gender on course selection, the relationships between different assessment points, performance variability, and the predictive power of early-course outcomes. Some of these questions were resolved through statistical analysis, while others leveraged machine learning models.

Key findings reveal a distinct gender-based difference in course selection, with male students typically opting for engineering fields like mechanical, electrical, and informatics, while female students gravitated towards biology and chemistry-related courses. Additionally, activity points proved to be the most reliable predictor of final grades, whereas self and peer-assessment points exhibited lower correlations, indicating a shift in the weighting of these points in the evaluation system. A detailed analysis of demographic and performance data showed that male students had a higher rate of failure than female students, despite enrolling in larger numbers.

While the clustering analysis identified potential trends in early-course data, it did not provide substantial actionable insights in isolation. However, its integration into predictive models could offer valuable contributions in forecasting student performance in future research. The machine learning models demonstrated the effectiveness of early-course data in predicting final grades, with Random Forest and CatBoost models consistently outperforming others.

The dissertation emphasizes the technical capabilities of platforms such as Google Colaboratory, highlighting its cost-effectiveness and efficiency for conducting large-scale educational data analysis using mid-range Graphical Processing Units (GPUs). Moreover, the research underscores the importance of standardizing data collection through Learning Management Systems (LMS), specifically Moodle, which can enhance the precision and efficiency of educational data analysis.

The research concludes with recommendations for future work, including the development of Moodle plugins to track student performance in real-time, further exploration of machine learning models for grade prediction, and an in-depth analysis of computational resource costs. This study contributes to the field of Educational Data Mining (EDM) and provides practical insights for improving Science, Technology, Engineering, and Mathematics (STEM) education through data-driven strategies and the integration of Artificial Intelligence (AI).

Keywords: Artificial Intelligence in Education, Educational Data Mining, Knowledge Discovery, Google Colaboratory, Machine Learning, STEM Education, Predictive Modeling, Moodle, Student Performance Analysis

Resumo

Esta dissertação explora as potencialidades da extração de dados educacionais (*Educational Data Mining*, EDM) na melhoria do ensino em engenharia, com foco específico na disciplina obrigatória para alunos do primeiro ano da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa: Competências Transversais em Ciência e Tecnologia, mais conhecida como CTCT. São utilizados algoritmos de aprendizagem automática (*Machine Learning*, ML), incluindo técnicas de *clustering* (agregação), classificação e regressão, para analisar dados de desempenho dos alunos e são abordadas diversas questões de investigação relacionadas com a população de alunos de CTCT e a sua evolução no contexto desta disciplina. Estas questões incluem aspetos como se a seleção do curso por parte do aluno tem relação com o sexo do mesmo, bem como o estudo de correlação entre os diversos pontos utilizados na avaliação dos alunos (resultados de atividades, pontos de participação e pontos de auto e heteroavaliação). Também se estuda a variabilidade do desempenho dos alunos em função de diversos fatores e tenta-se aferir o valor preditivo dos resultados das primeiras duas semanas da disciplina, resolvidas em parte com testes estatísticos e em parte com modelos de ML.

As principais conclusões mostram uma clara diferença de género na seleção de cursos, com os estudantes do sexo masculino a escolherem predominantemente cursos de engenharia, como mecânica, elétrica e informática, enquanto as estudantes do sexo feminino optam por cursos mais orientados para a biologia e a química. Além disso, os pontos de atividade foram identificados como o indicador mais confiável das notas finais (como se esperava), enquanto os pontos de auto e heteroavaliação apresentaram diferentes níveis de correlação, sugerindo uma evolução no peso deste tipo de pontos ao longo da história da disciplina. A análise de *clustering* identificou possíveis tendências nos dados iniciais dos cursos, embora, isoladamente, não tenha fornecido informações suficientemente conclusivas. No entanto, a sua integração em modelos preditivos pode oferecer contribuições valiosas para prever o desempenho dos estudantes em pesquisas futuras.

Este estudo destaca as capacidades tecnológicas de plataformas como o Google Colaboratory, que oferecem uma solução eficiente e economicamente viável para análises educacionais em larga escala, utilizando Unidades de Processamento Gráfico (*Graphics*

Processing Units, GPUs) de gama intermédia. Além disso, realça a importância de padronizar a recolha de dados através da utilização de Sistemas de Gestão de Aprendizagem (*Learning Management Systems*, LMS), em particular o Moodle, que tem o potencial de melhorar significativamente a precisão e a eficácia da análise de dados educacionais.

Esta pesquisa oferece recomendações para trabalhos futuros, incluindo o desenvolvimento de *plug-ins* para o Moodle que permitam acompanhar em tempo real o desempenho dos alunos, a investigação aprofundada de modelos de aprendizagem automática para a previsão de notas finais, e uma análise detalhada dos custos associados ao uso de recursos computacionais de uma das ferramentas usada neste trabalho: o Google Colaboratory, que funciona com Jupyter Notebooks. Este estudo contribui para o campo da EDM e oferece conselhos práticos para melhorar o ensino superior nas áreas de Ciência, Tecnologia, Engenharia e Matemática (STEM), através da integração de técnicas avançadas de análise de dados impulsionadas pela Inteligência Artificial (IA).

Palavras-chave: Inteligência Artificial na Educação, Extração de Dados Educacionais, Descoberta de Conhecimento, Google Colaboratory, Aprendizagem Automática, Educação STEM, Modelagem Preditiva, Moodle, Análise de Desempenho Educacional

Contents

List of Figures

List of Tables

List of Code Snippets

Glossary

Society 5.0 A concept introduced by Shinzo Abe, emphasizing technology's role in enhancing human well-being through the integration of physical and cyberspace. Officially launched in Japan's Fifth Science and Technology Basic Plan by the CSTI in January 2016, it seeks to address societal issues and improve quality of life by leveraging ICT's for socioeconomic development. Key challenges such as poverty, demographic changes, and natural disasters are addressed, with the aim of fostering digital reforms and new business models [2, 3] *(pp. 1, 80)*

Acronyms

AE Autoencoder *(pp. 8, 12)*

AI Artificial Intelligence *(pp. 1, 2, 6–10, 13, 20, 21, 80, 175)*

ANN Artificial Neural Network *(pp. 8, 10–12, 36, 65, 74)*

AP Activity Point *(pp. 3, 31, 34, 35, 40, 45, 51, 54, 56, 71, 117, 130, 150)*

API Application Programming Interface *(pp. 176, 179)*

BN Bayesian Network *(pp. 10, 12, 23)*

CBRS Case-Based Reasoning System *(p. 9)*

CH Calinski-Harabasz *(pp. 36, 59–62, 151, 154, 157–159, 163)*

CNN Convolutional Neural Network *(p. 12)*

CPU Central Processing Unit *(pp. 3, 64–67, 75, 76, 169–174)*

CRISP-DM CRoss-Industry Standard Process Model for Data Mining *(pp. 24, 27, 34, 175)*

CSTI Council for Science, Technology and Innovation *(p. xviii)*

CSV Comma-Separated Value *(pp. 14, 15, 32, 177)*

CTCT Competências Transversais em Ciências e Tecnologia - Transversal Skills in Science and Technology *(pp. 1–4, 19, 20, 27–32, 38, 40, 42, 45, 49, 52–54, 56, 57, 62, 69, 70, 72, 74, 78–80, 89, 90, 93–96, 100, 102, 103, 107, 117, 118, 124, 140, 146, 175–177, 183–185)*

cu compute unit *(p. 76)*

DBI Davies-Bouldin Index *(pp. 23, 36, 59–62, 151, 154, 157–159)*

DBM Deep Boltzmann Machine *(pp. 8, 12)*

DIKW Data, Information, Knowledge, Wisdom *(p. 17)*

DL Deep Learning *(p. 8)*

DM Data Mining *(p. 9)*

DMP Data Management Plan *(pp. 37, 176)*

DNN Deep Neural Network *(p. 12)*

DSS Decision Support System *(p. 20)*

DT Decision Tree *(pp. 8, 10, 11, 33, 36, 65, 67, 73, 74, 141, 142, 145, 146, 168–173)*

EAI Educational Artificial Intelligence *(p. 8)*

EDM Educational Data Mining *(pp. 2, 3, 5, 9, 10, 20, 22, 150)*

FCT Faculdade de Ciências e Tecnologia - NOVA School of Science and Technology *(pp. 1, 19, 21, 38, 49, 77, 78, 80, 90, 118, 184)*

FL Fuzzy Logic *(p. 13)*

FR F-regression *(pp. 33, 65, 67, 68, 73, 169, 170, 172–174)*

GAN Generative Adversarial Network *(p. 8)*

GBT Gradient Boosting Tree *(p. 11)*

GPU Graphical Processing Unit *(pp. 3, 17, 64–69, 75, 76, 79, 169–174)*

GUI Graphic User Interface *(pp. 14–16)*

HE Higher Education *(pp. 1, 6, 8, 10, 12, 13, 80)*

HMM Hidden Markov Model *(p. 12)*

ICT Information and Communication Technology *(pp. xviii, 19)*

ITS Intelligent Tutoring System *(pp. 9, 12, 13)*

KBES Knowledge-Based Expert System *(p. 8)*

KMC K-means Clustering *(pp. 10, 11, 13, 23, 35, 59–62, 73, 150, 152–154, 157, 158, 163)*

KMS Knowledge Management System *(pp. 18–21)*

kNN k-Nearest Neighbors *(pp. 8, 10, 11, 13, 36, 65, 67, 69, 74, 168–173)*

LA Learning Analytics *(pp. 9, 10, 20–22, 80)*

LASSO Least Absolute Shrinkage and Selection Operator *(pp. 33, 34, 65, 67, 68, 73, 75, 168–170, 172–174)*

LD Learning Design *(p. 10)*

LDA Linear Discriminant Analysis *(p. 8)*

LMS Learning Management System *(pp. 1, 9, 20, 21, 77)*

LR Logistic Regression *(pp. 10, 11, 23, 36, 65, 66, 74, 170, 171)*

MAE Mean Absolute Error *(pp. 36, 66, 163, 164, 172)*

MI Mutual Information *(pp. 33, 35, 65–67, 73, 74, 141, 145, 146, 168–173)*

MIS Management Information System *(p. 20)*

ML Machine Learning *(pp. 3–10, 20, 27, 73)*

MSE Mean Squared Error *(pp. 23, 36, 66–69, 74, 163–165, 167, 171–174)*

NB Naive Bayes *(pp. 10–13)*

NLP Natural Language Processing *(p. 8)*

NN Neural Network *(pp. 8, 10, 65–67, 73, 169, 170, 172, 173)*

ORM Object-Relational Mapping *(p. 178)*

PCA Principal Component Analysis *(pp. 8, 34, 59, 132, 151, 154, 163, 164, 168)*

PP Participation Point *(pp. 3, 31, 34, 35, 40, 45, 51, 71, 117, 130, 150)*

R^2 Coefficient of Determination *(pp. 36, 63, 66–69, 163–165, 167, 171–174)*

RBM Restricted Boltzmann Machine *(p. 8)*

REST Representational State Transfer *(pp. 176, 178, 179)*

RF Random Forest *(pp. 10, 11, 23, 33, 36, 64–69, 73–75, 141, 142, 145, 146, 168–174)*

RL Reinforcement Learning *(p. 8)*

RNN Recurrent Neural Network *(p. 12)*

RQ Research Question *(pp. 3, 4, 27, 32, 35, 78)*

SCORM Shareable Content Object Reference Model *(pp. 21, 22, 77)*

SECI Socialization, Externalization, Combination and Internalization *(p. 19)*

SPA Self and Peer Assessment Point *(pp. 3, 34, 35, 40, 45, 49, 51, 71, 78, 117, 130, 140)*

STEM Science, Technology, Engineering and Mathematics *(pp. 1, 2, 4, 71, 80, 184, 185)*

SVM Support Vector Machine *(pp. 8, 10, 23, 36, 67, 74, 163, 170, 172–174)*

SVR Support Vector Regressor *(p. 36)*

UI User Interface *(p. 175)*

URL Uniform Resource Locator *(pp. 15, 182)*

USA United States of America *(p. 21)*

VBA Visual Basic for Applications *(pp. 27, 79, 175, 183, 184)*

WCSS Within-Cluster Sum of Squares *(p. 152)*

1 Introduction

This chapter provides an overview of the context and motivation behind the research conducted in this dissertation. It begins by introducing the challenges and opportunities in modern engineering education and highlights the importance of the Competências Transversais em Ciências e Tecnologia - Transversal Skills in Science and Technology (CTCT) course, whose data serves as the foundation for this study. The chapter outlines the shift in focus from the initial project scope to the current data-driven analysis, offering insights into the relevance of transversal skills development in preparing engineering students for the demands of Society 5.0.

1.1 Contextualization

Engineering is facing a transformational age after the digital revolution. To achieve Society 5.0, where technology enhances human well-being, educational content must adapt to changing market needs and social shifts [2, 3]. The focus of this alignment is on integrating transversal skills in Higher Education (HE), particularly in Science, Technology, Engineering and Mathematics (STEM) university degrees, where the gap between present educational practices and 21st-century workforce competencies is significant.

This dissertation explores the curricular unit CTCT at Faculdade de Ciências e Tecnologia - NOVA School of Science and Technology (FCT) using data mining techniques to extract insights and address the existing educational gaps. Initially, the project aimed to develop a system that should use Artificial Intelligence (AI) for real-time assistance and performance metrics to enhance user experience and inform teaching strategies, as it's explained in more detail in Annex I, but significant issues with the web application meant for data collection and real-time support surfaced during the development stage. The scalability of the system was limited to three users at a time due to its infrastructure's inability to manage several concurrent users. Moreover, creating a whole Learning Management System (LMS) from the ground up proved to be too ambitious for the project's duration. It was clear after almost a year of development and a deeper comprehension of the CTCT discipline that the application could not handle the planned metric development or data

collecting. As a result, the dissertation's emphasis was changed to use available data to analyze student performance over several CTCT course editions, making the study more practical and significant.

Introduced in 2012 to emphasize the importance of cross-cutting abilities in engineering education and preparing students for the evolving job market, the CTCT course covers a wide range of essential topics, including employability-focused curriculum planning, time management, leadership, teamwork, information retrieval from scientific databases, bibliographic reference management, science and technology communication, ethics and deontology, and advanced Excel spreadsheet skills. This curriculum is designed to enhance students' transversal skills for both academic and professional success. Initially a five-week program, it has evolved over the years into a four-week course as of 2024, comprising 48 practical hours and 8 theoretical hours. Each week focuses on different thematic areas, although the specific content has varied along the several discipline editions. Attendance is mandatory, with a requirement of at least 80% attendance in practical classes and attendance in at least three theoretical classes to pass. Evaluation is continuous, without a final exam, and special accommodations are provided for students with verified special statuses.

1.2 Objectives

Diverse datasets are evaluated using statistics and AI to identify patterns and make recommendations for improving education. This data corpus covers a wide range of topics, including a quantitative analysis of student performance across numerous activities to identify trends and correlations and determine the effectiveness of course components; a comprehensive review of existing research on AI applications in education, learning processes, and the development of transversal abilities in STEM education; qualitative insights from questionnaires administered to both CTCT students and instructors, offering a detailed view of perceived course value and impact; and the author's firsthand observations as a CTCT discipline monitor provide a unique viewpoint on course activity execution and student participation.

This dissertation seeks to contribute to the discussion on engineering education reform and establish new paradigms that prioritize a comprehensive skill set by integrating AI and Educational Data Mining (EDM) into curriculum development. The intention is to provide aspiring engineers with the skills they need to thrive in an increasingly technologically sophisticated and complicated society. This is the issue that this study was trying to address, but there are limits to the knowledge that can be extracted from data and it depends on the data that is accessible. This initiative also serves as a call to action to utilize the Moodle Learning Management System correctly, rather than merely as a place to store resources and use the data that this system can store to gain insights about how to enhance education.

Also, this research relies on the data that is now accessible, which is why the objectives for data mining were established only after receiving the initial collection of datasets. In order to align with the business objectives, it would be valuable to identify the specific activities that have the greatest impact on developing the abilities that the CTCT course aims to empower students with. However, there is currently no data available to substantiate this claim. This data mining project aims to analyze the qualities and characteristics of first-year students, as well as how these characteristics evolve and relate to each other. The objective is to comprehend the students in order to enhance their learning experiences throughout their science and engineering undergraduate program.

This dissertation also focuses on investigating and assessing Machine Learning (ML) tools and algorithms, with a focus on clustering, regression, and classification techniques, in order to overcome this constraint and broaden the research's scope. In addition to gaining a deeper understanding of the educational data under investigation, the goal is to disseminate these discoveries to the scientific community in order to promote Educational Data Mining (EDM) techniques. This technical approach seeks to investigate the potential of Google Colaboratory, a platform that provides high-performance Graphical Processing Unit (GPU) and Central Processing Unit (CPU) resources, allowing for in-depth comparisons of various approaches, and to evaluate the effectiveness of algorithms using robust performance measures.

By integrating advanced data mining techniques with an in-depth analysis of student performance, this research not only aims to enhance educational practices within the CTCT course but also to contribute to the broader field of Educational Data Mining.

Research Questions

The research questions, which are derived from the accessible data, are shown in Table 1.1 and emerged from the business knowledge phase of this project.

Table 1.1: Summary of this project research questions.

ID	Research Questions
RQ1	How does gender influence the choice of academic courses among students at the NOVA School of Science and Technology over various academic years?
RQ2	What is the nature of the relationships among different types of assessment points — Activity, Participation and Self and Peer Assessment Points (AP, PP, and SPA) — across various academic years in the CTCT discipline? Additionally, how do these points correlate with the final grades obtained by the students?

Continued on next page

Table 1.1: (continued)

ID	Research Questions
RQ3	In what ways does student performance in the CTCT discipline vary across editions, student demographics (course and gender), specific weeks, and thematic content, and how are these variations reflected through average points scored?
RQ4	What distinct patterns can be identified in student performance based on their assessment scores in the CTCT discipline?
RQ5	Can early performance in a specific theme or week of the CTCT course predict student outcomes in later themes or weeks? Furthermore, how predictive is the early performance of the final grade (`FinalGrade`) achieved in the course?

This framework aims to investigate each research question thoroughly. For instance, RQ1 (Research Question 1) addresses the influence of gender on course selections within STEM disciplines, highlighting potential demographic impacts on academic paths. RQ2 explores how different assessment types interrelate and their collective impact on student evaluations, important for understanding how engagement in various academic activities correlates with performance outcomes.

RQ3 addresses performance variability across different demographics (sex and student university degree), CTCT editions, weeks, and thematic content, trying to get insights about what influences this specific curricular unit grades. RQ4 focuses on identifying performance clusters and RQ5 examines the predictive value of early course performance on later outcomes, offering insights into effective curriculum sequencing and potential adjustments to enhance learning trajectories.

Beyond the educational analysis, this work intends to explore ML from a technical and methodological perspective, with the goal of making major contributions to the scientific community. This dissertation examines several ML algorithms, such as clustering, regression, and classification techniques, across different computational resources offered by Google Colaboratory using student performance data from the CTCT course. The objective is to assess these algorithms' efficiency and efficacy in educational settings, therefore laying a solid basis for further research aimed at implementing ML in the classroom. This technical focus has significance for both verifying the validity of the selected algorithms and ensuring that the produced approaches are practical and efficient in real-world settings, maximizing the use of computer resources.

1.3 Document Structure

This dissertation is structured to guide the reader through the research process in a logical flow. Chapter 2 begins with a comprehensive review of the state of the art in Educational Data Mining (EDM), identifying existing knowledge gaps and establishing the theoretical foundation for the study. Following this, Chapter 3 describes the overall methodology, including feature selection and model construction, providing a detailed outline of the data analysis process and justifying the chosen approaches.

Chapter 4 presents the research findings in relation to the previously defined research questions, with further analysis and code examples included in the appendix. Chapter 5 then interprets these results from both educational and scientific perspectives, highlighting their implications for improving teaching practices and contributing to the field of ML. Finally, Chapter 6 summarizes the key findings and limitations, and suggests areas for future research. The appendices and annex include additional results, side projects, and the practical work done using Google Colaboratory notebooks.

This chapter reviews the use of Artificial Intelligence (AI) in higher education, focusing on both recent developments and essential background knowledge. It starts by discussing AI's role in education, particularly through Educational Data Mining (EDM) and Learning Analytics (LA). The chapter then explains key AI methods, including Decision Trees, Artificial Neural Networks (ANNs), and other common techniques. It also examines data mining tools and their application to educational datasets. Next, it explores how knowledge is organized and managed, covering Knowledge Management Systems, Learning Management Systems (LMS), and the Knowledge Discovery Process, with a specific look at Moodle and its analytics features. This structure provides the necessary context to understand the methodology and results obtained within this project.

2

Literature Review

This chapter serves as a literature review, providing both foundational knowledge and insights into the state-of-the-art in educational technology and data-driven learning enhancement. It explores the integration of technology into education, emphasizing the role of Learning Management Systems (LMS) in modern pedagogy, with a particular focus on Moodle and its analytics capabilities. Additionally, the chapter introduces the knowledge discovery process, detailing how data extraction, processing, and mining techniques are applied to educational datasets. It also delves into the field of Educational Data Mining (EDM), highlighting its potential to improve teaching strategies and learning outcomes. To support the contextual understanding, the chapter includes an overview of relevant Artificial Intelligence (AI) algorithms used in EDM, offering a bridge between technical methodologies and their practical applications in educational contexts.

2.1 Artificial Intelligence in Higher Education

Artificial Intelligence (AI) has proven to be a driving force in the transformation of various sectors, including HE [4–6]. According to one of the founders, John McCarthy [7], AI is the science and engineering of creating intelligent machines, namely computer programs, and it is analogous to using computers to comprehend human thought processes, but it is not limited to biologically observable methods. However, his definition is just one of countless others, as can be observed in other publications [8, 9], that have gathered definitions of both AI and the concept of intelligence itself. Despite the absence of a universally agreed-upon definition, it is feasible to identify certain attributes that delineate AI: environment perception, including consideration of the real world's complexity; information processing (data collection and interpretation); decision-making (including reasoning and learning); performing actions and tasks with a certain level of autonomy; and achieving specific goals while being able to adapt and respond to changes in the environment [10].

AI systems, developed through Machine Learning (ML), logic- and knowledge-based

approaches, can generate outcomes like content, predictions, recommendations, or decisions that influence the environments they interact with [10]. Tasks such as visual perception, voice recognition, decision-making, and language translation, in addition to the ability to solve cognitive problems like learning from experiences, solving complex problems, and identifying patterns, demonstrate AI's diversity [10–12].

Figure 2.1 illustrates the classification of AI according to the definition provided by the European Commission Joint Research Centre in their publication from 2021 [8]. The classification of the domains and sub-domains of AI is divided into two distinct categories: "Core" and "Transversal". The "Core" domains relate to the essential elements of AI, while the "Transversal" domains involve the application of the core domains or their association with ethics and philosophy. The illustration further classifies ML based on the type of training and includes further information on the algorithm families of supervised, unsupervised, and reinforcement learning techniques at the bottom of the picture, highlighting the families that will be implemented in this project.

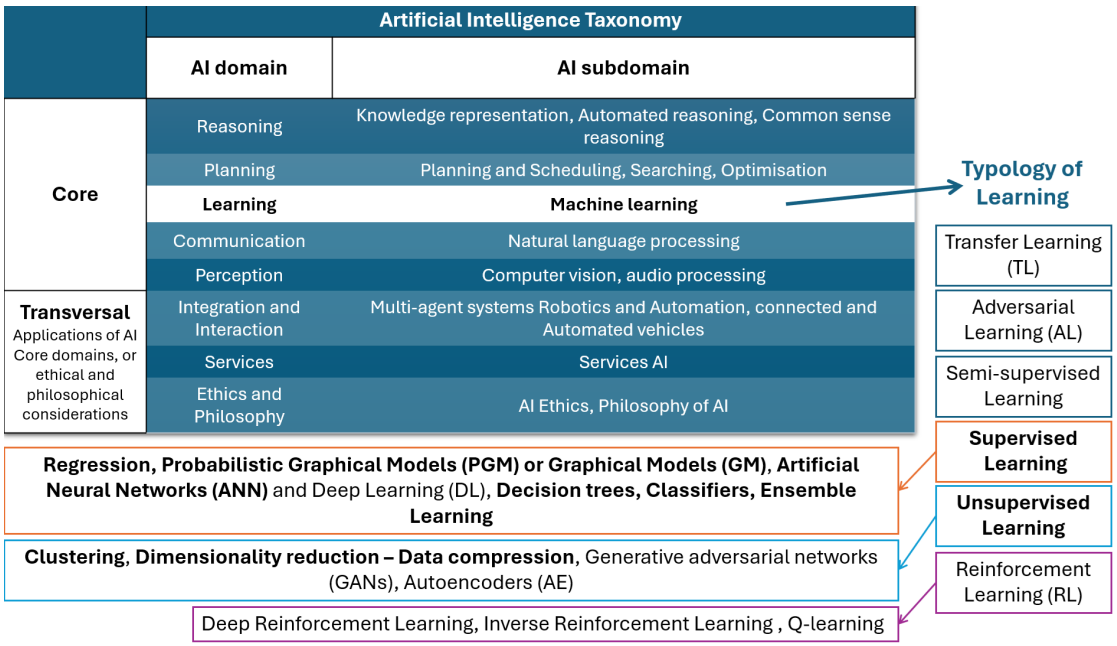


Figure 2.1: Taxonomy of artificial intelligence algorithms: domains and subdomains.

This taxonomy it's not absolute because it doesn't include all types of AI and doesn't have a hierarchical approach in terms of families (some classifications overlap). For instance, graphical models and neural networks can be trained as a supervised or unsupervised learning algorithm. Also, classifiers can be used for regression, and adversarial and transfer learning apply to all learning [8].

The sub-area of AI known as ML, is the scientific study of computer algorithms used to build models that improve automatically through experience, without being explicitly programmed to do so [8]. It has gained relevance in education, using advanced statistics to build structures capable of learning from available data, identifying patterns, and making predictions without human intervention [10, 13, 14], and it will be the sub-domain of AI

that is explicitly used on this dissertation project. This sub-domain of AI encompasses paradigms such as Reinforcement Learning (RL), concerned with how intelligent agents take actions in an environment to maximize the notion of cumulative reward; supervised ML, which uses labeled data sets to train algorithms to classify data or predict outcomes accurately; semi-supervised learning, considered a category between supervised and unsupervised learning, where the data contains both labeled and unlabelled data; and unsupervised ML, dealing with the process of inferring underlying hidden patterns from historical data [5, 8, 10]. Each of these types of learning has distinct applications and significant potential.

Figure 2.1 demonstrates that supervised learning includes a wide range of algorithmic groups. The Regression family encompasses various methods, such as Linear, Non-Linear, Gaussian Process Regression, Logistic, and Probabilistic Graphical Models, such as Restricted Boltzmann Machine (RBM), Deep Boltzmann Machine (DBM), and other algorithms, are essential in supervised learning tasks. Other notable individuals in this group are Decision Tree (DT). Classifiers such as Discriminant analysis, Support Vector Machine (SVM), and k-Nearest Neighbors (kNN), as well as Artificial Neural Networks (ANNs) [8]. Within the domain of HE, Deep Learning (DL), a specific branch of ML that heavily relies on ANNs, has emerged as a highly effective tool. It has exceptional ability in addressing complex cognitive issues, imitating human behavior, and facilitating the observation, learning, and adaptation of deep learning algorithms [5, 10, 15]. Moreover, unsupervised learning includes categories such as clustering, dimensionality reduction (which includes techniques like Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), Generative Adversarial Networks (GANs), and Autoencoder (AE)). Furthermore, many categories of reinforcement learning, such as Deep Reinforcement Learning and Inverse Reinforcement, are also worth mentioning. The process of learning, particularly through the use of Q-learning, has been important in solving problems that require making decisions in a certain order and planning for the long term [8].

AI, with its areas of computational vision, expert systems, multi-agent systems, and Natural Language Processing (NLP), plays a predominant role in HE, with applications in knowledge management, teaching personalization, and process automation [10, 12]. The fusion of AI and learning science resulted in Educational Artificial Intelligence (EAI), which aims to understand how learning is influenced by AI and how pedagogical, psychological, and sociological knowledge can be expressed through it [6]. Additionally, AI-based Knowledge Management Systems are revolutionizing HE, driven by the combination of ML and DL techniques [10, 16, 17].

2.1.1 State of the Art of Artificial Intelligence in Education

AI has been essential for the evolution of higher education, driving efficiency, personalization of learning, and new possibilities for teaching and collaboration, through technologies such as Neural Networks (NNs), Knowledge-Based Expert System (KBES),

and Case-Based Reasoning System (CBRS) [13, 17]. In different articles it's highlighted that AI not only improves technical aspects but also transforms the educational process as a whole, affecting content, teaching methods, assessment, and communication [11, 13].

AI plays various roles in education, from performing administrative tasks, and identifying students' learning styles and preferences, to predicting students' performance and early detection of learning difficulties [13]. Advances in AI applications also include Intelligent Tutoring Systems (ITSs), educational robots, and intelligent assessment. ML and Data Mining (DM) techniques, such as EDM, provide valuable insights into student progress, informing intelligent and adaptive interventions [6].

ITSs, operating as virtual teachers, have revolutionized higher education through their adaptability to individual students' needs [14]. Originating in the 60s and 70s, and structured around four main components - the student communication interface, the domain model, the student model, and the tutor model [13–15]. ITSs uses machine learning to personalize lessons, interact using natural language, and automatically generate exercises and assessments [14]. Despite challenges such as student knowledge representation and assessment, privacy issues, and costs [14], ITSs continue to evolve with the development of Adaptive Intelligent Tutoring Systems that employ advanced algorithms to monitor student progress and recommend appropriate resources [18].

To better understand the contributions of AI in education, it is useful to examine the specific scenarios in which these technologies are employed. For instance, in student assessment, adaptive learning methods and personalized teaching approaches are employed alongside academic data analysis. Exercise correction relies on image recognition, computer vision, and prediction systems. Intelligent and personalized teaching leverages data mining and inference through Bayesian knowledge, as well as intelligent teaching or tutoring systems and learning analytics. In the context of Smart Schools, facial recognition, voice recognition, virtual labs, Augmented Reality (A/R), Virtual Reality (V/R), and sensory technologies play essential roles. Similarly, remote education, whether online or mobile, benefits from edge computing, personalized virtual assistants, and real-time analytics. The utilization of these diverse artificial intelligence techniques underscores the multifaceted nature of AI's impact on education [13].

2.1.1.1 Educational Data Mining and Learning Analytics

Educational Data Mining (EDM), a branch of Data Mining, plays a significant role in analyzing large volumes of educational data, aiding in the assessment of student performance through Learning Management System (LMS). Interdisciplinary by nature, EDM extracts knowledge from data originated in educational contexts, capable of predicting student success, grouping students by similarities, and extracting relationships between variables [6, 19–22].

Learning Analytics (LA), in turn, is an emerging field that crosses education with data science, promising to enhance educational policies and practices [23]. LA is instrumental

in current learning design and benefits from the application of AI algorithms. For example, clustering modeling is a widely used unsupervised classification technique in LA. It is employed to forecast students who are at risk, determine the factors that influence their performance and results, and classify students based on their behaviour [24]. However, LA presents challenges, including the secure collection of real-time student interaction data, the integration of LA and Learning Design (LD), and issues of privacy and ethics [21, 25, 26].

Using EDM and LA can enhance student performance and aid in making informed educational decisions by offering important insights into students' characteristics and progress [22]. For institutions and educators to successfully use EDM and LA, they must have a comprehensive understanding of the underlying concepts and capabilities, as well as get acquainted with the essential tools and techniques involved [20–22].

Table 2.1 displays various ML methods employed in different applications within HE.

Table 2.1: Artificial Intelligence algorithms used in Education.

Applications in HE	Algorithms	References
Predicting Student Performance	RF	[20, 21]
Grouping Students	KMC	[20]
Analyzing Students Performance	SVM	[20]
Personalized Teaching	DT	[27]
Student Modeling	BN	[14]
Adapting Learning Content	NN	[14]

Algorithms employed in EDM not only includes Naive Bayes (NB), DT, rule-based algorithms, ANNs, kNN, and advanced forms such as Multilayer Perceptron (ANNs with multiple layers) [27]. These tools are leveraged for critical tasks like predicting student performance, with additional mention of REPTree (regression DT), OneR (One Rule for classification), Iterative Dichotomiser 3 (a DT algorithm), and PART (Partial DT) for their specific contributions in this domain. [28] also applied DT to predict student performance using data collected by an e-learning management system, utilizing several assessment and behavior parameters. Furthermore, the decision tree algorithm was combined using ensemble approaches such as boosting, bagging, stacking, and voting procedures to explore the possibility of improving prediction accuracy.

Based on a comprehensive study comparing various data mining methods on educational data [29], NB emerges as a consistent performer, particularly excelling in small sample sizes. NB exhibits higher accuracy rates and produces fewer erroneous estimates compared to other methods. Additionally, Logistic Regression (LR) showcases superior performance across diverse conditions, demonstrating lower error rates and higher

accuracy estimates, especially in medium and large sample sizes.

Additionally, [27] highlights the application of the field of Social Network Analysis and the algorithms JRip, along with KMC and ANNs, for the detection of undesirable student behaviors and the categorization of students into homogeneous groups.

While NB and LR methods demonstrate excellence in various scenarios, ANNs and kNN may yield different results depending on the dataset's characteristics. Furthermore, increasing the sample size generally enhances classification performance for all methods, emphasizing the importance of larger sample sizes for achieving high performance [29]. Therefore, while NB and LR methods are recommended for their overall superior performance, the selection of the most appropriate method should be based on careful consideration of the dataset's structure and analysis conditions [29].

2.1.2 Decision Trees and their Variations

One of the most prominent techniques is the DT, that represents a form of supervised machine learning that establishes rules derived from labeled data, forming a structure similar to a tree [28]. Hierarchical segmentation of the data facilitates the identification of common patterns, improving the accuracy of predictions. The primary goal of data splitting is to discover common behaviors in the dataset, which can then be used to assess prediction accuracy [10]. This approach builds and trains a classification tree made up of decision and leaf nodes using logical rules. Furthermore, leaf nodes represent paths that use logical values to reach a certain decision node [28]. Decision Tree family includes Classification or Binary Trees for discrete data, Regression Trees for continuous data, Random Decision Trees, Random Forests, Bagged Decision Trees, Boosted Decision Trees, and Behaviour Trees, among others [8].

DTs are quick, effective, and easy to comprehend and interpret [5]. Besides being extensively used, this algorithm is simple to design and low-cost to build and particularly good for simultaneous categorization and prediction [10, 28]. It can handle many data types and has fast computation, learning, and classification capabilities. However, DTs are inefficient at managing overfitting, noisy, and correlated data, they are inappropriate for regression applications, and have medium accuracy [5].

As a result of the evolution of this technique, ensemble methods such as RF have emerged, which aggregate multiple decision trees, minimizing variance and improving the accuracy of results [28]. To integrate the trees, this technique often employs an aggregation or voting approach with randomly generated trees. Another variation is Gradient Boosting Tree (GBT), which combine several "weak" trees into a single "strong" model, generates a predicted score through a progressive learning mechanism and it is useful for analytic training [28].

2.1.3 Artificial Neural Networks (ANNs)

ANNs are computational systems inspired by the biological brain, composed of nodes called artificial neurons, organized into layers: input, hidden, and output. This structure allows them to adapt and learn from data, being effective in detecting complex patterns. Moreover, according to [14], the use of ANNs in HE, specifically in ITSs, is particularly effective, thanks to their ability to link learner characteristics with topics, and update information on the screen and in educators' records while using the system. Despite their advantages,ANNs can be prone to overfitting if the network is too large and sensitive to noise in the training data [10].

By convention, the term ANNs is commonly used when we are referring to networks with up to four layers and Deep Neural Networks (DNNs) otherwise [8]. There are various types of neural networks, such as Convolutional Neural Networks (CNNs), suitable for image processing tasks, and Recurrent Neural Networks (RNNs), appropriate for sequential data. Beyond these, there are AEs (autoencoders), a type of unsupervised DNNs technique that reduces the dimensionality of the original input space to eliminate noise and irrelevant features. AEs consists of two parts: an encoder that maps input data to a lower-dimensional representation, and a decoder that maps the encoded representation back to the original input space. During training, the network learns to minimize the difference between the input data and the reconstructed output, adjusting the weights of the encoder and decoder [10]. Besides this, the DNNs also include DBM (Deep Boltzmann Machine) and Deep Belief Networks [8].

2.1.4 Other Important Techniques

Probabilistic Graphical Models encompass several techniques such as Hidden Markov Model (HMM) and BN, also known as Belief Networks or Causal Networks. These models also include NB, Semi-Naïve Bayes, and Pure Bayes [8]. HMMs are statistical models that find application in numerous domains, such as pattern recognition and computer vision [10]. This approach is especially valuable for investigations in which the temporal structure is essential. The efficacy of this model is heavily reliant on the quality and representativeness of the training data. Conversely, BN are graphical models that use probability to express a group of variables and their relationships, shown through a directed acyclic graph. This type of model is highly adaptable and robust, with the ability to represent intricate connections and dynamics between variables. This characteristic renders the family of algorithms highly valuable in various applications, such as the modeling of students in ITSs. It enables the recommendation of customized teaching techniques by integrating the content structure with the student's profile and learning style [14].

A specific instance of BN is the NB algorithm, which is a probabilistic classification model recognized for its high efficiency in handling big data sets. The technique is based on Bayes' Theorem. This method operates under the assumption that all input variables

are mutually exclusive, enabling predictions to be made based on the probability of each class [28]. Because of its adaptability and resilience, NB is widely used in different fields, including cybersecurity, where it performs well even in situations with limited data availability [10]. Nevertheless, it is important to acknowledge that the straightforwardness of the model might result in instances of overfitting, particularly when the assumption of variable independence is not accurate.

The kNN algorithm is a supervised classification strategy that assigns a new data point to a class based on its resemblance to the k nearest training points. The capacity of kNN to manage intricate decision boundaries and irregularities renders it particularly valuable in situations where linear models may be inadequate [28].

Cluster analysis is a technique that groups similar objects into clusters [30]. In the context of ITSs, clustering involves categorizing students based on their learning behavior and performance data, enabling more efficient customization of the teaching process [14]. KMC is an unsupervised learning method that categorizes data into a predetermined number of clusters, using the distance between data points as a basis [10]. However, this method assumes that all clusters have identical variance and are perfectly spherical, which may not hold true in practical scenarios.

Another clustering method is hierarchical clustering, visualized using a Dendrogram [30]. Unlike the Elbow Method commonly used for determining the optimal number of clusters in KMC, hierarchical clustering employs the Dendrogram, a tree-like structure that records the sequence of merges and splits, by choosing the farthest separation in clusters [30].

Fuzzy Logic (FL) is a decision-making methodology that aids in handling situations with uncertain information [14]. FL is extensively utilized in the development of the student model in ITSs, primarily to assess student's educational performance. This logic, in contrast to Boolean logic, encompasses a spectrum of intermediate possibilities rather than just true or false. This enables a more nuanced and accurate evaluation of student performance.

While we have emphasized some AI methodologies and procedures, it is fundamental to acknowledge the wide array of technologies that can be utilized in HE. By employing a suitable combination of these strategies, the learning environment may be enhanced and revolutionized, resulting in increased efficiency and engagement [17].

2.1.5 Data Mining Tools

Numerous tools are available for data mining and machine learning, each with its unique features and strengths. On Table 2.2 is a comparative summary highlighting the key features, strengths, and weaknesses of several popular tools.

Table 2.2: Comparative summary of Data Mining tools.

Tool	Key Features	Strengths	Weaknesses
RapidMiner	GUI, process-oriented workflows, comprehensive algorithms	User-friendly GUI, supports CSV and Excel, drag-and-drop	Proprietary license limits user base, free version limited [31]
Orange	Python-based, visual programming (Orange Canvas)	Visually appealing, user-friendly, structured functionalities	Limited widgets, lacks WEKA integration [31]
WEKA	Java-based, multiple interfaces (Explorer, Experimenter, Knowledge Flow)	Extensive algorithms, user-friendly, supports multiple formats	Inefficient with large datasets, limited big data/deep learning support [31–33]
R	Extensive packages, optimized for matrix calculations	Powerful for statisticians, Rattle GUI improves usability	Steep learning curve, complex command-line interface [31]
KNIME	Eclipse-based, visual programming	Extensive nodes, integrates with WEKA and R	Purely visual programming may not suit all users [31]
scikit-learn	Python-based, extensive machine learning algorithms	Comprehensive, strong community support, thorough documentation	Requires Python proficiency, lacks some algorithms [31, 34]
pandas	Data manipulation and preprocessing	Rich functionality, strong community support	Learning curve for complex operations [34]
Plotly, Seaborn, Matplotlib	Data visualization	Extensive capabilities (Plotly), intuitive (Seaborn), customizable (Matplotlib)	Different strengths, choice depends on project needs [34]

Continued on next page

Table 2.2: (continued)

Tool	Key Features	Strengths	Weaknesses
TensorFlow, Keras, PyTorch	Deep learning	TensorFlow for customization, Keras and PyTorch for prototyping	TensorFlow can be complex, Keras and PyTorch may be less versatile [34]
Hadoop Streaming, PySpark	Big data processing	Strong community support, integration with Hadoop and Spark	Requires understanding of big data ecosystems [34]

2.1.5.1 Detailed Descriptions

RapidMiner, a Java-based tool developed by RapidMiner, Germany, offers a user-friendly GUI and a comprehensive set of data mining algorithms and preprocessing tools [31]. It supports process-oriented workflows with a visual component system, allowing users to drag and drop operators to construct complex data flows. Although its transition to a proprietary license might limit its user base, the free Starter version provides sufficient functionality for many users, including support for CSV and Excel files [31].

Orange is a Python-based tool for data mining developed at the Bioinformatics Laboratory of the Faculty of Computer and Information Science at the University of Ljubljana [31]. It offers both Python scripting and visual programming through its interface, Orange Canvas. The visual programming interface provides a structured view of functionalities grouped into categories such as data operations, visualization, classification, regression, evaluation, and more. Users can program by placing widgets on a canvas and connecting their inputs and outputs, making it visually appealing and user-friendly. However, the number of available widgets is somewhat limited compared to other tools like RapidMiner and KNIME, and the lack of integration with WEKA reduces its extensibility [31].

WEKA, or Waikato Environment for Knowledge Analysis, developed at the University of Waikato, New Zealand, is another robust open-source data mining tool written in Java [31, 32, 35]. It offers multiple interfaces, including Explorer, Experimenter, and Knowledge Flow, with the Explorer being the most popular for defining data sources, data preparation, machine learning algorithms, and visualization [31, 32]. Data can be imported from several sources such as files, Uniform Resource Locators (URLs), and databases, and supports multiple formats for data import, including ARFF, CSV, LibSVM, and C4.5 [32]. Despite its user-friendliness and extensive collection of algorithms, WEKA has some limitations, such as less efficient handling of large datasets and limited support

for big data and deep learning methods [31, 33]. It also lacks adequate documentation and does not support Multi-Relational Data Mining (MRDM), which requires converting multiple linked tables into a single table format using external software [33]. Nonetheless, its integration with other tools and the availability of numerous extensions make it a valuable resource for data mining tasks [32].

R, a programming language and environment popular among statisticians, provides a powerful platform for data mining through its extensive collection of packages [31]. While R itself is optimized for matrix-based calculations and supports a wide range of data mining algorithms, its steep learning curve and command-line interface can be challenging for new users. The Rattle package, developed to provide a more user-friendly GUI similar to WEKA's Explorer, helps mitigate this issue by loading specific R packages for various analyses [31]. Despite these efforts, R's complexity remains a barrier for some users, although it remains a preferred choice for those proficient in its use [31].

KNIME (Konstanz Information Miner) is a general-purpose data mining tool based on the Eclipse platform, developed by KNIME.com AG in Switzerland [31]. KNIME uses a visual programming paradigm where users can place and connect nodes on a canvas to create workflows. With more than 1000 nodes available through core installation and various extensions, KNIME supports a wide range of data mining tasks. It also integrates well with other tools, such as WEKA and R, providing extensive functionality. KNIME is highly recommended for users who prefer a purely visual programming approach and need a variety of nodes for different data mining tasks [31].

The popularity of Python is growing, especially in the field of data science, leading to an increasing number of free libraries available for use [34]. Among these, scikit-learn stands out as a comprehensive machine learning library that extends the functionality of NumPy and SciPy with numerous algorithms [31]. It also leverages the matplotlib package for plotting charts. scikit-learn is well-regarded for its thorough online documentation and strong community support, including contributions from INRIA and Google Summer of Code. While it supports most core data mining algorithms, it currently lacks some significant groups like classification and association rules. Despite its advantages, scikit-learn requires proficiency in Python, which can be a barrier for some users [31].

For data preprocessing and manipulation, the pandas library is highly recommended due to their strong community support and rich functionality [34]. In the field of data visualization, the choice of the library depends on the project: Plotly offers extensive capabilities, seaborn is intuitive and easy to use, and Matplotlib provides many customization options. All three libraries have robust communities [34]. Deep learning is supported by libraries such as TensorFlow, Keras, and PyTorch. TensorFlow is preferred for projects requiring extensive customization, while PyTorch and Keras are suitable for quick prototyping [34]. For big data, Hadoop Streaming and PySpark are the best libraries, as they offer APIs for native libraries like Hadoop and Spark, ensuring large community support [34].

Overall, no single tool is best for all data mining tasks. Each tool has its strengths

and weaknesses, but RapidMiner, WEKA, R, Orange, scikit-learn, and KNIME offer most of the desired features for a comprehensive data mining platform [31]. Their varied functionalities make them suitable for different users and use cases, ensuring that there is a tool available for almost any data mining requirement.

To facilitate the implementation of machine learning and data mining tasks in this work, we will utilize Google Colaboratory (Colab) [36]. Colab allows users to write and execute Python code directly in their browser with zero configuration required and provides free access to GPUs, making it an ideal platform for intensive data processing and machine learning tasks. Colab comes pre-installed with many popular machine learning libraries, such as pandas [37], NumPy [38], Matplotlib [39], Seaborn [40], SciPy [41], and scikit-learn [42], reducing the need for manual installation and setup that is typically required when using Jupyter Notebooks.

In addition, Colab's interface with Google Drive provides effortless storage and sharing features, which are especially beneficial for collaborative projects such as this one, where resources are maintained in Google Drive. Collaborative work and project management are improved by the features of sharing notebooks, commenting on cells, and tracking modification history. Colab enables the inclusion of both code and text cells, facilitating the combination of executable code and documentation, similar to Jupyter Notebooks. Colab's ease of use and robust computing resources, along with its features, make it an invaluable tool for effectively performing and sharing data science and machine learning workflows.

2.2 From Data to Knowledge

In building a new society, developing a new approach to education is imperative, being intrinsically linked to knowledge. Thus, higher education institutions assume the role of facilitators and promoters of knowledge heritage, diffusion, transfer, and innovation. They encourage knowledge sharing, drive innovation, cultivate talents, and provide intellectual knowledge and support for societal development [43, 44].

The Data, Information, Knowledge, Wisdom (DIKW) pyramid, represented in Figure 2.2 and inspired by [19], illustrates the evolution of knowledge from raw, decontextualized data to wisdom. This process is facilitated by methodologies for transforming and contextualizing data until it becomes information and subsequently knowledge, evaluated for its value and relevance for decisions or actions [45, 46]. Wisdom, at the top of the pyramid, is seen as the ability to make correct judgments, acquired both individually and collectively, requiring effective management of data, information and knowledge [19]. Higher education institutions have a key role in the transition to a knowledge society, promoting the production and effective management of knowledge [47].

Understanding the nature of knowledge is a fundamental step in designing new educational approaches. The origin of knowledge begins with data, which are sets of discrete and objective facts about events. However, these data need to be transformed

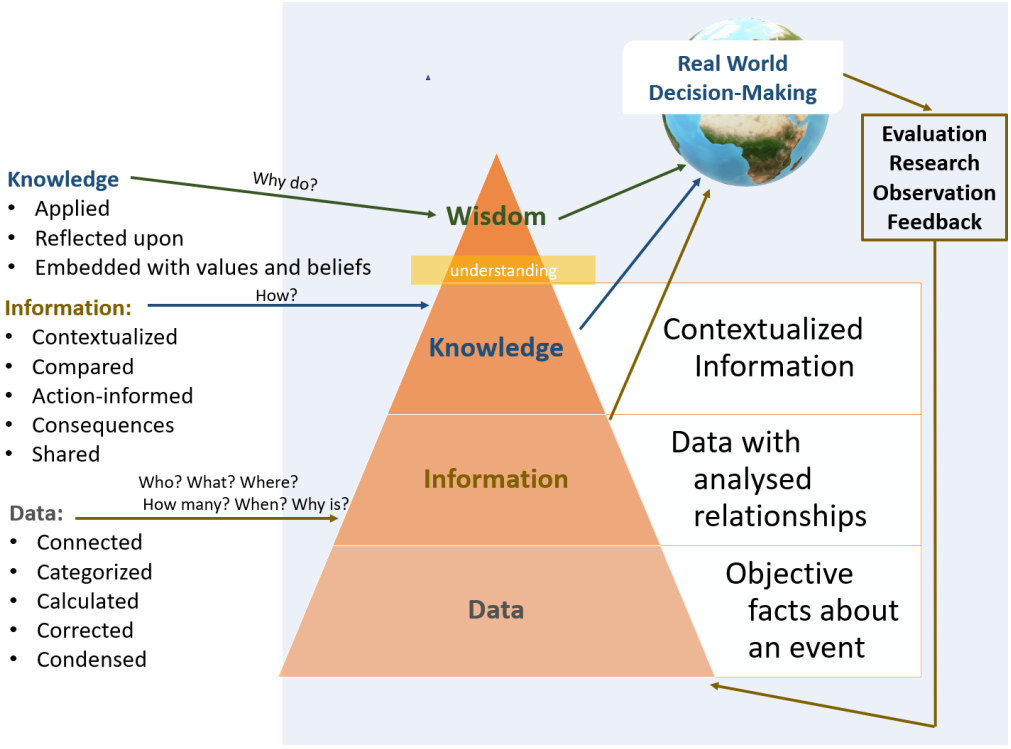


Figure 2.2: Data, Information, Knowledge and Wisdom (DIKW) pyramid.

to have practical value. Transforming data into information involves contextualisation, categorisation, calculation, correction, and condensation [45].

Information, resulting from the transformation of data, has a sender and a receiver, and exists to change something in the receiver, impacting their judgment and behaviour. This transformation process does not end at the information stage. Information can be converted into knowledge through comparison, analysis of consequences, establishing connections, and discussion [45]. In this context, Davenport and Prusak define knowledge as "a fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information" [45]. This framework is often embedded in routines, processes, practices, and organizational norms. However, [46] offer a complementary perspective, defining information as a flow of messages and knowledge as being created and organized by the flow of information itself, deeply tied to the commitment and beliefs of the knowledge holder. In the transformation of information into knowledge, it is important to consider that what is delivered is more important than the delivery vehicle. Knowledge should be evaluated by the decisions or actions taken based on it. Components of knowledge include experience, fundamental truth, complexity, judgment, intuition, and values and beliefs. According to [45], these values and beliefs play a important role, determining what the knower sees, absorbs, and concludes from their observations.

Given the complexity of classifying knowledge, the importance of Knowledge Management System (KMS) gains prominence. According to [48], this can be done based on

its source, form, and area. In terms of source, knowledge can be tacit, acquired through personal experience and subjective actions, or explicit, systematically codified and easily shared [16, 48]. Regarding form, knowledge can be classified as declarative, focused on facts and concepts (knowing what), or procedural, which involves the ability to perform tasks or activities (knowing how), often acquired through practice and experience. Moreover, knowledge can be general, widely accessible, or specific, restricted to a limited group of individuals [48].

The goal of this project is to get knowledge from the available data. Knowledge about the CTCT course and about the evolution and characteristics of the students of FCT. And, because CTCT is a course to empower students with some transversal skills, most the knowledge that is intend to be transmited to students is tacit and procedural, types that are hard to transmit and measure.

2.2.1 Knowledge Management Systems

Following the classification of knowledge, it's important to emphasize the relevance of KMS, which are tools designed to support the knowledge management processes. Knowledge management is a systematic process in which the knowledge needed for an organization to succeed is created, captured, shared, and leveraged. Thus, an effective KMS is more than just a simple tool or platform system for processing and regrouping knowledge. It involves a dynamic cycle that includes the management of knowledge as a subject, different types of valuable knowledge as objects, and the processes of utilizing knowledge [16, 49].

Various authors discuss the use of knowledge management practices, supported by ICT (Information and Communication Technology), in educational contexts, focusing on practices such as the creation of knowledge communities and the use of repositories for best practices concerning the processes performed [17, 47, 50]. Challenges are faced, such as the need for appropriate management and a favorable organizational culture [17]. [47] also emphasizes the importance of various tools in knowledge management, like knowledge repositories, workflow tools, and ontologies.

A KMS should be capable of handling different types of knowledge - tacit and explicit - and support their transformation and interaction. The Socialization, Externalization, Combination and Internalization (SECI) model is a conceptual tool often used to describe this process. Socialization refers to the conversion of tacit knowledge into new tacit knowledge, Externalization to the conversion of tacit knowledge into new explicit knowledge, Combination involves the creation of explicit knowledge by merging, categorizing, reclassifying, and synthesizing existing explicit knowledge, and Internalization is the conversion of explicit knowledge into new tacit knowledge [43, 46, 49].

Besides the SECI Model, the Inukshuk Model is a significant tool in knowledge management. This model, an extension of the SECI model, adds Leadership, Culture, and Technology as the foundations of the four stages of the SECI model. The model

highlights the need for effective measurements during the implementation of a KMS, the importance of leadership that shares decisions based on explicit knowledge, the construction of a knowledge-sharing culture, and the appropriate use of technology. Additionally, [43] suggests an expansion of the Inukshuk model that adds "People" and "Policies" as fundamental pillars, assigning to the organization members responsibilities to initiate knowledge management projects and implement rules that encourage knowledge sharing.

AI Systems have the potential to enhance the effectiveness of KMS, especially in the classification and management of knowledge, but it's important to note that these systems have limitations, particularly in the management of tacit knowledge [16, 49].

A knowledge management system can be divided into four main processes: knowledge discovery systems, which include the transfer and conversion of knowledge; knowledge capture systems; knowledge sharing systems; and knowledge application systems. These processes ensure that knowledge is discovered, captured, shared, and applied effectively for the benefit of the organization [43]. This project consists on using statistics and ML to discover knowledge from data, that include the available datasets, the course content and observations of CTCT classes.

2.2.2 Learning Management Systems

LMS have become essential tools in contemporary education, driven by the knowledge society's emphasis on efficient knowledge management within organizations, especially in higher education. These systems represent significant advancements in how organizations process and share information and knowledge, evolving from Management Information System (MIS) and Decision Support System (DSS) to current KMS and LMS [50].

LMS are online platforms that capture and store all user interactions in log files and are designed to offer higher education and continuing education to adult students, regardless of their geographical location [24, 51]. Data can be exported, processed, and analysed to acquire novel insights, leading to the emergence of disciplines like EDM and LA [24].

With features that encompass the creation, management, and distribution of learning materials, as well as the administration of classes and users, LMS facilitate students' interest, enthusiasm, and dedication, promoting the exchange of knowledge between instructors and students [51]. These systems allow the completion and submission of assignments for assessment and include many learning tools, including document management, tasks, quizzes, and facilitate communication between students and teachers through forums, messaging, and wikis [24].

LMS, such as Blackboard, Moodle, and Canvas, are widely adopted in higher education institutions worldwide [51]. These platforms play a significant role in facilitating learning and the relationship between teachers and students. While Blackboard is known for its user-friendly interface but limitations in web applications, Moodle values student participation and the creation of learning communities [51]. Canvas stands out for its intuitiveness

and advanced features, being widely preferred by many institutions in the United States of America (USA). These LMS platforms have been essential for transforming higher education, making the teaching and learning experience more accessible and comfortable for students [51].

The adaptability of LMS, evidenced by their ability to integrate with other systems and to be customized to optimize the learning environment, is transforming pedagogy and redefining the expectations of learners [52, 53]. The global pandemic in 2019 accelerated the use of LMS in distance education, where, when well implemented, they can serve as the core of virtual universities, making learning resources instantly available and allowing immediate feedback to students [52]. However, the efficacy of LMS is dependent on the digital literacy of users.

In higher education, a diverse array of tools and platforms are employed to gather, organize, and present data for LA purposes. These tools encompass LMS like Blackboard, Moodle Plugins, and specialized platforms such as Learning Analytics for Prediction and Action, Social Network Analytics and Pedagogical Practices (SNAPP), Smart Class LMS, and GradeCraft Gamified [30]. Grades, time, and historical performance data are used to predict learning outcomes at one university to improve retention. Another school lets students compare their performance to peers using a Blackboard LMS feedback tool for self-awareness. Another one tracks course access, engagement, and assignment scores to evaluate student progress. Another education institution referred by [30] uses the "Gamified Grade Book GradeCraft," which engages students and teachers with visualization and information." Finally, Capella University's "Competency Map Interface", which graphically shows the progress made and expected, helps students track their academic development.

Despite their widespread application, LMS and KMS platforms still face various challenges and limitations. The ability to customize is often limited, which can hinder the adaptation of these platforms to meet specific teaching and learning needs. Moreover, many of these systems were not designed with the use of AI in mind, limiting their potential to optimize knowledge management. The lack of proper integration between different systems and the excess of complexity are also persistent issues.

Moodle and Learning Analytics

Moodle, the LMS adopted by FCT, currently in version 4.1, serves as a repository of data for LA. The data in Moodle is thoroughly documented in the Activity Log, easily accessible by teachers without requiring server access or technical ability. Furthermore, this data can be effortlessly exported for use in different software applications [24]. Exploring the functionalities of this LMS is necessary for suggesting improvements to future work in this dissertation and facilitating direct data collection from the system.

With approximately 20 different types of activities available, such as forums, glossaries, wikis, assignments, quizzes, Shareable Content Object Reference Model (SCORM) players,

and databases, Moodle offers extensive customization options. Its activity-based model empowers educators to combine activities into sequences and groups, guiding participants through learning paths and enabling each activity to build on the outcomes of previous ones [54].

In addition to these functionalities, Moodle can be personalized with plugins. For instance, MILA is an interactive LA plugin built by researchers, capable of analyzing regular logs and SCORM tracking data. MILA's effectiveness in processing Moodle logs and SCORM tracking data, particularly related to course video lessons, has been verified through thorough testing, providing valuable insights on courses, learning resources, and students' behavior [55].

A literature review of Moodle learning analytics and visualization tools from 2010 to 2020 revealed that most tools aim to compare student metrics and are designed for teachers. Additionally, there is a growing interest in prediction and intervention methods. However, the LA Dashboard's collection, processing, and storage of personal data lack ethical information [56].

Despite its widespread adoption, open-source nature, and robust feature set, Moodle's complexity may pose challenges for non-technical users, and installation can be daunting for novice technicians due to technical terminology. Effective use often requires skilled course administrators to collaborate with teachers and technicians, while support forums predominantly in English may limit accessibility for non-English speakers. Nevertheless, Moodle's affordability, flexibility, and strong pedagogical foundation continue to democratize Virtual Leaning Environment technology, making it accessible to institutions with limited resources [57].

For further exploration of Moodle's capabilities, refer to `http://moodle.org`.

2.2.3 Knowledge Discovery Process

The process of EDM involves a multi-stage process of Knowledge Discovery that extends from data extraction to detailed dataset analysis with the aid of external tools. This intricate process, as described by [22], encompasses a sequence of operations including data selection, preprocessing, transformation, mining, evaluation of patterns, and the final representation of knowledge. Complementing this perspective, De Menezes et al. [20] delve into the EDM process through the lens of their study on the interactive learning system LEIA, highlighting a slightly nuanced architecture for knowledge discovery. In their work, EDM corresponds to the mining phase referred to by Ampadu [22], and preprocessing should include a data cleaning stage (that remove noise and inconsistent data), a data integration stage (where numerous data sources can be merged), data selection stage (which, in Ampadu's approach [22], occurs before the preprocessing stage), and data transformation stage (the process of transforming and consolidating data into forms that may be mined, summarized, or aggregated). This dual insight into EDM's methodology reveals the field's depth, showcasing the varying but complementary

approaches to harnessing educational data for insightful analysis.

These stages of the knowledge discovery process were applied in various analyses, such as identifying learning behaviors in programming classes and developing predictive analyses to estimate the final average of engineering students. Various models and algorithms were utilized. For example, the RF (Random Forest) model was used to identify influential factors, such as the pace of learning in synchronous and asynchronous activities. The KMC (K-means Clustering) model was utilized to categorize students into specific groups, while SVM (Support Vector Machine) classification models, along with logistic regression, were applied in analyzing the performance of engineering students [20].

In the context of educational data mining, data selection involves identifying and gathering relevant data sources, which may include student grades, online activity logs, questionnaire responses, and participation in discussion forums [20, 22]. However, for this project, the phase of data selection is bypassed due to the reliance on available datasets. Access to Moodle data being unavailable, the project is constrained to Excel datasets provided by the course coordination for specific academic years. Additionally, questionnaire results for the current academic year and course materials are accessible, as the author served as a course monitor.

Following data selection, the next stage, data preprocessing, involves cleaning, integrating, and compressing collected data to address its voluminous, duplicative, and uncertain characteristics [58]. This phase includes rectifying missing values, eliminating duplicates, identifying irrelevant segments, and addressing noisy data. Techniques such as statistical refinement, orchestration, virtualization, and blending operations are employed to refine the data, although challenges persist in managing various data types, erroneous and noisy data, and data duplication.

Once preprocessed, the data undergoes transformation to convert raw data into a format suitable for extraction [20, 22]. This transformation may involve converting textual data into numeric representations, facilitating subsequent analysis.

Data mining encompasses employing various extraction techniques, including classification, regression, clustering, and association, to find patterns within the data [20, 22]. Commonly utilized techniques such as RF, KMC, SVM, LR, and BN have been highlighted for their success across various studies [20–22].

The phase after data mining, the interpretation and evaluation of patterns involves assessing the quality of models or patterns created or discovered. While techniques such as cross-validation and evaluation metrics like Mean Squared Error (MSE) and R-Squared coefficient are commonly used for evaluation [59], the field offers a diverse array of evaluation metrics tailored to specific data mining techniques.

For instance, in clustering techniques, various evaluation metrics provide insights into the quality and performance of formed clusters. Internal evaluation metrics, such as the Dunn index, assess the quality of clusters by considering both intracluster diameter and intercluster distance. Similarly, the Davies-Bouldin Index (DBI) evaluates clustering performance by minimizing the average distance between each cluster and its most similar

cluster. The Silhouette coefficient provides insights into the distribution of objects within clusters by calculating silhouette widths for each data point [60].

Externally, when true labels are available, metrics like the Rand index measure agreement between class labels and formed clusters. The Adjusted Rand index quantifies the algorithm's correctness percentage, adjusting for chance, while the Jaccard index measures object similarity between clustered and true labels. The F-measure computes the harmonic mean of precision and recall, and the Fowlkes-Mallows index gauges the similarity between two clusters. Additionally, the Mutual Information index quantifies label similarity within the same dataset [60].

Normalized metrics like the Normalized Mutual Information index, which scales results between 0 and 1, and completeness and purity indices, offer further insights into clustering quality and completeness. This diverse range of evaluation metrics provides researchers and practitioners with valuable tools to assess clustering techniques across different contexts and datasets [60].

Typically, creating the model does not mark the conclusion of the endeavor. Even if the model's purpose is to increase understanding of the data, the knowledge acquired must be structured and presented in a way that the client can use [61]. This last stage of this process, known as Knowledge Representation, may include visualizations, reports, or integration into specific systems, depending on the goals [20, 22, 61]. Platforms like Highcharts, Tableau, Hierarchical Clustering, and Sunburst Visualization offer intuitive ways to visualize complex educational data [30].

Educational data mining necessitates the utilization of several methodologies and process models to uncover knowledge. The dissertation in Computer Engineering project [62] examines several data mining models and approaches, providing a comprehensive understanding of various information discovery methods and their advantages and disadvantages. It covers more than just the information discovery process described in this section and the data mining framework. The appropriate framework is contingent upon the specific project, so in this work, it will be used an adaptation of the CRoss-Industry Standard Process Model for Data Mining (CRISP-DM) framework. It is a well-known data mining process model that provides a structured approach for machine learning that consists of three hierarchical levels and serves as a helpful guide [61].

Figure 2.3 illustrates how the conventional CRISP-DM process has been modified, for this research. Unlike the traditional approach, which only displayed data flow from data understanding to data preparation, this modification shows a clear arrow indicating bidirectional process flow between these two steps of this data mining methodology, but the data mining reference model itself [61] also advises about data preparation has to be made many times along the data mining project life cycle.

The first phase of Business Understanding centers around comprehending the project's requirements and objectives from a business standpoint. Using this information, a data mining problem is defined and a preliminary plan is created to accomplish the goals and it involves collecting data and taking actions to familiarize oneself with the data, identify

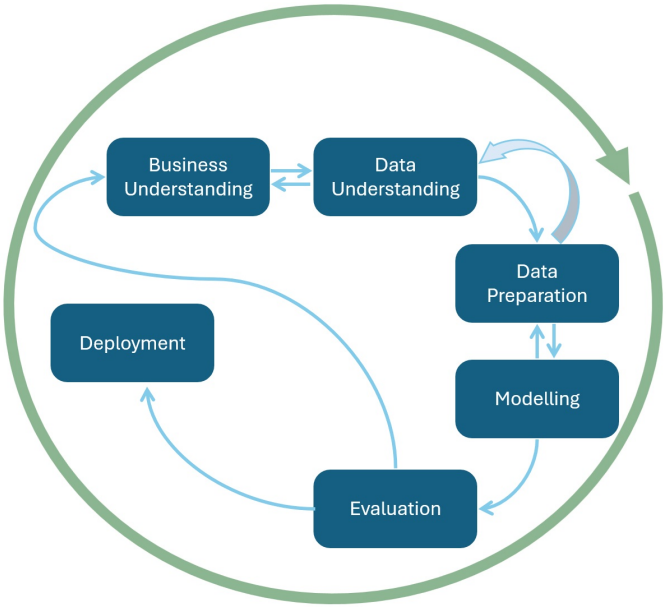


Figure 2.3: Data Mining lifecycle diagram illustrating the phases of the CRISP-DM Reference Model.

any issues with data quality, gain initial insights, and identify interesting subsets of data that can be used to develop hypotheses about hidden information [61].

The data preparation phase encompasses the necessary steps to transform the original raw data into the final dataset that will be used as input for the modeling tool(s). Data preparation methods are prone to being performed several times and in a non-prescribed sequence. The responsibilities encompass the selection of tables, records, and attributes, as well as the processing and cleansing of data for modeling tools [61].

During the modeling phase, various modeling approaches are chosen and implemented, with their parameters adjusted to maximize performance. There are typically multiple approaches available for each specific data mining assignment. Certain methodologies have distinct prerequisites on the format of the data. Consequently, the data preparation procedure is frequently revisited [61].

During the evaluation phase of the project, you have created one or more models that seem to be of excellent quality, as determined by the analysis of data. Prior to finalizing the deployment of the model, it is essential to conduct a comprehensive evaluation of both the model itself and the methodologies employed in its creation, in order to verify its alignment with the business objectives. The main objective is to ascertain if an important business matter has not been adequately addressed. The conclusion of this process should involve a decision on the utilization of the outcomes obtained from data mining [61].

Usually, the completion of the project does not occur when the model is created. While the primary goal of the model is to enhance comprehension of the material, it is essential that the acquired knowledge be organized and communicated in a manner that is practical for the client to utilize. The complexity of the deployment step might vary depending on the objectives, ranging from a straightforward report creation to the establishment of a

periodic data mining process. Typically, the customer will be responsible for managing the deployment processes, rather than the data analyst. Consumers must have a clear understanding of the essential tasks required to effectively utilize the models that have been created [61].

3

Methodology

The methodology adopted in this research is presented in this chapter, giving special emphasis on data understanding and data preparation steps of the CRoss-Industry Standard Process Model for Data Mining (CRISP-DM). This chapter starts by providing an overview of the research design and the data mining approach that was adopted, followed by an explanation of the reasoning behind the choice of development environment and programming language used for this project. The project's phases—data collection, data preparation, and exploratory analysis—are then covered in more detail; however, Appendix A contains a more thorough explanation of these procedures. The chosen Machine Learning (ML) algorithms and assessment techniques used to address the clustering and prediction study questions (RQ4 and RQ5, respectively) are described at the end of this chapter.

3.1 Overview of Research Design

The CRISP-DM methodology was employed in the research, which covers six stages described in Chapter 2, Subsection 2.2.3.

Figure 3.1 depicts the first three stages of the adopted methodology applied to this specific research, showing not only the return to data preparation when issues were found with the data but also some of what was done in the first year of the project and the additional tasks that were done to understand the CTCT course, corresponding to the business understanding stage of the CRISP-DM methodology. These auxiliary projects can be seen in more detail in Annex I, and include both the development of a web application with frontend in VueJS and backend using Django and Visual Basic for Applications (VBA) code development for automatic correction of CTCT Excel activities.

Figure 3.2 illustrates the methodology used in this project for the modeling, assessment, and deployment phases of the CRISP-DM methodology. Using this method, the final two research questions were addressed: What distinct patterns in student performance can be identified based on assessment results in the CTCT discipline, and can student outcomes in later themes or weeks of the CTCT course be predicted based on early performance in

a given topic or week? Because the data are simple, the first three research questions can be easily answered with a few visualizations and statistical tests.

Software and Tools Used

Google Colaboratory was selected based on its strong organizational features and Python integration. It has various required libraries loaded, interacts directly with Google Drive, and permits dynamic alterations to studies depending on real-time results. Python 3.10 was chosen as the language of choice because of its broad library support for machine learning and data processing, which includes tools like Pandas for data cleansing and Scikit-learn for data manipulation, especially missing value imputation. In the early stages of this project, Python was selected for these libraries since the plan was to create a web application using Django to gather the data that would be analyzed (Django is a Python framework).

3.2 Data Collection

The project's original plan was to employ a web application to gather data, but as Figure 3.1 illustrates, tests did not yield the desired outcomes. As a result, the study was reorganized, with an emphasis on the seven datasets that were supplied in Microsoft Excel Worksheet file (.xlsx) format and contained student evaluations from CTCT courses across seven academic years. This required addressing common inconsistencies, such as differences in column names and the occurrence of blank rows and columns. Given by CTCT coordination, two batches of datasets with the properties shown in Table A.1 were received; nevertheless, Section A.1 provides further information on the data used.

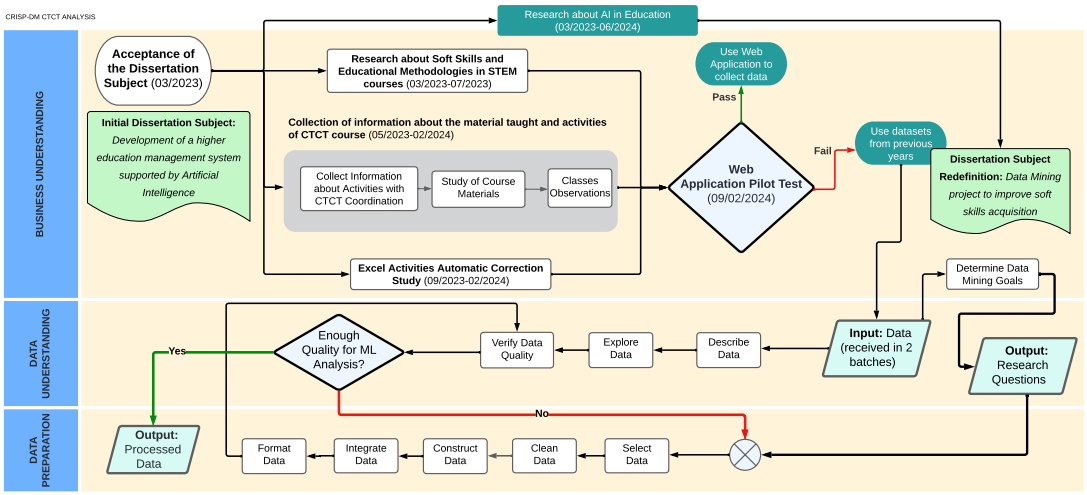


Figure 3.1: CRISP-DM project flow diagram: from business understanding to data preparation.

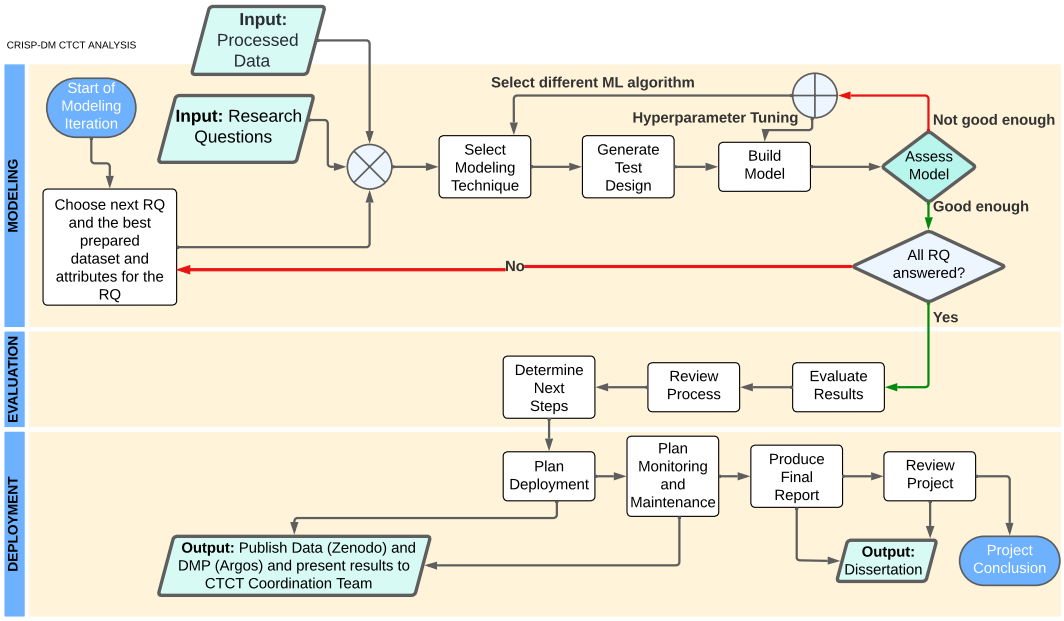


Figure 3.2: CRISP-DM project flow diagram: from modeling to deployment.

The datasets were first qualitatively assessed as part of the research process. This involves looking into the properties in the Excel files and running Python functions to understand the typology of the data. For instance, the `Numero` column in the original dataset was created by combining the year of the dataset with a sequential number that had three or four digits, depending on the dataset. This unique identifier within each dataset ensures the confidentiality of student records within that specific dataset. However, since there are more than 1000 students in each dataset, datasets with three identifying digits cannot include just one year, meaning that if multiple years were combined, some information would be duplicated. It does have another disadvantage: if a student retakes CTCT, their appearance in many datasets will be recorded without their repetition being recognized. Additionally, to publish the data internationally, the datasets' categorical attributes—which were originally defined as objects—had to be converted to the proper Python data types and had to have their names changed to English. Additionally, the data frame index was created by first converting the column `Numero` to `ID`, as shown in line 22 of Code Snippet A.1.

Besides the several numerical data (most of them integers within a range), the data included five categorical features, of which only two will be analyzed: `Sex` and `Course`. It was decided not to use the data regarding `Commuting`, `SS`, and `freq` columns. The first one, `Commuting`, an integer value in a Linkert scale to assess the time it takes for the student to go to its main home, was discarded because it didn't exist in all years and the categories were not consistent along the several editions. The second excluded categorical attribute, `SS`, was discarded because it has a lot of missing data (most students don't have special status) and the categories are specific for the year of the dataset. The third, `freq` column,

was excluded because it didn't give much information, indicating only if the student attendance is enough, not enough, or absent and is not available for one of the years (but could be analyzed using participation points data). All students that failed CTCT either had not enough or absent attendance.

After the receiving the second batch of data that included changes in the previously processed datasets and to streamline this process and enhance efficiency, the scripts were reorganized into two primary notebooks—one for data preparation and another for data understanding, which is described in the next two sections of this chapter and complemented with the content of Appendix A. Before the second batch it was used two notebooks for each received file.

3.3 Data Preparation

The data preparation phase focused on organizing and setting up the data for analysis to ensure that the datasets could be used for subsequent studies. Iteratively, when new insights were discovered, data transformation tasks were regularly reviewed during this period. Table 3.1 describes the key steps involved in the initial phase of preparing the data for analysis, but for a more detailed description of the data preparation steps, refer to Section A.2 of Appendix A.

Table 3.1: Steps in developed Data Preparation notebook.

Step	Description
Importing Libraries	Necessary libraries are imported for data manipulation, numerical computing, and visualization.
Dataset Parameters Definition	Parameters for each dataset are defined, such as filenames, rows, columns to skip, and specific columns like commute information and grade values.
Data Cleaning	Missing values are verified and categorical data are corrected to ensure data quality.
Generating Lists	Lists of features are generated to separate types of features from the dataset, such as categorical and asynchronous features.
Identifying and Handling Missing Values	Integer missing values were filled with zero and it was identified two columns that remained with missing values: `FinalGrade`, `FinalGradeInteger`, `Commuting` and SS (special status).

Continued on next page

Table 3.1: (continued)

Step	Description
Feature Engineering	Created columns with a total of points per type (bonification, self and peer assessment, participation, and activity points) and AP and PP totals per week.
Handling Outliers	Outliers on totals were analyzed at the 1st and 99th percentiles, but it only identified zero outliers that were not outliers but students that failed.
Verifying and Correcting Activities	Activities are verified and corrected based on their types of evaluation to ensure all columns are accurately characterized.
Combining and Standardizing Columns	Course names that have changed over the years are combined and columns like 'Sex' are changed categories to 'Female' and 'Male' instead of '0' and '1'.
Updating Column Names	Columns in datasets are updated based on an activity dictionary, mapping old names to new, standardized names.
Updating Lists	Created a copy all_lists dictionary with new column names for each year with common ID for all datasets.
Combining DataFrames	DataFrames for each year are combined into a single DataFrame, adding a 'Year' column to differentiate the data.
Removing Duplicate Columns	Duplicate columns in datasets combined from several years are removed to ensure each column is unique.
Splitting Datasets by Missing Values	The combined data frame is split into multiple datasets based on the number of missing values in columns, creating separate datasets for different missing value counts.
Saving Processed Data	Processed datasets and lists are saved in Feather, CSV, and Pickle formats, and output folders are zipped for download.

The Jupyter notebook `data_preparation` was created with two purposes in mind: first, it prepares the data for use in all other data analyses; second, it prepares the data to be saved in the appropriate formats for use in other scenarios, like publishing open access to the data and sending the results to CTCT coordination. Consequently, the usefulness of the data was determined early on, and distinct datasets were created for each of the available years in addition to multiple datasets with a mix of years (in which a column indicates which year a given student record belongs to) and several dictionaries that contained details about various aspects of the datasets.

The data preparation phase of this project uses a variety of file formats to ensure that

the processed data is usable and accessible on a range of platforms and for a variety of user requirements. The Feather format is utilized to expedite data transmission between Python environments and enhance the efficiency of subsequent data processing operations on the processed datasets in this project. The most common type of data file, CSV files, hold less data than feather files, which are a faster, lighter, and more user-friendly binary format for storing data frames. The data was published in CSV format due to its wide compatibility with various software applications, databases, spreadsheets, and data analysis tools. Excel files are available for stakeholders, including the CTCT coordination team, who may prefer a more interactive and user-friendly interface.

After the import of some Python libraries described in more detail in Section A.2, it was declared the main dictionary used to characterize the several datasets, `dataset_params`. It was created after a thorough analysis of the features (type of evaluation, if it's written or oral activity, group or individual, homework or class activity, just for some students or for all, and thematic) of all the activities of each of the analyzed CTCT editions, as explained in further detail in Subsection A.2.

Updating column names and lists is an important phase in the data preparation process that guarantees consistency between datasets. The activity dictionary maps old column names to the activity's unique `ID` (defined on the referred supplementary Excel file developed that is shown in Figure A.1) and it is specifically used to update columns in the datasets. To make sure that every dataset had a unique ID, a duplicate of the `all_lists` dictionary was then made, with new column names for every year. The data frames for each year were then merged into a single data frame and differentiated by an extra 'Year' column.

Duplicate columns were eliminated from the merged datasets to assure uniqueness in order to efficiently manage missing values. After that, distinct datasets covering different missing value counts were created by dividing the consolidated data frame into individual datasets according to the number of missing values in the columns. This approach ensured the quality of data and improved the datasets for a variety of studies. Consult Section A.2 of Appendix A for an expanded description of the data preparation procedures, including specific methods and operations employed.

Feature Selection

In the context of addressing RQ4 and RQ5, feature selection is important to identify the most relevant attributes for predicting student performance and clustering student groups based on performance metrics such as `FinalGrade`, `Total_AP`, `Total_PP`, and `Total_SPA` (final grade and the total of activity, participation and self and peer assessment points). Due to the dual nature of these targets—some being continuous (e.g., `FinalGrade`) and others categorical (e.g., `Passed`)—both regression and classification models were employed. Regression models were necessary to predict continuous outcomes, allowing for detailed predictions on student grades and performance points. In contrast, classification models

were used for categorical predictions, such as determining whether a student would pass or fail the course.

The selection of feature selection techniques and algorithms was driven by the need to balance model interpretability with predictive power. Techniques such as Random Forest (RF) were chosen for their ability to handle large datasets and provide robust feature importance scores, while Least Absolute Shrinkage and Selection Operator (LASSO) was selected for its capacity to enforce sparsity in the model by penalizing less significant features, thereby enhancing model simplicity and generalizability. The feature selection process was conducted using a combination of methods to ensure a robust and comprehensive evaluation of feature relevance, implemented in different phases of this project and across different notebooks. The techniques employed in this study are summarized in Table 3.2.

Table 3.2: Summary of feature selection techniques.

Technique	Description
Variance Threshold	This method was applied to filter out features with low variance, under the assumption that such features do not contribute significantly to the model's predictive power. By setting a predefined threshold, features whose variance fell below this level were removed, simplifying the dataset and reducing noise.
MI	Mutual Information (MI) was utilized to assess the dependency between features and the target variable, capturing both linear and nonlinear relationships. Features with higher MI scores were deemed more informative, offering deeper insights into which attributes most strongly correlate with student performance metrics.
RF	Random Forest (RF) was chosen due to its built-in feature importance mechanism, which calculates the contribution of each feature to the model's overall accuracy. This method is particularly effective with large, complex datasets and is resilient against overfitting due to its ensemble nature.
DT	Decision Tree (DT) was used for its simplicity and efficiency in providing feature importance scores. While less complex than RF, DTs are effective for preliminary investigations and provide a clear hierarchical structure of feature importance.
SelectKBest	The `SelectKBest` method with F-regression (FR) as the scoring function was employed to select features with the strongest linear relationships to the target variable in regression problems.

Continued on next page

Table 3.2: (continued)

Technique	Description
LASSO	Least Absolute Shrinkage Selection Operator (LASSO) was applied to perform feature selection by penalizing the absolute size of the regression coefficients. This technique is particularly effective in models with a large number of features, as it forces some coefficients to zero, effectively selecting a subset of relevant features and enhancing the model's simplicity and generalizability.
PCA	Principal Component Analysis (PCA) was employed to reduce the dimensionality of the dataset while retaining the most important features that explain the majority of the variance in the data. This technique is essential for simplifying complex datasets and ensuring that the most informative components are utilized in the modeling process.

In addition to selecting features for prediction using the strategies outlined in Table 3.2, features for the clustering analysis were also selected. On the dataset of each year and the combined dataset with no missing data, clustering was done using three different sets of attributes: the first set corresponding to the total value of each type of assessment point (AP, PP, SPA, and bonification points); the second set was used to perform clustering analysis using the classification of each activity of the first two weeks; the third set was used to perform clustering with the classification of every activity, using Principal Component Analysis (PCA).

The feature selection was implemented in the `Feature_selection_and_clustering` Jupyter Notebook file, developed using Google Colaboratory, and can be analyzed in more detail in Subsection B.1. This approach ensured that the models were trained on data that genuinely influenced student outcomes, thereby enhancing the predictive performance and validity of the educational interventions proposed.

3.4 Exploratory Data Analysis

After the data was prepared, an extensive study was performed to determine its characteristics and quality. This stage involved statistical analyses and visualization techniques to identify patterns and anomalies that could impact the outcome of the project, as shown with detail in Section A.3 of Appendix A.

The notebook performs the data understanding phase based on the CRISP-DM methodology. Initially, libraries for data manipulation, visualization, and statistical analysis were

imported. Datasets from the academic years 2015/16 to 2023/24 were loaded in various formats. Descriptive statistics were generated to understand the data distribution and variable characteristics. The data was then segregated into groups of students who passed and failed to identify factors influencing performance.

Heatmaps were used to conduct correlation analysis and uncover correlations between attributes. To initially investigate the data, feature selection approaches such as Variance Threshold and MI were used. Hierarchical clustering was used to investigate the data structure and group-related variables. To validate assumptions, other statistical tests were used, such as the Mann-Whitney U, Chi-squared, and Shapiro-Wilk tests. The results of these analyses were presented systematically to ensure documentation and interpretation.

Several hypothesis tests were performed to validate assumptions and understand the relationships within the data. The Mann-Whitney U test was used to compare the distributions of grades between two independent groups, such as male and female students. This non-parametric test was chosen because it does not assume a normal distribution of the data, making it suitable for the grade distributions observed. The Chi-squared test was applied to examine the independence between categorical variables, such as gender and course enrollment, to identify if certain courses were preferred by different genders.

Additionally, the Shapiro-Wilk test was employed to assess the normality of the data distributions, particularly for continuous variables like grades and various assessment points. Normality tests are used to determine whether parametric tests are appropriate for the data. By conducting these hypothesis tests, it ensured the robustness of this analysis and it was provided a solid foundation for interpreting the relationships within the data. These tests also helped in answering the first three research questions: RQ1 on how gender influences course choice, RQ2 on the relationships among different types of points and their correlations with final grades, and RQ3 on how student performance varies across years, courses, genders, weeks, and themes.

Visualization techniques were used to enhance data understanding. Histograms and bar charts visualized numeric and categorical data distributions, while heatmaps displayed correlation matrices among different types of points (AP, PP, SPA). Pairplots facilitated multivariate analysis, and pie charts showed course distributions by gender. Boxplots compared grade distributions by gender across courses, and line and bar plots tracked the weekly evolution of points. Plots aided in the analyze of performance by thematic areas for each sex and course. All visualizations and results were saved into Word documents and a zip file for documentation and sharing. This notebook is not intended for training and evaluating machine learning algorithms but for understanding the data and answering the first three research questions.

3.5 Modeling, Evaluation and Deployment

For clustering analysis aimed at answering RQ4, algorithms such as K-Means Clustering (KMC) and hierarchical clustering were applied to segment the data and discover patterns.

These techniques helped in identifying groups of students with similar characteristics, which could then be analyzed further to have more knowledge (to be able to improve educational strategies). Before applying these algorithms, categorical variables were encoded using one-hot encoding, and continuous variables were scaled using standard scaling techniques. The optimal number of clusters was determined using the Elbow Method and Dendrogram. The clustering results were evaluated using Silhouette Scores, the Davies-Bouldin index (DBI), Calinski-Harabasz (CH) index, and visual inspection of clusters, with patterns within clusters analyzed to understand common characteristics and trends.

The predictive modeling phase of this research focused on five target variables: `Total_AP` (total of activity points), `Total_PP` (total of participation points), `Total_SPA` (total of self and peer assessment points), `Passed`, and `FinalGrade`. The first three variables were analyzed both as continuous (regression) and discrete (classification) targets, while `Passed` was treated as a binary classification problem. `FinalGrade`, although continuous, presented missing values for students who failed (`FinalGradeInteger` < 10), necessitating special treatment.

A consistent methodology was applied across all target variables using a custom function to train and evaluate various machine learning models. This function allowed for both classification and regression tasks, selecting the appropriate models and hyperparameters depending on the task at hand. For regression tasks, models such as Linear Regression, Decision Trees (DTs), Random Forests (RFs), Support Vector Machines (SVMs), kNN (k-Nearest Neighbors), and Artificial Neural Networks (ANNs) were employed. For classification tasks, models like Logistic Regression (LR), DT, RF, SVM, kNN, and ANNs were used. The appropriate variant of each model (regressor or classifier) was selected based on whether the target variable was continuous or categorical.

Hyperparameter optimization was conducted using GridSearchCV, allowing for fine-tuning of model parameters based on cross-validation performance. The key metrics for model evaluation included MSE (Mean Squared Error), Coefficient of Determination (R^2), and Mean Absolute Error (MAE) for regression tasks, and accuracy and F1-score for classification tasks. The best models for each target variable were selected based on their cross-validation scores and test set performance.

For the `FinalGrade` target, which presented unique challenges due to its mixed nature (missing values for failures), more advanced ensemble techniques were explored. These included Bagging with Random Forest to reduce variance, Boosting via Gradient Boosting Regressor to iteratively correct errors, and Stacking with a combination of base models (Random Forest, Gradient Boosting, and Support Vector Regressor (SVR)) and a linear regression meta-learner. Additionally, more sophisticated algorithms such as XGBoost, LightGBM, and CatBoost were employed to further enhance the predictive accuracy for `FinalGrade`, addressing complex non-linear relationships in the data. It was also tried several imputation procedures in conjuctions with this ensemble techniques, including mean, median, most frequent, zero, and K-Nearest Neighbors (kNN).

The overall modeling approach ensured that each target variable was analyzed using the most appropriate techniques, with ensemble methods and hyperparameter tuning applied where necessary to optimize performance. The results were validated through cross-validation and rigorous testing, ensuring the models' robustness and generalizability.

The final phase of the study focused on preparing the data and models for sharing and reproducibility. Instead of deploying the models in a practical educational setting, the emphasis was placed on creating a comprehensive Data Management Plan (DMP) to facilitate data sharing and validation among researchers. All datasets were anonymized to ensure student privacy, and the research outputs were uploaded to Zenodo, a research data management platform, enabling organized and accessible data storage. The DMP outlined procedures for data collection, processing, storage, and sharing, ensuring that the study could be replicated and its findings validated by other researchers, as shown in Annex **??**. This meticulous approach to data management underscores the study's commitment to transparency and reproducibility.

4

Results

This chapter presents the findings from the data analysis, focusing on the insights drawn from the statistical and machine learning analyses conducted on the available educational data with evaluation results from several curricular years. The results elucidate the dynamics of student performance within the CTCT course, give a different perspective on the evolution of this curricular unit, and bring knowledge about the population of science and engineering students in the first year of their degree in FCT (NOVA School of Science and Technology).

4.1 Data Description and Initial Processing

Data from several educational datasets was collected and preprocessed as specified in Chapter 3 and comprised categorical variable uniformization, missing value handling, and numerical data transformation. Refer to Section A.1 of Appendix A for a more detailed explanation of the data characteristics.

The heatmap in Figure 4.1 depicts the generated pivot_table from the process_activities method on the data_preparation Google colaboratory notebook mentioned in Section A.2. It presents the absolute frequency and types of activities (e.g., written or oral assessments, group or individual tasks, homework or in-class activities, activities designed for all or selected students) across each edition of the discipline studied, and it is a necessary instrument to assess how the studied discipline's activities evolved.

The heatmap illustrates the distribution and frequency of different activity types over the years. As shown in Figure 4.1, activities categorized as type 'C' appear infrequently, often with fewer than five occurrences per category each year. This low frequency complicates the analysis but highlights the uniqueness of these activities in influencing student outcomes. The most consistently frequent activities are written group tasks completed in class by all students and oral individual presentations done by some students. The latter has seen a decline in frequency since 2022, following the introduction of a new evaluation method that credits participation differently, affecting how activities are categorized from this year onwards. Activities marked as "not evaluated" appear in

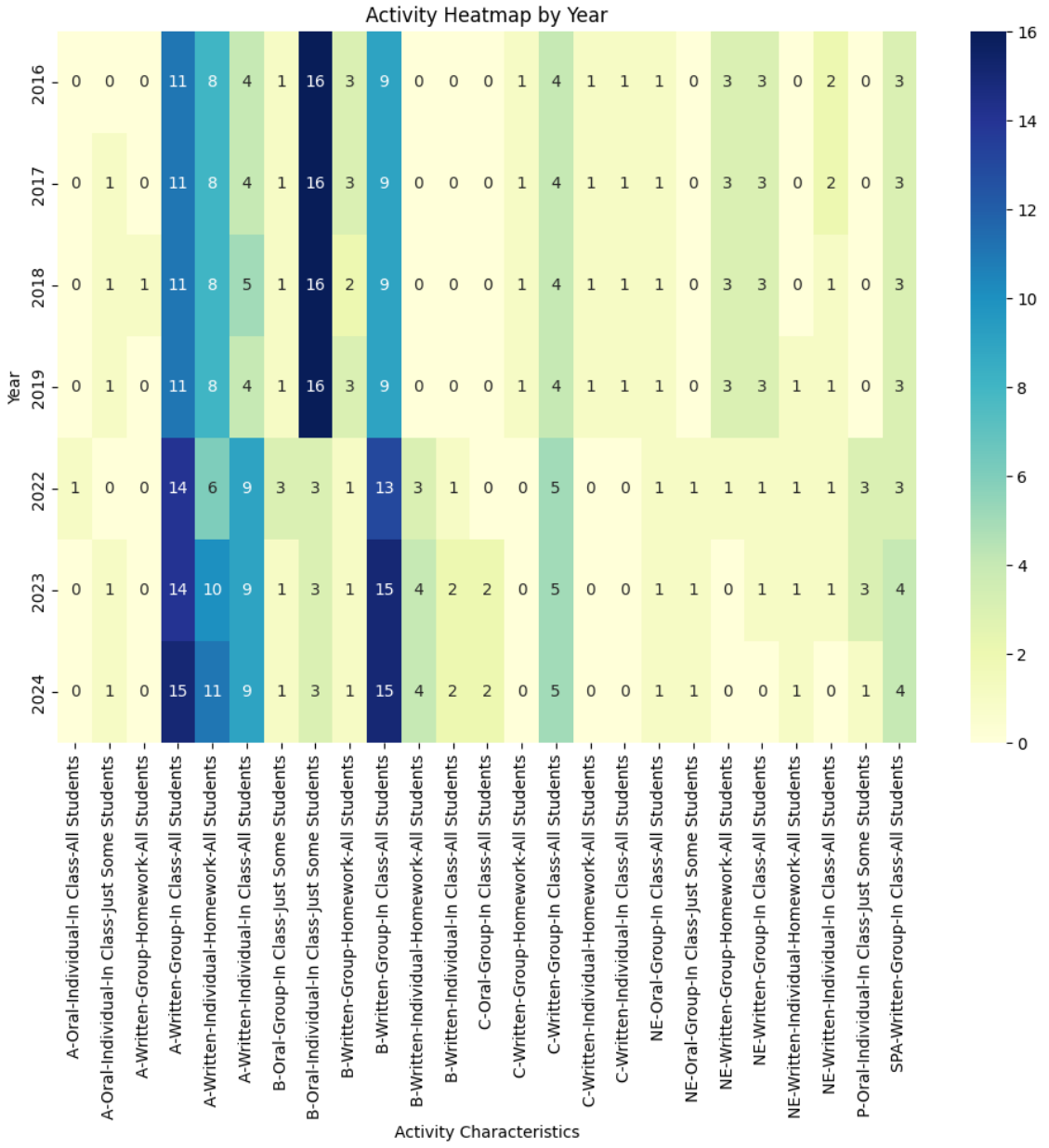


Figure 4.1: Activity type heatmap.

the dataset but do not impact student evaluations, providing limited insights into their educational effectiveness.

Also, it was chosen to perform various pre-processing processes in the same notebook as the modeling and assessment, because some parts are dependent on the specific analysis and algorithm used. So scaling and feature selection were not realized during the initial processing.

4.1.1 Descriptive Statistics

Descriptive statistics are fundamental to understanding data properties, including the study on central tendencies, dispersion, and distribution shapes. This subsection explores different types of assessment points. Participation Points (PP) assess students' involvement and contributions in class, including good (adding points) and disruptive (subtracting points). Activity Points (AP) represent the points earned by students participating in evaluation types A, B, and C (with maximum values of 1, 3, and 5 points, respectively). The Self and Peer Assessment (SPA) points are produced from self-evaluations and peer reviews and enable students to critically reflect on their own and their peers' contributions to group work. Students must distribute a set of points among the group but the overall number of points assigned to each group to deliver among their elements was not always consistent, and the number of SPA moments also varied (from three in 2022 and before, to four times of this type of assessment, as shown in Figure 4.1). Bonification Points (Bon) are awarded at the discretion of the practical class teachers and CTCT coordination at the end of each course edition; these points serve as a bonus for significant student contributions and are not included in the usual evaluation.

Table 4.1 provides thorough descriptive statistics for the CTCT discipline throughout many academic years. It describes how final grades are distributed as well as the many categories of academic points. Each category is further broken down by statistical measurements like mean, standard deviation, skewness, kurtosis, and range, which provide information about their distribution characteristics. The table doesn't show the minimal values for AP, PP, and SPA because these values are always zero (to simplify the table was omitted). In the case of bonification points, the table clearly shows the number of students who received zero and those who received one bonification point, providing a better picture of how rarely these points are given. The inclusion of total student counts per academic year expands the data's context, allowing for a more in-depth analysis of demographic changes.

Table 4.1: Descriptive statistics of columns `FinalGrade`, `Total_AP`, `Total_PP`, `Total_SPA` and `Total_Bon`.

Type of Points	Measure	2016	2017	2018	2019	2022	2023	2024
Student Count	**Total**	1169	1152	1105	1055	1179	1109	1121
	Passed	1106	1070	1076	1013	1111	1060	1062
	Failed	63	82	29	42	68	49	59
`FinalGrade`	**mean**	15.98	15.75	17.01	16.62	15.59	15.94	15.21
	std	2.13	4.49	1.50	1.47	3.84	1.53	1.61
	min	6.6	0.0	8.8	10.2	0.0	7.5	7.9
	25%	14.9	15.71	16.3	15.8	15.54	15.2	14.2
	50%	16.5	17.08	17.4	16.9	16.75	16.2	15.5
	75%	17.5	18.08	18.0	17.7	17.40	17.0	16.5
	max	19.9	20.0	19.9	19.8	19.44	19.5	18.6
	skew	-0.94	-2.80	-1.35	-0.90	-3.11	-1.11	-0.85
	kurt	0.73	7.05	2.77	1.01	9.46	2.17	0.72
`Total_AP`	**mean**	80.95	84.11	86.64	85.26	90.82	107.01	104.33
	std	21.36	26.32	20.84	22.09	29.64	28.80	28.91
	25%	78.00	84.00	85.00	84.00	94.00	105.00	102.00
	50%	86.00	92.00	92.00	91.00	100.00	115.00	113.00
	75%	92.00	97.00	96.00	96.00	105.00	120.00	119.00
	max	107.00	111.00	109.00	110.00	114.00	136.00	131.00
	skew	-2.91	-2.59	-3.40	-3.10	-2.57	-3.00	-2.87
	kurt	8.46	5.61	11.49	9.34	5.15	8.39	7.71
`Total_PP`	**mean**	18.27	18.89	18.47	19.81	14.30	19.04	21.40
	std	4.90	5.82	4.53	5.24	4.25	5.24	5.83
	25%	17.00	18.00	18.00	19.00	13.00	18.00	20.00
	50%	19.00	20.00	19.00	21.00	15.00	20.00	22.00

Continued on next page

Table 4.1: (continued)

Type of Points	Measure	2016	2017	2018	2019	2022	2023	2024
	75%	21.00	22.00	21.00	22.00	16.00	22.00	24.00
	max	27.00	28.00	26.00	29.00	22.00	28.00	31.00
	skew	-2.45	-2.31	-2.70	-2.40	-1.97	-2.30	-2.39
	kurt	7.13	5.15	9.15	7.38	5.07	6.39	6.74
`Total_SPA`	**mean**	28.02	26.70	27.84	27.63	6.12	8.40	8.37
	std	7.57	9.81	8.53	8.69	2.14	2.44	2.51
	25%	28.00	28.00	29.00	29.00	6.00	8.00	8.00
	50%	30.00	30.00	30.00	30.00	7.00	9.00	9.00
	75%	31.00	31.00	31.00	31.00	7.00	9.00	9.00
	max	35.00	36.49	36.00	35.46	9.00	12.00	12.00
	skew	-3.14	-2.21	-2.76	-2.72	-2.04	-2.33	-2.34
	kurt	8.97	3.34	6.32	5.91	3.56	6.22	5.87
Number of Students with `Total_Bon`	**=0**	1133	1122	1086	1028	1179	1093	1046
	=1	36	30	19	27	0	16	75

The success rates of students over the years 2016 to 2024 have been calculated and analyzed to assess the effectiveness of educational strategies. The highest success rate was observed in 2018 with 97.38% of students passing CTCT course. Conversely, the lowest success rate was recorded in 2017, where only 92.88% of students succeeded. This variation in success rates could suggest changes in course difficulty, assessment standards, or student characteristics.

The analysis of descriptive statistics from Table 4.1 reveals significant patterns and features for points and `FinalGrade` from 2016 to 2024, with varying degrees of positive kurtosis and negative skewness. Final grades, which are often stable with very minor fluctuations, demonstrate consistent grading standards. The lower skewness in subsequent years suggests a fair and equitable assessment approach. It's important to remember that the final grade data only includes students who finished CTCT. The final grade attribute for failed students was set to 0, leaving these students with missing values. Figures 4.2 and A.5 depicts the distribution of final grades for students who passed, before rounding

to the closest integer. As seen, in 2016, 2022, and 2024, students passed with grades below 9.5.

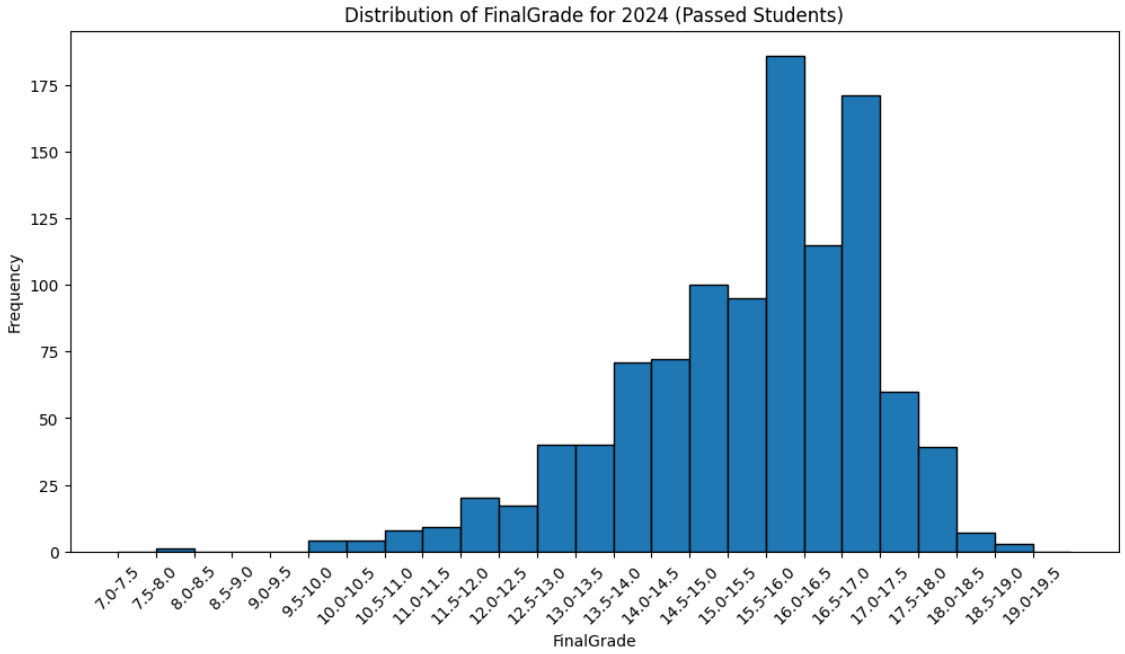


Figure 4.2: Histograms representing the distribution of students final grades for the students that didn't reprove (`FinalGrade`) for 2023-24 school year.

With the highest value rising from 114 in 2022 (the COVID year) to 136 in 2023, `Total_AP` (total of activity points) indicates changes to the scoring system. The overall trend is upward, especially from 2022 to 2023. On the other hand, `Total_PP` (total of participation points per student) exhibits minor fluctuations annually with no apparent upward or downward trend, indicating that the scoring standards have not changed. The steady decline of the initial high values of skewness and kurtosis points to a tendency towards a more uniform distribution of participation ratings. For further details on the distribution of these points, see Figures A.6, A.7, A.8, and A.9 in Section A.3. The histograms for `Total_Bon` (total value of bonification points) were also generated by data_understanding notebook but the results are not shown because have less information than what was presented in Table 4.1.

The noticeable drop in average `Total_SPA` (total of self and peer assessment per student scores) from 2019 onwards, that is also confirmed by the histograms presented in Figures 4.3 and A.10, suggests potential changes in the assessment criteria or a shift in the emphasis placed on these points in overall evaluations. However, contrary to the initial observation, the skewness and kurtosis values do not consistently indicate a significant increase in the spread of scores. For example, while the average `Total_SPA` dropped sharply, indicating lower scores overall, the skewness and kurtosis remain relatively stable, with only slight variations. This suggests that while the central tendency of the `Total_SPA` scores decreased, the distribution of scores did not become significantly more spread out or less peaked in a uniform way across all years.

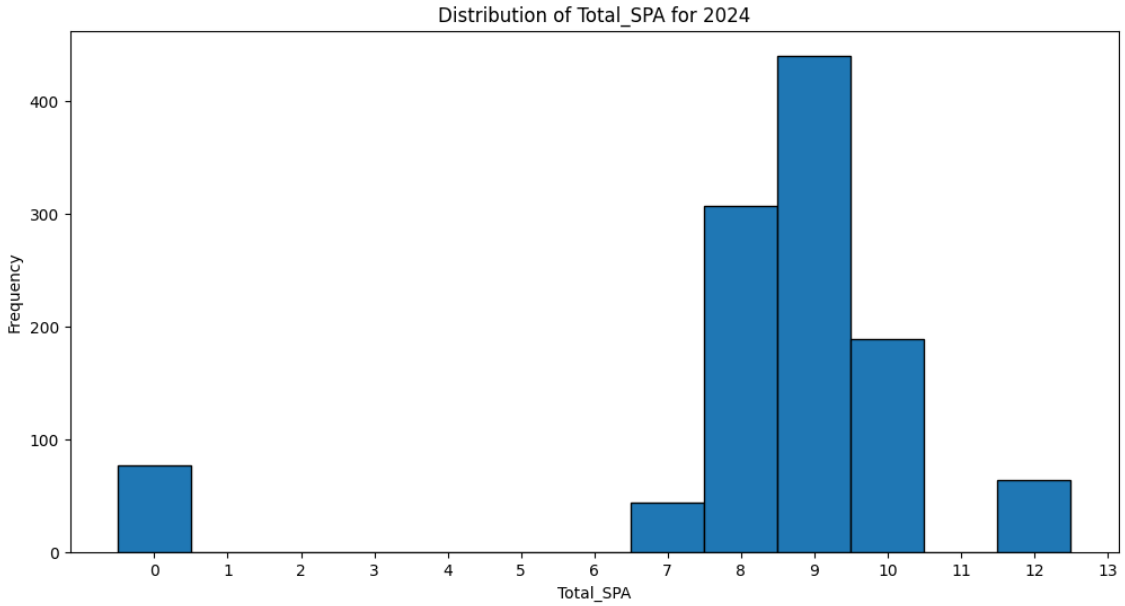


Figure 4.3: Histograms representing the distribution of the total value of self and peer assessment points (`Total_SPA`) for 2023-24 school year.

4.1.2 Statistical Testing

4.1.2.1 Shapiro-Wilk Test Analysis

The Shapiro-Wilk test, a method for assessing normality, was applied to evaluate the distribution characteristics of various attributes across multiple academic years. This test determines whether a sample likely originates from a normally distributed population, with a p-value above 0.05 indicating typical distribution traits. However, our analysis revealed that all tested attributes deviated significantly from normality, as evidenced by consistently low p-values far beneath the 0.05 threshold.

Each attribute was tested unless it presented only a single unique value, which inherently precludes a normal distribution. The results showed no attributes adhering to a normal distribution across the years, with maximum W values indicating some attributes were close but still outside acceptable normality limits. Specifically, the W values ranged from as low as 0.010 in 2016 to a high of 0.955 in 2024, while p-values were consistently less than 2.03×10^{-20}, far from the threshold needed to deem the distributions normal.

From 2016 to 2024, the pattern of non-normality persisted, with the 'Normal' generated column affirming the absence of normal distribution for all tested attributes. This consistent deviation suggests that parametric methods, which assume normality, may not be suitable for this dataset. Instead, non-parametric approaches could be more appropriate for analyzing these data, providing a basis for robust, alternative analytical strategies that accommodate the actual distribution characteristics observed in the dataset. This comprehensive analysis confirms the need for adaptable, non-parametric techniques in handling and interpreting the complex data effectively.

4.2 Analysis of Research Questions

Statistical analyses were conducted to explore the research questions summarized in Table 1.1, focusing on gender influence in course selection, relationships among assessment types, variability in student performance, identification of performance patterns, and the predictive value of early performance on final outcomes. The results regarding this analysis are presented in this section.

4.2.1 Gender Influence on Course Selection

This analysis examines the impact of gender on students' course selection at NOVA School of Science and Technology from 2016 to 2024 using data from multiple curricular years in the CTCT discipline. As can be seen in Table 4.2, annual chi-squared tests revealed a strong and consistent correlation between gender and course enrollment (attributes Sex and Course, respectively); p-values were almost zero, indicating highly significant results. This persistent pattern reveals notable disparities in the courses chosen by males and females, suggesting deeply rooted prejudices or preferences that influence academic careers.

Table 4.2: Chi-Squared test results.

Year	Chi-Squared Statistic	P-value
2016	285.298	6.30×10^{-52}
2017	285.695	5.21×10^{-52}
2018	231.971	6.30×10^{-41}
2019	258.037	1.16×10^{-45}
2022	322.704	4.98×10^{-59}
2023	274.995	3.75×10^{-49}
2024	262.618	5.48×10^{-46}

Chi-squared statistics revealed pronounced values, ranging from 231.971 in 2018 and a peak of 322.704 in 2022. Such values highlight a strong bias in course enrollments along gender lines. The chi-squared values for 2016, 2017 and 2022 were particularly telling, both exceeding 285, with corresponding p-values below 1×10^{-50}, further emphasizing deep-rooted gender disparities in educational choices at the institution.

4.2.2 Interrelations Among Assessment Points and Final Grade

This analysis studies the correlation between Activity Points (AP), Participation Points (PP), and Self and Peer Assessment Points (SPA) and the final grade of CTCT discipline

throughout many academic years. The study seeks to determine how participation in various academic activities relates with overall success among students.

Multivariate analysis, which includes correlation coefficients and regression models, provides a thorough understanding of these interdependencies. Pairplots, like the one in Figure 4.4, visualize the distribution and pairwise relationships among assessment points for the 2024 curricular year. Figures A.2, A.3, and A.4 in Appendix A show the remaining pair plots. .

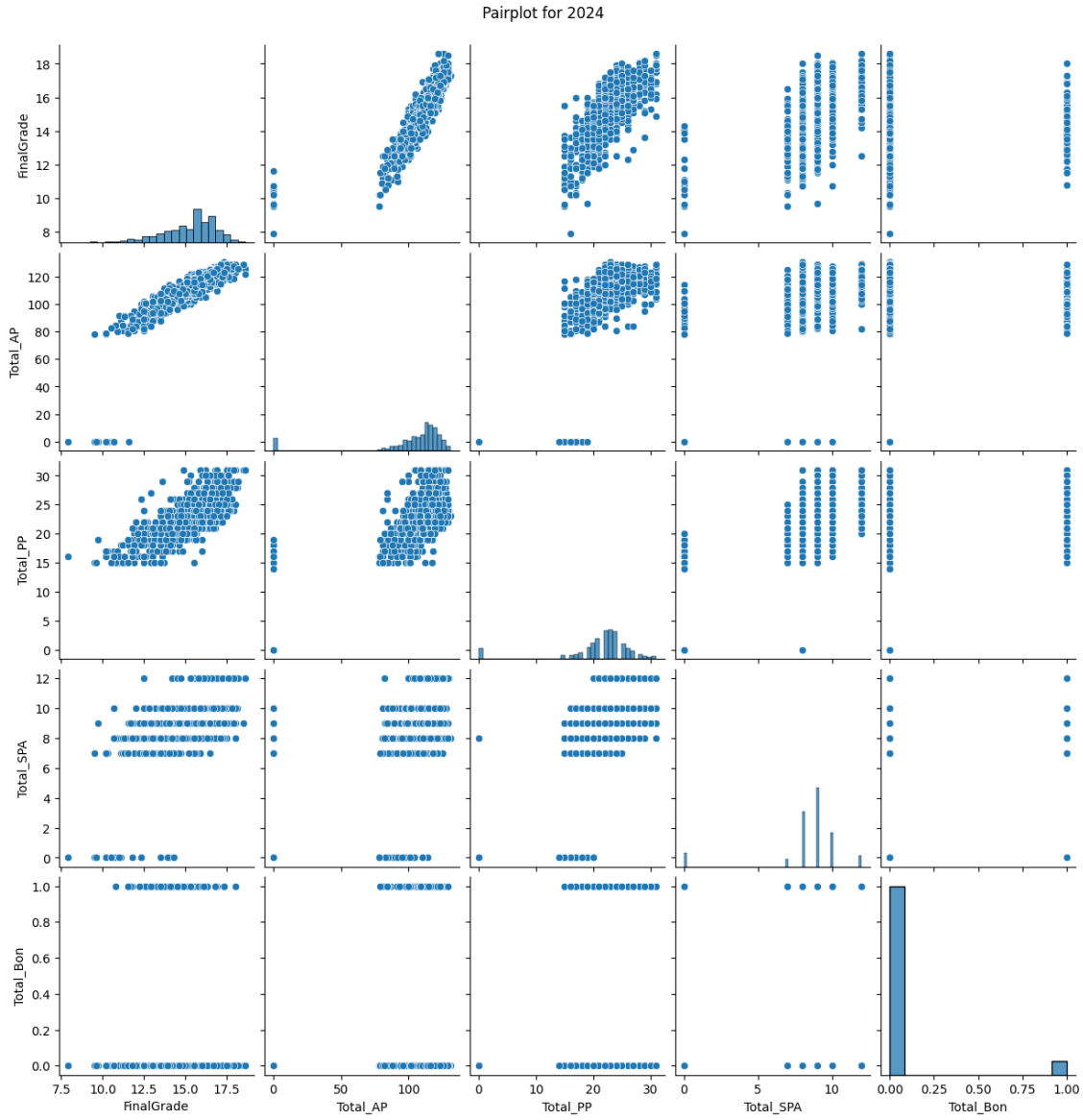


Figure 4.4: Pairplot depicting relationships between `Total_AP`, `Total_PP`, `Total_SPA`, `Total_Bon`, and `FinalGrade` for the 2024 edition of the CTCT course.

The pair plots show a consistently favorable association across numerous metrics. The scatter plots for `Total_SPA` in 2022, 2023, and 2024 differ significantly from the previous ones due to evaluation revisions. A substantial association between `Total_AP` and `Total_PP` suggests that students who actively participate in coursework are also

highly engaged in class discussions and activities. This association is supported by dense clustering in their scatter plots. The positive relationship between `Total_AP` and `Total_SPA` indicates that students who do well in course activities tend to score higher in self and peer assessments. A positive connection between `Total_PP` and `Total_SPA`, however slightly weaker than others, indicates that class participation has a beneficial impact on peer evaluation ratings. The distribution of Bonification Points (`Total_Bon`) is different from other metrics, predominantly skewed towards zero, indicating uncommon distribution and not much relationship with other assessment types.

Further analysis through correlation matrices from 2016 to 2024 quantifies the strengths of these relationships and the results are presented in Table 4.3.

Table 4.3: Yearly correlation matrices for total of sum of assessment points.

Attribute	Total_AP	Total_PP	Total_SPA
2016			
Total_AP	1.000	0.896	0.842
Total_PP	0.896	1.000000	0.832
Total_SPA	0.842	0.832	1.000
2017			
Total_AP	1.000	0.909	0.731
Total_PP	0.909	1.000	0.736
Total_SPA	0.731	0.736	1.000
2018			
Total_AP	1.000	0.777	0.595
Total_PP	0.777	1.000	0.593
Total_SPA	0.595	0.593	1.000
2019			
Total_AP	1.000	0.823	0.695
Total_PP	0.823	1.000	0.665
Total_SPA	0.695	0.665	1.000
2022			

Continued on next page

Table 4.3: (continued)

Attribute	Total_AP	Total_PP	Total_SPA
Total_AP	1.000	0.766	0.739
Total_PP	0.766	1.000	0.731
Total_SPA	0.739	0.731	1.000
2023			
Total_AP	1.000	0.752	0.777
Total_PP	0.752	1.000	0.737
Total_SPA	0.777	0.737	1.000
2024			
Total_AP	1.000	0.870	0.775
Total_PP	0.870	1.000	0.817
Total_SPA	0.775	0.817	1.000

As shown in Table 4.3, the correlation coefficients in 2016 were particularly strong, with `Total_AP` and `Total_PP` at 0.896 and `Total_AP` and `Total_SPA` at 0.842. While the relationships remained generally strong in subsequent years, variability is observed, such as a drop in the `Total_AP` and `Total_SPA` correlation to 0.595 in 2018. This variability underscores the potential influence of year-specific educational strategies or external factors on assessment outcomes. For example, it's important to remember that 2022 was different from the other years because this discipline that it's mostly presencial was given in remote mode due to Covid-19 and was also only three weeks.

Table 4.4 provides insights into the correlation betsween assessment metrics and final grades. Firstly, there is a consistently strong positive correlation between final grades and both the sum of activity points per student (`Total_AP`) and the sum of participation points (`Total_PP`) across all years. Correlations range from 0.75 to 0.84 for `Total_AP` and 0.57 to 0.81 for `Total_PP`.

Table 4.4: Correlation between final grade and total points for each assessment typology.

Year	Total_AP	Total_PP	Total_SPA	Total_Bon
2016	0.82	0.81	0.52	-0.19

Continued on next page

Table 4.4: (continued)

Year	Total_AP	Total_PP	Total_SPA	Total_Bon
2017	0.78	0.73	0.33	-0.17
2018	0.75	0.71	0.46	-0.1
2019	0.78	0.67	0.45	-0.11
2022	0.83	0.61	0.59	-
2023	0.83	0.57	0.56	-0.094
2024	0.84	0.73	0.5	-0.055

In addition, the correlation between final grades and the sum of all self and peer assessment points (`Total_SPA`) varies significantly, with lower correlations in earlier years (0.33 in 2017) and stronger correlations in later years (0.59 in 2022). This means that the relevance of self and peer appraisals may have grown over time, which contradicts the impression given by Figure 4.3 that the SPA is less since 2022 compared with previous editions of CTCT. The sum of bonification points given to each student (`Total_Bon`) has a poor and often negative association with final grades, indicating that they may benefit some students but are not a major predictor of overall academic achievement.

4.2.3 CTCT Performance Variability

This analysis explores the variability in student performance across different editions of the CTCT discipline at FCT measured by average points. It aims to identify patterns and trends in performance influenced by factors such as course, gender, week, and subject.

4.2.3.1 Variations in Performance by Course and Gender

Bar charts that offer a comparative study of the total enrollment (represented by blue bars) and failure percentages (represented by orange bars) across various sex categories in the CTCT discipline for each year are shown in Figure 4.5, where the x-axis denotes the gender of the students and y-axis represents the percentages.

Despite a greater enrollment rate for male students, the study of gender-related data indicates that male students had higher failure rates in the CTCT discipline than female students, as shown in Figure 4.5. This tendency gains more validity from the data in Table A.7, which provides the average points for each assessment for each gender throughout the course of the seven academic years under consideration.

The number of students who fail the discipline annually, the overall failing percentage, and the university degree courses with the highest and lowest failure rates in relation to their total enrollment in the CTCT discipline are all displayed in Table 4.5. It illustrates that

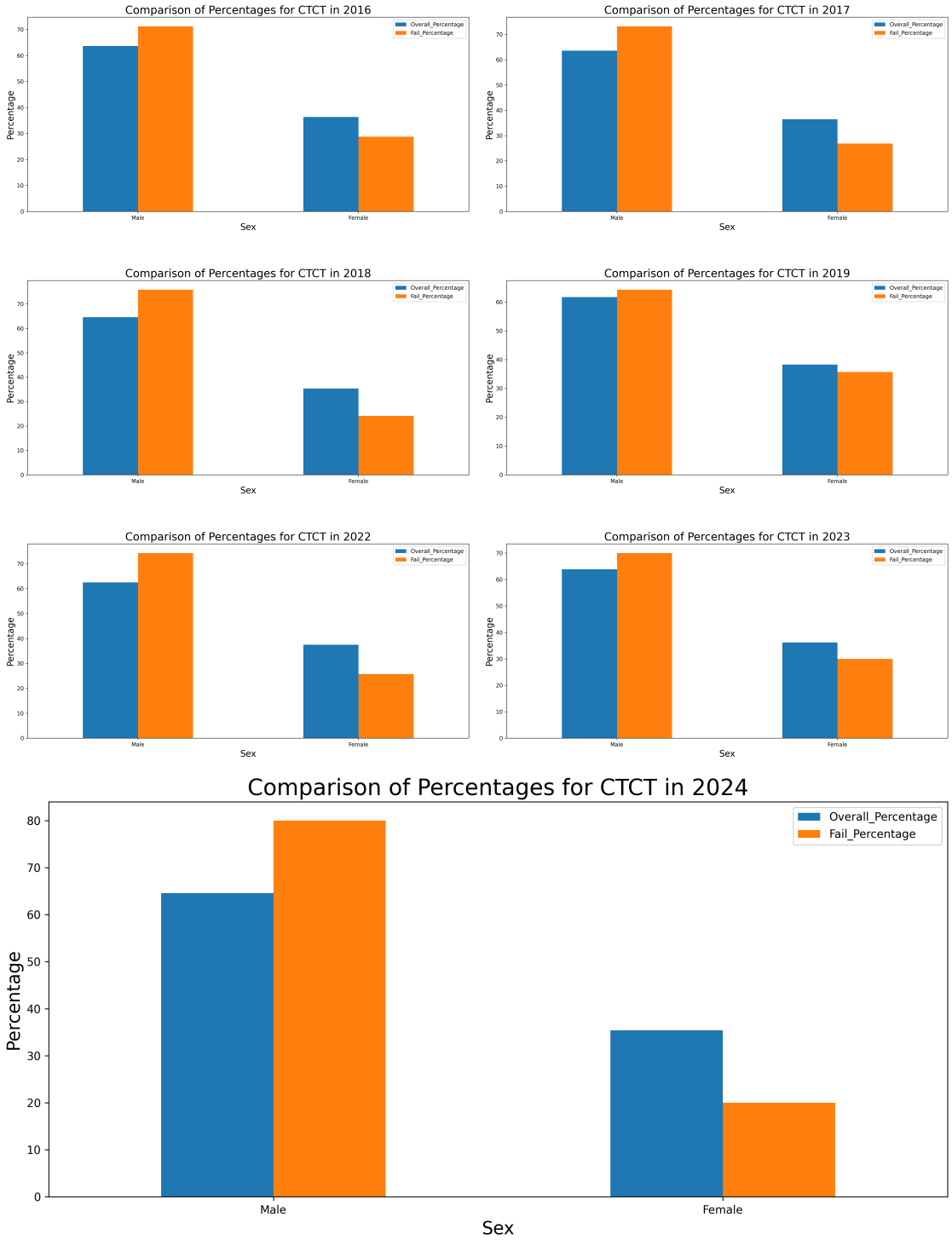


Figure 4.5: Comparison of the enrollment percentage of students by sex and the proportion of students who fail the course.

informatics, civil, electrical, and biomedical engineering courses had the highest failure rates (depending on the year), while applied chemistry and physical engineering courses had the lowest rates.

Table 4.5: Comparison of failure rates by course.

Year	Courses with Fail Percentage Higher Than Overall Percentage	Courses with Lower Fail Percentage	Fail Percentage (%)
2016	LBq, LM, LEC, LEI (highest difference)	LQA, LEGI, LEMt	5.39% (63 students)
2017	LEG, LM, LEA, LEC, LEF, LEGI, LEI (highest difference), LEMt	LBCM	7.12% (82 students)
2018	LEG, LM, LEA, LEB (highest difference), LEEC, LEF, LEI	LBCM, LQA, LEMN, LEMt, LEQB	2.62% (29 students)
2019	LEG, LMAGR, LEC, LEEC (highest difference), LEMt, LEMc, LEQB	LEF	3.98% (42 students)
2022	LBq, LEC (highest difference), LEEC, LEGI, LEI, LEQB, LM	LEMt, LMAGR, LQA	5.77% (68 students)
2023	LBq, LEA, LEB, LEC (highest difference), LEG, LEMN, LEQB, LMAGR	LEF, LEMt	4.42% (49 students)
2024	LEC, LEEC, LEI (highest difference), LTAI	LEF	5.26% (59 students)

The average per course of AP, PP, and SPA are shown as bar charts in Figures A.11, A.12, and A.13, respectively. These charts allow a more abrangent visualization of the performance per course. The significance of each course initials can be deduced from Table A.2.

The results of the Mann-Whitney U tests, which were used to look into the disparities in grade distributions between male and female students in a variety of courses, are shown in Table 4.6. Since the grades do not follow a normal distribution, the non-parametric test was chosen since it is appropriate for comparing two independent samples when the t-test's assumptions are not satisfied.

Table 4.6: Courses with statistically significant differences in grades between genders.

Year	Course	Mann-Whitney U Statistic	p-value
2016	**LEMc**	271.5	0.019
	LEEC	793.5	0.009
	LEMt	29.5	0.017
2017	**LEI**	1017.5	0.000
	LEF	28.0	0.008
2018	**LEGI**	261.5	0.007
	LEMN	499.0	0.040
	LEC	160.0	0.032
2022	**LEQB**	454.0	0.025
	LMAGR	27.0	0.006
2023	**LBq**	364.0	0.011
2024	**LEGI**	286.0	0.018

The findings point to a number of courses in which gender differences are statistically significant. An especially low p-value ($p < 0.001$) was seen in the informatic engineering (LEI) in 2017, for example, indicating a significant difference in the grade distributions between male and female students. Similar differences were seen in courses like engineering and industrial management in 2024 and electrical and computer engineering in 2016 ($p = 0.018$ and $p = 0.009$, respectively).

A detailed boxplot showing the distribution of final grades by sex for different courses in 2024 is shown in Figure 4.6. The interquartile range (IQR), which includes the middle 50% of the data, is represented by the box. The median grade is shown by the line inside the box. Two boxplots are used for each subject; the blue box represents the male students and the red box the female students.

In comparison to their male counterparts, female students in this course edition received higher median grades in CTCT that are enrolled in courses including biology-related courses (LBCM, LBq) and a variety of engineering courses (LEG, LEEC, LEGI, LEMN, LEMc, and LEMt). In addition, there is a significant amount of gender overlap in the grade variability across the majority of courses, as evidenced by the length of the boxes and the whiskers. Males that are enrolled in mathematics-related courses LM and

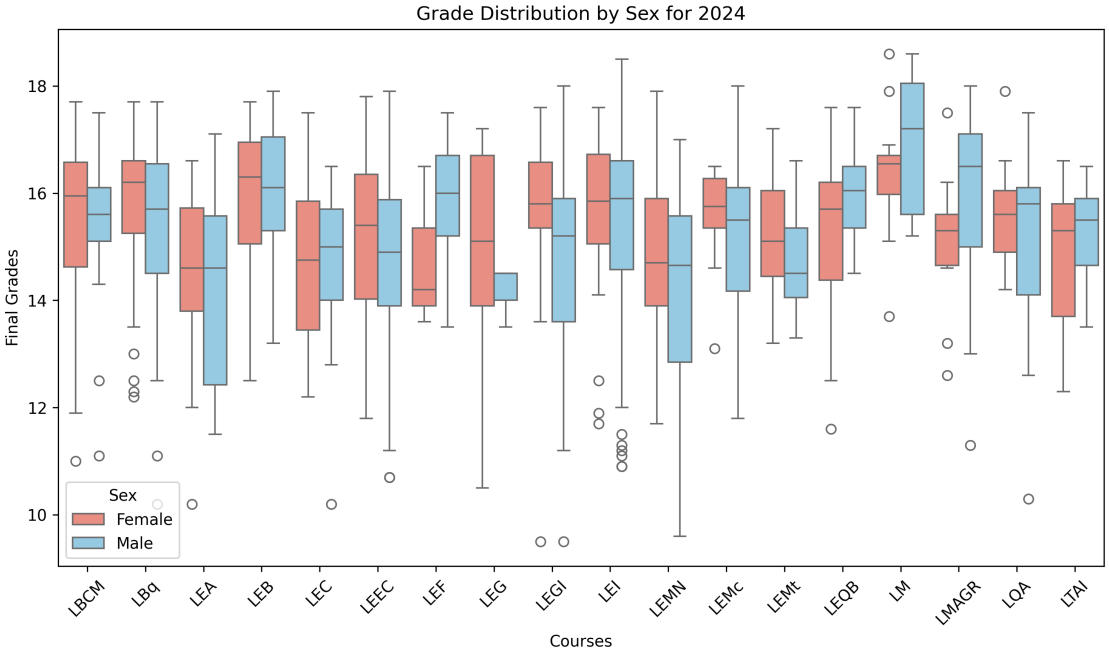


Figure 4.6: Boxplot with FinalGrade per Course and Sex for 2024.

LMAGR, as well as in Applied Chemistry (LQA), Physics Engineering (LEF), and the recently established Agro-Industrial Technology (LTAI) score better. The existence of both extraordinarily high and low grades outside of the average range is indicated by outliers, which are individual points outside of the boxplots. This highlights the diversity of academic outcomes within each gender group. Because the distribution of final grades is not symmetric, it makes sense that the majority of outliers are lower.

ANOVA was used in a more thorough investigation to determine the impact of gender and course choice on final grades, using sex and course as categorical factors. The effects on student outcomes were assessed using the interaction model FinalGrade $\sim C(\text{Course}) + C(\text{Sex}) + C(\text{Course}) : C(\text{Sex})$ and the results are presented in Table A.9. In 2016 and 2017, course choice had a greater influence on final grades than gender, with no discernible interaction, indicating that course choice had a constant effect on grades for all genders. From 2018 to 2024, differences were seen in the influence of course and gender on grades, with little evidence of interaction effects. Particularly, in 2018 gender effects were more prominent than in other years, especially in 2024, when course effects predominated. These results imply that although course and gender have a substantial impact on academic achievements, their interactions are often small, suggesting that the influence of course selection is constant for both genders.

4.2.3.2 Variations in Performance by Week and Thematic

In addition to examining the correlation between gender and course with CTCT performance, research was conducted on the association between course week and discipline subject.

The themes are divided primarily by week during the three to five weeks of each edition of CTCT, though the sequence of the nine selected subjects hasn't always been consistent. The first two weeks have largely consistent themes; the first is typically associated with curriculum preparation that emphasizes employability, while the second is associated with time management, teamwork, and leadership. The course's single technical subject matter, which was a third-week focusing on advanced Excel spreadsheet abilities from 2016 to 2019 and again in 2022, was moved to the fourth week in 2023 and 2024. Before 2020, the fourth week of the course covered bibliographic research, information analysis, ethics, and deontology; the fifth week covered communication in science and technology; these topics were combined to become the third week of 2023 and 2024.

Table 4.7 displays the average activity points per week for the best- and poorest-performing university degrees in the CTCT discipline. During the first week, the values ranged from 7.12 AP for the weakest course in 2017 (Geological Engineering) to 15.13 AP for the strongest course in 2024 (Physical Engineering). Regarding the second week, the values ranged from 19.32 AP in 2017 for Geological Engineering to 33.17 AP for Physical Engineering, which also achieved the best performance in 2023. In the third week, the average AP per course ranged from 16.10 (LEG course) in 2019 to 40.38 for Applied Chemistry. The fifth week, which is the only week available in the data from 2016 to 2019, is not included in this table.

Table 4.7: Best and worst university degree in CTCT in terms of average of activity points per week.

Year	Rank	Week 1	Week 2	Week 3	Week 4
2016	**First**	LQA (10.00)	LQA (27.24)	LEGI (21.20)	LQA (17.36)
	Last	LEI (8.47)	LEI (22.78)	LEC (18.60)	LEG (14.29)
2017	**First**	LBCM (10.81)	LEQB (26.92)	LBCM (22.78)	LBCM (22.55)
	Last	LEG (7.12)	LEG (19.32)	LEG (16.40)	LEG (16.48)
2018	**First**	LBCM (11.42)	LQA (27.48)	LBCM (21.96)	LBCM (22.03)
	Last	LEGI (10.00)	LEG (24.52)	LEG (18.67)	LEC (18.45)
2019	**First**	LM (11.20)	LEF (27.54)	LEF (21.29)	LEB (21.66)
	Last	LEG (8.90)	LEMt (23.27)	LEG (16.10)	LEC (18.06)
2022	**First**	LQA (22.89)	LMAGR (39.00)	LQA (40.38)	No data
	Last	LEC (18.39)	LEC (31.77)	LEC (31.51)	No data

Continued on next page

Table 4.7: (continued)

Year	Rank	Week 1	Week 2	Week 3	Week 4
2023	**First**	LEMt (13.13)	LEF (33.17)	LEMt (31.22)	LEF (38.38)
	Last	LMAGR (11.28)	LEC (28.89)	LEC (27.32)	LEC (31.96)
2024	**First**	LEF (15.13)	LEF (31.25)	LEB (30.65)	LM (37.18)
	Last	LEA (12.98)	LTAI (28.03)	LEG (26.78)	LEG (28.33)

The correlation matrices presented in Figures 4.7 and 4.8 offer insight into the relationships between different assessment points and how each one relates to the discipline week's theme topic. In contrast to the following weeks, the assessment points from the first week of the 2024 course exhibit less association with the total value of each sort of evaluation point. In line with the reasonable assumption that engagement and participation are related within the same timeframe, it is also noticed that the overall value of activity points for a certain week correlates more strongly with the participation points of that same week than with those of other weeks. This was also verified for the remaining years.

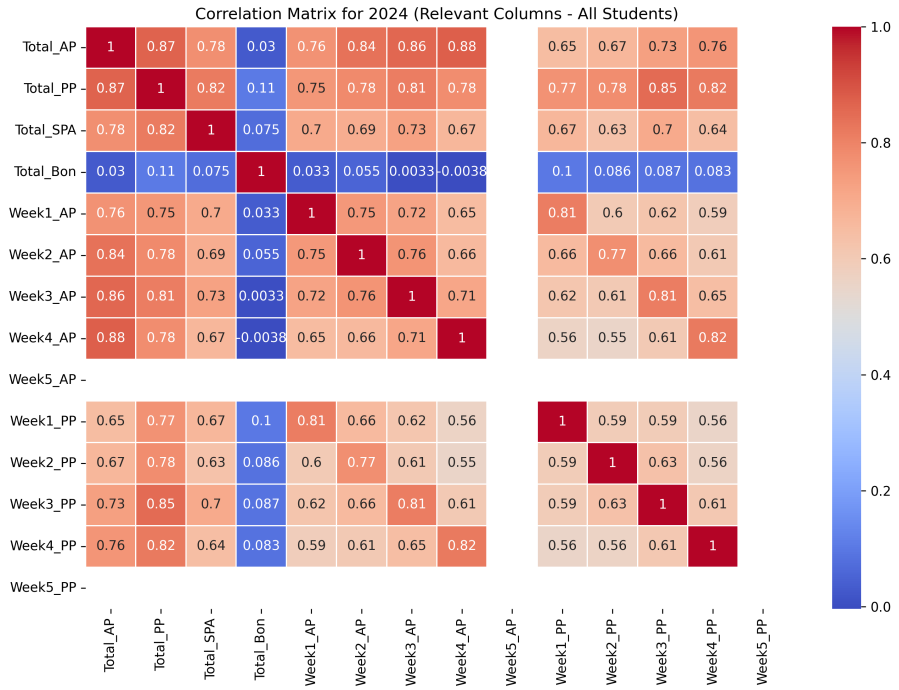


Figure 4.7: Correlation matrix for the 2024 edition of the CTCT course, illustrating relationships among different assessment metrics.

Additionally, it was noted that the week having the highest association with the final grade across all investigated years' correlation matrices corresponds to advanced

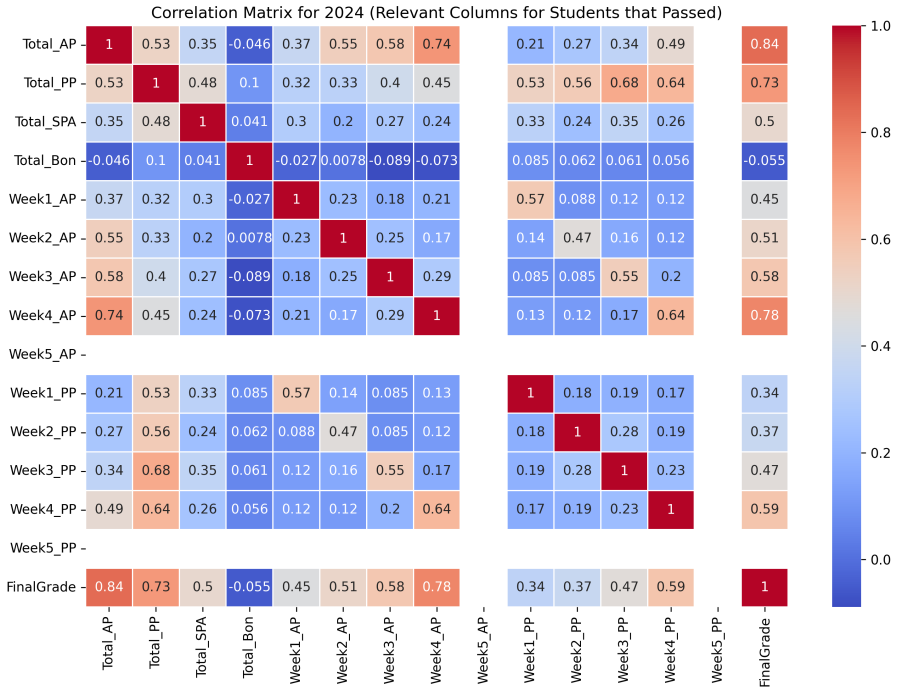


Figure 4.8: Correlation matrix for the 2024 edition of the CTCT course, including only students who passed, highlighting different interactions compared to the full student body.

spreadsheet use dedicated week. Additionally, bonus points have a generally low or negative association with other assessment measures, which suggests that students with lower performance in other areas may receive bonus points only as a corrective tool to help them fulfill passing requirements. Over all years, there was a lower association between `Total_SPA` and final grades than `Total_PP`. Additionally, there is less association between `Total_PP` and final grade than `Total_AP`.

Focusing solely on students who passed the course reveals that the correlation values among assessment metrics are noticeably lower, indicating more variation in how passing students acquire points. This variation suggests that different students may excel in various assignments or at different stages of the course. Notably, the `FinalGrade` exhibits strong correlations with `Total_SPA` and `Total_PP`, and an even stronger correlation with `Total_AP`, underscoring the predictive power of each type of assessment point.

Based on all weeks, the average total activity points for each sex are shown in Table 4.8. Specifically, compared to their male peers, female students typically have more activity points. It is also possible to assess the weekly weight allocation in accordance with the associated point value (of AP). For instance, there were more activity points in week two (with the themes of time management, teamwork, and leadership) than in week three (advanced spreadsheet use) or week four (bibliographic research, information analysis, ethics, and deontology) of the five-week CTCT classes prior to 2020. Less activity points were allocated to curriculum planning for employability and communication in science

and technology.

Table 4.8: Average of the total of activity points per week and gender.

Year	Sex	Week 1	Week 2	Week 3	Week 4	Week 5
2016	**Female**	9.42	25.50	20.33	16.15	12.15
	Male	8.99	24.34	19.97	15.22	11.47
2017	**Female**	9.75	25.45	21.32	20.78	11.51
	Male	9.11	23.86	20.23	19.12	10.48
2018	**Female**	10.85	26.47	20.27	21.03	11.47
	Male	10.54	25.77	19.92	19.85	11.01
2019	**Female**	10.72	25.72	19.32	20.43	11.36
	Male	10.53	25.36	19.46	19.29	10.94
2022	**Female**	21.45	36.62	38.09	-	-
	Male	20.75	35.07	36.33	-	-
2023	**Female**	12.32	31.67	30.21	36.08	-
	Male	12.12	31.03	29.22	35.41	-
2024	**Female**	14.39	30.26	29.82	34.52	-
	Male	13.69	28.95	28.39	32.78	-

Thematic analysis of average normalized points by gender across various thematic areas reveals consistent patterns in the CTCT discipline. Female students generally outperform their male counterparts in most thematic areas, such as ethics and deontology, advanced Excel use, and information searching. For instance, in 2024, female students had an average normalized point of 0.87 in advanced Excel use compared to 0.83 for male students. Similarly, in ethics and deontology, females scored 0.78 while males scored 0.74.

Heatmaps such as the one shown in Figure 4.9 demonstrate the course-wise performance variability, with some courses consistently achieving higher average scores across thematic regions. For instance, in 2024, strong performance was demonstrated by the persistent high scores in the majority of thematic areas for courses like LBCM, LBq, LEB, LEF, LM, and LQA. Conversely, lower average points were demonstrated in courses like LEG (Geological Engineering), especially in advanced spreadsheet use. In terms of leadership, LMAGR and LTAI stand out as the worst in 2024, whereas LEB and LBCM stand out for their accomplishments in deontology and ethics.

Figure 4.9: Normalized average by thematic for 2024 CTCT edition.

It is noteworthy to mention that the normalized average determined by thematic heatmaps varies annually, based on the classification type (A, B, or C) of the activities designated as falling under a particular theme. For instance, all courses received less than 0.6 in communication in science, technology, ethics, and deontology from 2017 to 2019, and less than 0.5 in leadership topic. This low ranking may result from the fact that the evaluation method based solely on participation was developed later and that voluntary activities are counted as part of a certain theme, which normalizes them.

For more details about the first three research question results, consult Section A.6.

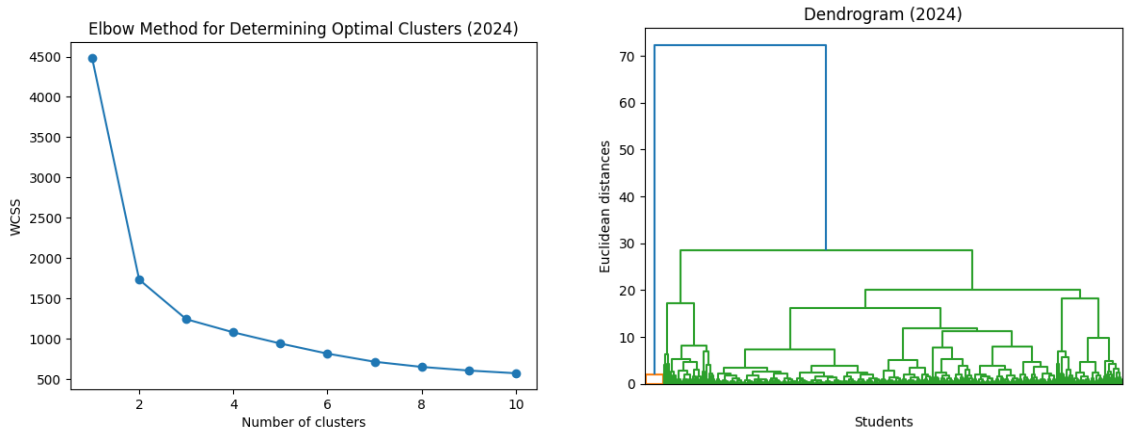
4.2.4 Clustering Student Performances

Without any prior knowledge of these group assignments, clustering is a potent data mining approach that can be used to find naturally occurring groupings or patterns within a dataset. However, the attributes used to aggregate students will generate different outcomes, and a trial and error procedure is required to try to interpret the knowledge inherent in the available data. As a result, the clustering analysis was performed with various sets of features: using only two of the totals of a type of evaluation point, using all of the sums of each assessment point, alternating between totals and totals per week, and using the activity results for all of the activities of the first two weeks.

The dendrograms and elbow approaches used in the `Data Analysis Report` notebook

to test the ideal number of clusters when utilizing only `Total_SPA` and `Total_PP`, as well as `Total_PP` and `Total_AP`, consistently yielded the same result: two clusters. Furthermore, the representation of the two clusters in each of the two 2D plots added nothing to the pair plots produced for the multivariate data analysis, indicating that there are two categories of students: those with near-zero `Total_AP` points and those without. Several combinations of the following features were tried in this notebook: `Total_AP`, `Total_SPA`, `Total_PP`, `Week1_AP` (sum of activity points for the first week), `Week2_AP`, `Week3_AP`, `Week4_AP`, `Week5_AP`, `Week1_PP`, `Week2_PP`, `Week3_PP`, `Week4_PP`, `Week5_PP` (sum of participation points for the fifth week). All of the examined feature combinations yielded the best results for two clusters, with silhouette scores of more than 0.82 and Davies-Bouldin Index values of around 0.2.

Figure 4.10 presents the techniques employed to determine the optimal number of clusters using the elbow method and dendrogram for the 2024 dataset, using the `Week1_AP` to `Week5_AP` columns as clustering features. The results indicate that selecting two clusters is recommended for this analysis. The clustering results of K-Means clustering (KMC) and Hierarchical clustering for this number of clusters are shown in Figure 4.11, with Principal Component Analysis (PCA) reduction with two components (for being able to be displayed as a 2D plot and not because of an excessive amount of features). Additionally, for the more detailed clustering analysis, the PCA reduction yielded evaluation results identical to those obtained by employing the selected characteristics directly; nevertheless, since the PCA reduction enables the clusters to be shown in a two-dimensional scatter plot, it was decided to present the results with dimensionality reduction.

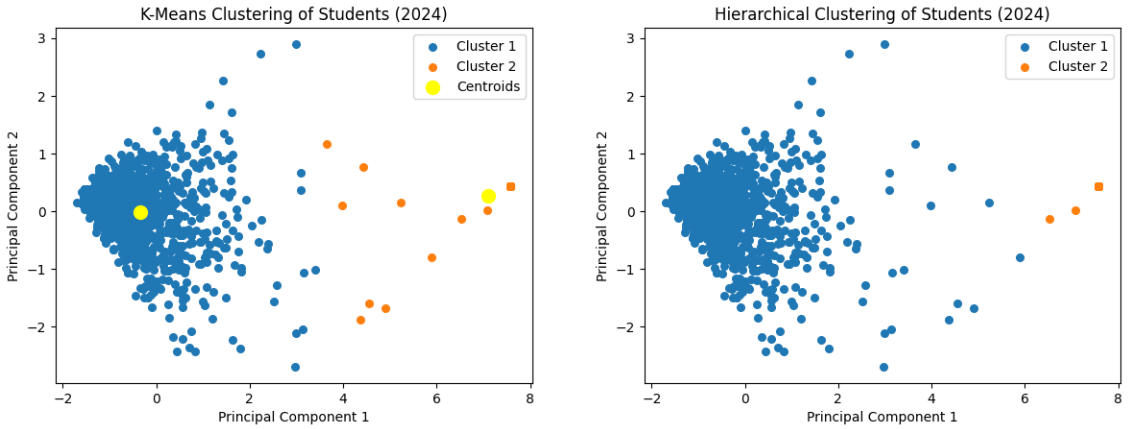


(a) Elbow method for K-Means clustering.

(b) Dendrogram for hierarchical clustering.

Figure 4.10: Methods used to determine the optimal number of clusters using each week total of activity points.

The clustering outcomes for both the KMC and Hierarchical clustering techniques were evaluated using the silhouette score (which ranges from -1 to 1), Davies-Bouldin Index (DBI, which shows how similar a cluster is with the closest cluster so best result corresponds to zero), and Calinski-Harabasz (CH) Index (which has no limit but a better



(a) K-Means clustering with two clusters.

(b) Hierarchical clustering with two clusters.

Figure 4.11: Clusters generated using each week total of activity points.

clustering corresponds to maximizing this index). In relation to 2024, a silhouette score of 0.8365, a DBI of 0.1227, and a CH Index of 2220.74 were obtained using the hierarchical clustering approach. These numbers imply that a high degree of cluster separation and a low degree of intra-cluster variation were attained by the hierarchical clustering. By contrast, the silhouette score was 0.8337, the DBI was 0.2295, and the CH Index was 2601.19 when using the KMC approach. Even with a marginally lower silhouette score than the hierarchical approach, KMC offers a good degree of cluster separation. When compared to hierarchical clustering, K-Means has a higher dispersion across the clusters, as indicated by its higher CH Index. Overall, both clustering approaches successfully identified two unique groups of students based on their weekly activity point totals. The hierarchical clustering method showed slightly better cluster compactness and separation, whereas the K-Means method had a higher total variation among the clusters.

In addition to the analysis in terms of student aggregation carried out in the `Data Analysis Report` Jupyter notebook file, a more extensive study was carried out in terms of clustering in the `Feature_selection_and_clustering` notebook, where, in addition to selecting the best attributes used to make specific predictions, we also took the analysis further in terms of clustering, starting looking into students cluster based on the total value of each type of assessment points. The KMC algorithm and Hierarchical Clustering are applied for multiple years, from 2016 to 2024, as well as for a complete dataset without missing data, and the results are displayed in Figures B.1a, B.2a, and B.3a as well as Table B.3. These display performance metrics like the silhouette Score, which calculates an object's similarity to its cluster relative to other clusters; the average similarity ratio between each cluster and the most similar cluster (DBI); and the ratio of the total of the dispersion within and between clusters (CH Index). With these results, it was constructed Tables 4.9 and 4.10 that identifies which algorithm and how many clusters it got better results according to each one of the evaluated metrics.

When using the full set of features with total of assessment points—`Total_AP`, `Total_PP`,

`Total_SPA`, and `Total_Bon`—K-Means consistently outperformed Hierarchical Clustering in most years, particularly in terms of the Silhouette Score and the CH Index (see Table 4.9). The Silhouette Score, which measures how similar an object is to its own cluster compared to other clusters, typically favored KMC, especially when the number of clusters was three or four. The CH Index, which evaluates the ratio of the sum of between-cluster dispersion to within-cluster dispersion, also generally favored K-Means. This indicates that K-Means was more effective in creating well-separated and compact clusters when multiple performance metrics were considered together and that the produced clusters were both distinct and internally cohesive, making KMC a preferable choice when selecting this features.

Table 4.9: Best results of clustering analysis when using `Total_AP`, `Total_PP`, `Total_SPA`, and `Total_Bon` as features.

Year	Silhouette Score	Davies-Bouldin Index (DBI)	Calinski-Harabasz Index (CH)
2016	K-Means (3 clusters)	Hierarchical (2 clusters)	K-Means (4 clusters)
2017	K-Means (3 clusters)	All similar results	K-Means (4 clusters)
2018	K-Means (3 clusters)	K-Means (3 clusters)	K-Means (4 clusters)
2019	K-Means (3 clusters)	K-Means (4 clusters)	K-Means (4 clusters)
2022	K-Means (2 clusters)	K-Means (2 clusters)	K-Means (2 clusters)
2023	K-Means (3 clusters)	K-Means (3 clusters)	K-Means (3 clusters)
2024	K-Means (3 clusters)	K-Means (3 clusters)	K-Means (4 clusters)
All Years	Hierarchical (3 clusters)	Hierarchical (3 clusters)	K-Means (4 clusters)

Conversely, when focusing on early activity data (the first two weeks of the course), Hierarchical Clustering provided more stable and consistent results, particularly when the goal was to identify a smaller number of clusters (typically two), as shown in Table 4.10. The Davies-Bouldin Index (DBI), which measures the average similarity ratio of each cluster with its most similar cluster, often favored Hierarchical Clustering in these scenarios. This suggests that Hierarchical Clustering was more effective at capturing broader, more fundamental distinctions in student behavior early in the course, such as differentiating between students who are engaged versus those who are not.

Table 4.10: Best results of clustering analysis when using the activity results from the first two weeks.

Year	Silhouette Score	Davies-Bouldin Index (DBI)	Calinski-Harabasz Index (CH)
2016	Hierarchical (2 clusters)	Hierarchical (2 clusters)	K-Means (2 clusters)
2017	Hierarchical (2 clusters)	Hierarchical (2 clusters)	Hierarchical (2 clusters)
2018	Hierarchical (2 clusters)	Hierarchical (2 clusters)	Hierarchical (2 clusters)
2019	Hierarchical (2 clusters)	Hierarchical (2 clusters)	Hierarchical (2 clusters)
2022	Hierarchical (2 clusters)	Hierarchical (2 clusters)	Hierarchical (2 clusters)
2023	K-Means (2 clusters)	K-Means (2 clusters)	K-Means (2 clusters)
2024	Hierarchical (2 clusters)	Hierarchical (2 clusters)	Hierarchical (2 clusters)
All Years	Hierarchical (2 clusters)	Hierarchical (2 clusters)	Hierarchical (2 clusters)

According to the presented results, the best clustering performance was achieved using K-Means (KMC) with three clusters for the individual yearly datasets, and KMC with two clusters for the combined dataset spanning multiple years. While the Silhouette Index remained above 0.7 for all individual datasets, it dropped to 0.567 for the combined dataset, indicating that the clusters are not as well-separated in this case. The Davies-Bouldin Index (DBI) also increased, suggesting a decline in clustering performance. Conversely, the Calinski-Harabasz Index (CH Index) showed an increase, which typically suggests better clustering. This discrepancy highlights the complexity of clustering in combined datasets and suggests that the performance metrics may not always align perfectly.

For more detail on the performed clustering analysis you can consult Subsection B.1.

4.2.5 Predictive Analysis of Student Performance

This subsection explores the potential of utilizing student performance data from the first two weeks within the CTCT course to predict outcomes in subsequent subject matters and overall academic success within this discipline dedicated to empower students with some soft skills. By analyzing historical data and applying advanced predictive models, this study aims to identify key indicators of student performance, providing insights that can enhance educational strategies. Additionally, given the varying computational resources available in Google Colaboratory, this work includes a detailed comparison of the computational efficiency and resource consumption across different processors. Although the primary focus of this dissertation is not on the predictive power of the models per se, the inclusion of this computational analysis offers valuable contributions

to the scientific community, guiding future research on the trade-offs between prediction accuracy, execution time, and resource usage in an educational data mining context.

To establish a baseline for predicting the `FinalGrade` and `Passed` attributes, logistic regression, linear regression, and multiple linear regression models were initially applied in the `Data_Analysis_Report` notebook described briefly in Section A.3. These models provided a preliminary understanding of the data's behavior and highlighted the potential challenges in prediction. The logistic regression analysis, focusing on `FinalGrade`, revealed significant class imbalances, resulting in low accuracy and poor precision-recall metrics across most grade levels. Similarly, the linear and multiple linear regression models exhibited low R^2 values, indicating weak predictive power and substantial deviations between predicted and actual values.

To improve the predictive analysis of student performance, individual datasets with chosen properties were constructed in the `Feature_selection_and_clustering` Jupyter Notebook. Different feature selection approaches, as listed in Table 3.2 of Chapter 3, served as a reference for the selection of these features. A number of important activities were consistently found to be significant predictors of student performance, including final grades, total activity points (`Total_AP`), participation points (`Total_PP`), and self and peer assessment points (`Total_SPA`). These activities included `SMART goals`, `Submission of CV on Moodle`, and `Wolis`. Some of these activities, such `Introduction to Google Calendar`, were only applicable in previous years, however they were chosen from a variety of approaches and years. When only the activity outcomes and participation scores from the first two weeks were used, the same activities were likewise thought to be significant in predicting course completion (`Passed`). However, it's important to remember that these feature selection techniques were used prior to scaling, in a different notebook, and without separating the training and test sets, thus there are certain limits in the results.

Even before feature selection techniques were used, the datasets were divided into three exclusion categories: `more_restricted`, `restricted`, and `less_restricted`. This was based on the features chosen for the analysis. Although it was decided to also make available a set with the results of all activities and a less restrictive group that includes all the features except the categorical ones, the first set was the most frequently used on the modeling techniques because it was in line with the research goals: finding out if the results of the first two weeks can give information about the end performance of the student. These improved datasets were then used to train and evaluate the resultant models.

The attributes `Total_AP`, `Total_PP`, and `Total_SPA`, which are originally integer-valued numeric variables within specific ranges, were analyzed both as continuous variables through regression and as discrete variables through classification. To perform the classification analysis, these continuous variables were discretized using a method that divides the target variable into a fixed number of bins. Specifically, the discretization was performed using the `convert_to_multiclass` function, which allows for different

binning strategies, such as uniform, quantile, and *k-means*. The *uniform* strategy, which divides the data into equal-width bins, was primarily utilized. This approach facilitated the conversion of continuous data into a multiclass format, suitable for classification tasks. On the other hand, the `FinalGrade` attribute, which is a continuous variable ranging from 0.0 to 20.0, was analyzed solely through regression techniques. It is important to note that `FinalGrade` contains missing values corresponding to instances where students scored below 10.0 in the `FinalGradeInteger` attribute, and these were considered during the analysis. Lastly, the `Passed` attribute, being a binary categorical variable, was treated accordingly, with the analysis focusing exclusively on classification models.

The results of the classification analysis for these variables will be presented in the following subsection, while the regression results will be detailed in the subsequent subsection. This structure ensures a clear and organized presentation of the findings from both modeling approaches.

4.2.5.1 Classification Results

To evaluate the effectiveness of various models in predicting student performance, a series of classification tasks were performed on different target variables: `Total_AP`, `Total_PP`, `Total_SPA`, and `Passed`. The first three variables were treated as categorical by discretizing continuous values into classes, as described earlier. The models were trained using different imputation strategies, and their performance was measured by accuracy, F1-score, and execution time. The hardware used for each task using Google Colaboratory resources is also reported, providing context for the computational efficiency of each approach. In this context, two types of GPUs were used: the `L4`, a high-performance GPU optimized for inference workloads, and the `T4`, a versatile GPU suitable for both training and inference tasks. For more detailed information about Google Colaboratory and its features, please refer to their official page [36].

Table 4.11 presents the best classification results after discretization. The term "None" in the "Imputation - Feature Selection Methods" column indicates that columns with missing data were removed before applying the classification models. As shown, the Random Forest (RF) model combined with mean imputation achieved the highest accuracy of 0.8466, with an F1-score of 0.8418. Execution times varied significantly depending on the hardware used, with GPU-based execution being substantially faster than CPU-based processing in most applications.

Table 4.11: Best classification results for `Total_AP` after discretization.

Imputation - Feature Selection Methods	Best Model	Accuracy	F1-Score	Execution Time (seconds)
None - None	Random Forest (RF)	0.823	0.814	415 (CPU)
None - None	RF	0.821	0.811	545 (GPU T4)
None - None	RF	0.821	0.811	491 (GPU L4)
None - MI	Neural Network (NN)	0.833	0.833	501 (GPU L4)
None - RF	NN	0.833	0.833	1568 (CPU)
None - RF	NN	0.833	0.833	1054 (GPU L4)
None - DT	NN	0.833	0.833	503 (GPU L4)
None - LASSO	RF	0.781	0.768	545 (GPU L4)
None - FR	NN	0.795	0.775	500 (GPU L4)
Mean - None	**RF**	**0.847**	**0.842**	1283 (CPU)
Mean - None	**RF**	**0.847**	**0.842**	921 (GPU T4)
Median - None	RF	0.843	0.837	1346 (CPU)
Median - None	RF	0.843	0.837	975 (GPU T4)
Median - MI	RF	0.842	0.836	1245 (GPU T4)
kNN - None	RF	0.846	0.842	1268 (CPU)
kNN - None	RF	0.846	0.842	891 (GPU T4)
Zero - None	Logistic Regression (LR)	0.842	0.839	1151 (CPU)
Zero - None	LR	0.842	0.839	881 (GPU T4)

For the classification of `Total_PP`, the best results were achieved using an Artificial Neural Network (ANN) model with zero imputation, yielding an accuracy and F1-score of 0.756 and 0.754, respectively. Detailed results for `Total_PP` classification are provided in Table B.5 in Appendix B. Regarding the execution times, one of the exceptions in which CPU code run took less time then when using GPU can be seen in the first two lines of this table.

For the classification of `Total_SPA`, Random Forest (RF) delivered excellent results, achieving an accuracy of 0.9797 and an F1-score of 0.9788 without any imputation. These results were obtained with a relatively short execution time of 414 seconds using a CPU. When executed on a GPU, the execution time was increased to 658 seconds when no

imputation was performed. Detailed classification results for `Total_SPA` can be found in Table B.6.

For the prediction of whether a student passed the course (`Passed`), Random Forest (RF) produced good results but Logistic Regression (LR) performed even better, achieving an accuracy and a F1-score of 0.9975 with mean imputation and Mutual Information (MI) feature selection. The fastest execution time of 72.92 seconds was observed on an L4 GPU with RF, no imputation strategy and no feature selection, with a low increased performance from when used the T4. Actually for this simple model, a CPU would be enough. A detailed breakdown of the classification results for `Passed` is available in Table B.7.

4.2.5.2 Regression Results

The regression tasks focused on predicting continuous outcomes for `Total_AP`, `Total_PP`, `Total_SPA`, and `FinalGrade`. Various imputation strategies were applied, and models were evaluated based on metrics such as mean squared error (MSE), R-squared value (R^2), and mean absolute error (MAE). Execution time and hardware usage were also recorded to assess computational efficiency.

Table 4.12 shows the best regression results for predicting `Total_AP`. The Random Forest model with mean imputation yielded a MSE of 139.75, an R^2 value of 0.817, and a MAE of 6.68. Notably, the median imputation method slightly outperformed this with a MSE of 135.49 and an R^2 of 0.8225. But when using the MI feature selection, these evaluation metrics got even better: 105.21 for MSE, 0.8622 for R^2, and 5.94 for MAE. Execution times varied significantly depending on the processing unit, with the T4 GPU consistently offering faster processing times compared to the CPU, reducing the computational time from 1283.34 seconds on the CPU to 920.94 seconds on the T4 GPU for the Random Forest model with mean imputation.

Table 4.12: Best regression results for `Total_AP` prediction.

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
None - None	Random Forest (RF)	175	0.771	415 (CPU)
None - None	RF	175	0.771	545 (GPU T4)
None - None	RF	175	0.771	491 (GPU L4)
None - MI	Neural Network (NN)	153	0.800	501 (GPU L4)
None - RF	NN	153	0.800	1568 (CPU)

Continued on next page

Table 4.12: (continued)

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
None - RF	NN	207	0.729	1054 (GPU L4)
None - DT	NN	153	0.800	503 (GPU L4)
None - LASSO	RF	212	0.722	545 (GPU L4)
None - FR	NN	207	0.729	500 (GPU L4)
Mean - None	RF	140	0.817	1283 (CPU)
Mean - None	RF	140	0.817	921 (GPU T4)
Median - None	RF	135	0.823	1346 (CPU)
Median - None	RF	135	0.823	975 (GPU T4)
kNN - None	RF	141	0.816	1268 (CPU)
Median - MI	**RF**	**105**	**0.862**	1245 (GPU T4)
kNN - None	RF	141	0.816	891 (GPU T4)
Zero - None	RF	139	0.818	1151 (CPU)
Zero - None	RF	139	0.818	881 (GPU T4)

For the regression analysis of `Total_PP`, Support Vector Machine (SVM) with median imputation yielded a good performance, achieving a mean squared error (MSE) of 7.81 and an R^2 value of 0.747. The execution time was 682 seconds on a T4 GPU. Similarly, SVM with zero imputation slightly improved the R^2 value to 0.761 with an MSE of 7.38, demonstrating the model's versatility across different imputation strategies. A detailed comparison of imputation strategies and models for `Total_PP` can be found in Table B.8 in in Appendix B, where it's shown that the best result was obtained with imputation by zero and both MI and DT feature selection techniques.

For the regression analysis of `Total_SPA`, the Support Vector Machine (SVM) model with zero imputation was the second most accurate, achieving an MSE of 18.40 and an R^2 value of 0.8708. The inclusion of decision tree (DT) feature selection slightly improved the R^2 to 0.8724, with a marginally lower MSE of 18.17. These results indicate that SVM is an effective model for predicting `Total_SPA` with minimal preprocessing. For detailed regression results on `Total_SPA`, refer to Table B.9.

For the prediction of `FinalGrade`, simpler models such as Support Vector Machine (SVM) with zero imputation performed well, yielding an MSE of 1.47 and an R^2 value of 0.514 when executed on a T4 GPU. The performance was consistent across different hardware setups, with similar results observed on a CPU. In comparison, Linear Regression

showed limitations, with a higher MSE of 1.84 and a lower R^2 of 0.391. Detailed results for simpler regression models predicting `FinalGrade` are available in Table B.10.

Given that the `FinalGrade` remains consistent across different academic years and is a critical measure of student performance, more advanced ensemble algorithms were explored to assess their predictive power specifically for this target variable. Unlike other variables, such as `Total_AP`, `Total_PP`, and `Total_SPA`, which may vary significantly from year to year due to changes in course content or structure, the `FinalGrade` offers a stable target for evaluation. This consistency makes it particularly suitable for testing the effectiveness of complex models like CatBoost and XGBoost. Although the other target columns were more challenging to predict due to their variability, focusing on `FinalGrade` allows for a clearer comparison of model performance. Moreover, while not implemented due to time constraints, a potential future enhancement involves using a pipeline approach where models first predict whether a student is likely to pass and then use this subset of students to predict the final grade. This strategy could refine the predictive accuracy further and provide deeper insights into student performance.

Table 4.13 presents the regression performance for predicting `FinalGrade` using advanced ensemble methods. The CatBoost model consistently delivered superior results, particularly when combined with feature selection methods like Random Forest (RF) and LASSO. Specifically, CatBoost with mean imputation and RF feature selection achieved a MSE of 1.39 and a R^2 value of 0.822, significantly outperforming simpler models. The same results were obtained using a T4 and a A100 GPU but the execution time decrease from 647 to 436 seconds.

Table 4.13: Best regression results for `FinalGrade` prediction using advanced ensemble methods.

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
Zero - RF	CatBoost	1.41	0.822	647 (GPU T4)
Zero - LASSO	CatBoost	1.41	0.822	687 (GPU T4)
Zero - FR	XGBoost	1.42	0.819	603 (GPU T4)
Mean - RF	**CatBoost**	**1.39**	**0.822**	647 (GPU T4)
Mean - LASSO	**CatBoost**	**1.39**	**0.822**	684 (GPU T4)
Zero - None	CatBoost	1.68	0.443	170 (GPU L4)
Mean - None	CatBoost	1.43	0.527	351 (GPU L4)
Median - None	CatBoost	1.42	0.528	430 (GPU L4)

Continued on next page

Table 4.13: (continued)

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
Zero - None	CatBoost	1.44	0.524	614 (GPU L4)
kNN - None	CatBoost	1.42	0.530	462 (GPU L4)
Most Frequent - None	CatBoost	1.42	0.528	390 (GPU L4)
Constant - None	CatBoost	1.44	0.524	604 (GPU L4)
Zero - RF	CatBoost	1.41	0.820	585 (GPU A100)
Mean - RF	**CatBoost**	**1.39**	**0.822**	436 (GPU A100)
kNN - RF	Stacking	1.43	0.818	468 (GPU A100)

This chapter presented a detailed analysis of educational data from the CTCT course, highlighting the dynamics of student performance over several academic years. The descriptive and statistical analysis revealed significant patterns, such as the variation in activities and their influence on student performance, as well as differences in performance between genders and courses. Furthermore, important relationships were identified between different types of assessment points and final grades, allowing a deeper understanding of academic success metrics. In the following chapter, the results obtained will be discussed in greater depth, exploring the implications of these findings and the conclusions that can be drawn, with a special focus on the practical application of the analyzes and the potential future impacts on the educational process.

5

Discussion

This chapter explores the educational implications of the results and the technological insights gained from applying machine learning methodologies. Additionally, this chapter addresses the limitations encountered during the research and provides recommendations for future work.

Several insights into the CTCT course can be drawn from the statistical analysis performed. The heatmap in Figure 4.1 gives a wider perspective of the evolution of this discipline along the studied years in terms of the change in the number of each type of activity (oral or written, individual or in group, homework or class activity, for all students or only for some). Some activities were dropped, others were added, and some changed the considered typology. For example, before 2020, participation points were primarily awarded based on attendance, punctuality, and classroom behavior. However, since 2020, these points have been increasingly awarded for active participation in specific volunteer activities during class, which were previously counted as activity points.

Descriptive statistics in Table 4.1 reveal fluctuations in `Total_AP`, `Total_PP`, and `Total_SPA` across the years. An increase in `Total_AP` from 2022 to 2023 and the decrease in `Total_SPA` post-2019 indicates a recalibration of each type of assessment point contribution to final grades. And the analysis of success rates and final grades shows year-to-year variability, with the highest success rate of 97.38% in 2018 and the lowest rate of 92.88% in 2017. Also, the consistent negative skewness and high kurtosis in final grade distributions, especially in 2017 and 2022, suggest a concentration of higher-performing students or a grading curve that favors the upper end of the scale.

These educational implications provide a backdrop for understanding the specific research questions addressed in this study.

5.1 Research Questions

Impact of Gender on Course Selection (RQ1)

The chi-squared tests conducted in this study reveal a statistically significant association between gender and course enrollment, with p-values consistently close to zero, as shown

in Table 4.2. These findings suggest that male and female students tend to choose distinct academic fields, reflecting broader societal patterns. For example, male students are more likely to enroll in courses related to electrical, mechanical, and informatics engineering, while female students show a greater preference for fields such as biology and chemistry. This gender-based division in course selection is a concerning trend, as it reinforces the existing gender gap in STEM disciplines, potentially limiting the diversity of perspectives and innovation within these fields. Addressing this issue requires further exploration into the root causes of these disparities, enabling more equitable educational outcomes.

Interrelations Among Assessment Points and Final Grade (RQ2)

As seen in Table 4.4, the analysis consistently showed high positive correlations between `Total_AP` and final grades, with coefficients ranging from 0.75 to 0.84 across different years. Since Activity Points (AP) are the points that represent student success on activities, it stands to reason that they would be a stable and important part of the course's assessment framework given the constancy of these relationships over several years.

Participation Points (PP) also showed a positive correlation with final grades, although the strength of this relationship varied more than that of AP. The correlation between `Total_PP` and final grades ranged from 0.57 to 0.81, indicating that while participation is a valuable component of the learning process, its impact on final grades is somewhat less pronounced than that of AP. The way participation is measured or valued might change depending on the specific class content or the instructor's approach, leading to fluctuations in its correlation with final grades.

In contrast, the relationship between `Total_SPA` and final grades displayed greater variability across years, with correlations ranging from 0.33 in 2017 to 0.59 in 2022. This variability suggests that the influence of Self and Peer Assessment Points (SPA) on final grades has shifted over time, potentially reflecting changes in how these assessments are weighted or perceived within the course. The increase in correlation in more recent years could suggest that SPA has become more significant in the grading process, contradicting the intuition given by the decrease of the total number of SPA (from more than 35 before 2020 to less than 12 as of 2022). However, the moderate correlation also implies that while SPA contributes to final grades, it does so to a lesser extent than more direct assessments like AP or PP.

Bonification Points (`Total_Bon`), on the other hand, exhibited negative correlations with final grades, what means that this type of points are given to students with lower grades. This raises questions about the effectiveness of this points in motivating students toward meaningful academic outcomes. Future considerations might include re-evaluating the role of Bonification Points within the course structure or exploring alternative ways to use them that could enhance their relevance to academic success.

Variability in Student Performance (RQ3)

Even though there are more male students enrolled, Figure 4.5 and Table 4.5 show that male students often have greater failure rates than female students. This pattern holds true for the majority of academic years and raises the possibility that male students are disproportionately affected by underlying variables, such as variations in learning styles or a different view of the value of soft skills in career development.

The Mann-Whitney U test results in Table 4.6 further highlight gender differences in academic performance, particularly in courses such as Informatics Engineering (LEI) in 2017 and Electrical and Computer Engineering (LEEC) in 2016. However, since these gender differences did not consistently appear in the same courses across different years, it suggests that the observed disparities may be tied to the specific group of male and female students in those courses during a given year, rather than indicating a recurring trend. This implies that the differences in performance could be influenced by the unique characteristics of students in a particular year rather than systemic issues within the courses in wich students are enrolled.

The weekly performance analysis in Table 4.7 reveals significant variability in student performance across different thematic areas and weeks. For instance, the advanced Excel use week consistently shows high correlations with final grades, indicating that this technical skill is a critical component of student success in the CTCT discipline. This was expected, especially in recent years where the number of Excel-related activities has increased. This week's activities include mostly type C evaluations, which offer a broader range of classifications, allowing for a more differentiated assessment of student performance.

The correlation matrices in Figures 4.7 and 4.8 suggest that while there is a strong correlation between weekly activity points and final grades across all students, this relationship becomes more nuanced when examining only the students who pass the course. The stronger overall correlation can be attributed to the fact that students who fail the course tend to miss a significant number of classes, leading to consistently low scores across all metrics. In contrast, among students who pass, performance tends to vary more depending on the specific thematic areas of the course. This variation reflects different strengths among passing students, with some performing better in certain topics like leadership or technical skills, while others excel in areas like time management or ethics. As a result, the correlation between weekly activity points and final grades is less uniform in this group, as different students accumulate points in different ways depending on the thematic focus of each week. This underscores the importance of both attendance and thematic engagement as key predictors of overall success in the course.

Also, courses with consistently higher performance across weeks, as shown in Table 4.7, tend to align with those exhibiting lower failure rates (see Table 4.5).

Identifying Patterns in Student Performance (RQ4)

In terms of optimal number of clusters, the majority of the results confirmed expectations: the best division for the majority of feature combinations experimented produced better results for two clusters. This is the case for the clustering analysis performed on the results using the first two weeks' activity results as features, as shown in Table 4.10. However, when applying clustering to the engineered features with the sum of points of each type, the best optimal number of clusters increases to three or four, depending on the year, as shown in Table 4.9. The exception was the COVID year, which also have two as the best number of clusters (but that year there were only 3 weeks of classes).

K-Means Clustering (KMC) consistently outperformed hierarchical clustering when using engineered features related to the total number of each type of assessment point. K-Means was more effective at creating well-separated and cohesive clusters, as evidenced by slightly higher silhouette scores (and Calinski-Harabasz indices, which indicate contradictory information). However, when focusing on early activity data, hierarchical clustering produced more stable and consistent results, particularly when identifying fewer clusters. The Davies-Bouldin Index frequently preferred Hierarchical Clustering in these cases, indicating that clusters were more compact when distinguishing between engaged and non-engaged students early in the course. These findings highlight the importance of selecting the appropriate clustering method based on the data's characteristics and the specific goals of the analysis.

Predictive Value of Early Performance on Later Outcomes (RQ5)

While Tables from 4.11 to 4.13 and from B.5 to B.10 detail the top-performing models across several Machine Learning (ML) analyses, with the best-trained models highlighted and corresponding performance metrics provided, Table 5.1 offers a concise summary. This summary outlines the best algorithms, their optimal imputation strategies, and the most effective feature sets used in the analysis. These feature sets include both the results from the first two weeks of the course and those selected through feature selection techniques such as Mutual Information (MI), Least Absolute Shrinkage and Selection Operator (LASSO), Random Forest (RF), Decision Tree (DT), and F-regression as a scoring function in SelectKBest (FR).

The most effective model for classifying `Total_AP` after discretization was a Random Forest (RF) with mean imputation and no feature selection. This model achieved an accuracy of approximately 0.84 when evaluated using a 5-fold cross-validation, which is comparable to the F1-score. The most optimal regression model was also the RF model and were achieved by using median imputation and MI feature selection.

The classification of `Total_PP` yielded the most favorable outcome through the utilization of a Neural Network (NN), following the imputation of missing data with a value of zero and employing MI feature selection. In regression, the Support Vector Machine

Table 5.1: Summary of predictive modeling results for different target columns.

Target Column	Model (Type)	Best Imputation Strategy	Best Feature Selection	Metric (Score)
Total_AP	RF for Classification	Mean	MI	Accuracy (0.847)
	RF for Regression	Median	MI	MSE (105)
Total_PP	ANN (Classification)	Zero	MI	Accuracy (0.756)
	Support Vector Machine (SVM for Regression)	Zero	MI / DT	MSE (6.6)
Total_SPA	SVM / RF for Classification	None / Mean	None	Accuracy (0.980)
	SVM for Regression	Zero	DT	MSE (18.2)
Passed	LR for Classification	Mean	MI	F1-Score (0.994)
FinalGrade	CatBoost (Regression)	Mean	RF	MSE (1.39)
	SVM for Regression	Zero	None	MSE (1.47)

(SVM) was utilized, employing the identical imputation and feature selection techniques as the top-performing classification model.

Following discretization, the classification of `Total_SPA` produced comparable outcomes to the SVM approach when employing median, k-Nearest Neighbors (kNN), or zero imputation techniques without feature selection. Additionally, the RF model was identified as the most optimal choice when utilizing zero imputation and DT feature selection results. The most accurate predictions for column values using regression were achieved with SVM when zero imputation was employed.

In order to mitigate the risk of overfitting, the number of folds on cross-validation was increased from five to ten when classifying students as going to pass or fail CTCT, because `Passed` classification yielded exceptionally good outcomes. At a value of 10 folds, consistent with the results presented in Table B.7, the Accuracy and F1-score for Logistic Regression (LR), using mean imputation and feature selection through MI, were remarkably high. The best outcome achieved was an accuracy of 99.8%. It is important to acknowledge that while the success rate in this field is high, an algorithm that predicts every student will pass will have a slightly lower categorization (between 92.88% and 97.38% for the available data). Therefore, these findings may not meet the expected level of quality.

The prediction of the `FinalGrade` was conducted using the same method applied to the other columns. The most favorable outcome was achieved by employing the SVM algorithm, with the imputation of missing data as zero and without any feature selection. In addition, other two methods were employed for prediction, utilizing more

complex algorithms like CatBoost, XGBoost, and Stacking. The most optimal results were achieved through the utilization of CatBoost with mean imputation and feature selection. Interestingly, the same outcomes were observed when employing RF and LASSO techniques.

In general, early performance metrics offer valuable predictive insights, but the effectiveness of the models can vary depending on the specific outcomes being predicted. Further research is necessary to address the limitations in the available data, specifically the lack of failing student grades. This highlights the importance of developing more comprehensive predictive frameworks. One potential approach could involve a two-step pipeline, where a model initially classifies students as either "Passed" or "Failed" before making predictions about their final grade. The motivation behind enhancing the prediction results of `FinalGrade` involved the utilization of more sophisticated ensemble models. Unfortunately, due to time constraints, the implementation of the pipeline was not carried out. However, it is worth noting that this research strongly recommends its inclusion as a key aspect for future work aimed at enhancing the analysis of this data.

Technological Insights

The study analyzed the performance of several models using different Google Colaboratory hardware configurations: Central Processing Unit (CPU), T4 Graphical Processing Unit (GPU), L4 GPU, and A100 GPU. Execution times and accuracy scores were recorded across multiple tasks, revealing notable variations between hardware resources available for this tool, as seen in Tables 4.11, B.5, B.6, and B.7.

The expected performance improvements from CPU to GPU were observed in most cases. For instance, models executed on the T4 and L4 GPUs generally ran faster than those on CPUs, particularly for tasks involving neural networks and decision trees. However, there were anomalies in some runs. For example, in Table 4.11, without imputation or feature selection, the GPU T4 took longer to execute than the CPU, highlighting potential inconsistencies due to varying system loads, caching effects, or even minor code changes such as debug statements. These factors can skew results and should be considered when analyzing execution times.

The regression tasks, summarized in Tables 4.12, B.8, B.9, and B.10, emphasized the balance between computational efficiency and cost. For instance, using a Random Forest model with mean imputation and no feature selection, the task took approximately 647 seconds when using ensemble methods on the A100 GPU, compared to 435 seconds using only the A100 GPU without feature selection. These results demonstrate the substantial time savings offered by more powerful hardware like the A100 GPU, particularly for ensemble-based models, which are computationally expensive.

Advanced ensemble methods, as detailed in Table 4.13, further underscore the trade-offs between execution time and model costs. For example, CatBoost with mean imputation and Random Forest feature selection achieved an execution time of 647 seconds on the

T4 GPU at 1.76 compute unit (cu) per hour and 436 seconds on the A100 GPU that costs 11.76 cu. But even the same code executed twice don't have the same time, as you can see by the results of *Constant - None* and *Zero - None* that actually correspond to the same code because the constant was defined as zero, with a difference of 10 seconds between runs.

The choice of hardware also has significant cost implications. As documented in this study, Google Colaboratory charges 0.07 cu per hour for CPU usage, approximately 1.76–1.82 cu per hour for T4 GPUs, 4.82 cu per hour for L4 GPUs, and 11.76 cu per hour for A100 GPUs. Considering that 100 cu cost 11.38 euros, researchers must balance their computational needs with their budget constraints. Although the A100 GPU delivers substantial power, it may not be cost-effective for less complex tasks, where cheaper alternatives such as the T4 or L4 GPUs could suffice.

For most of the tasks in this research, the T4 and L4 GPUs provided a suitable balance between cost and performance, especially when running the entire notebooks or conducting comprehensive testing. However, for testing individual models or methods, the CPU may be the most appropriate option due to its lower cost and sufficient computational power for smaller-scale tasks. Ultimately, the decision to use advanced GPUs like the A100 should depend on the complexity and scale of the project at hand.

5.2 Limitations

One drawback in evaluating performance variability was the anonymization of datasets that occurred prior to this research. This anonymization prevented the tracking of individual student progress across multiple course editions, making it impossible to assess their development over time. Further research should explore methods of connecting anonymized data or utilizing ethically sourced non-anonymized data to conduct a thorough analysis of performance trends over time. This analysis should specifically examine course repetition rates and their influence on student outcomes.

One additional limitation of this analysis was the feature selection process used in `Feature_selection_and_clustering` notebook, which was conducted prior to scaling and without the separation of training and test sets. This approach may have introduced bias into the results of the selection itself, but the prediction algorithms also were used without prior attribute selection. Further investigation is warranted to examine how various techniques for selecting and scaling features can improve the reliability and applicability of correlation analysis results. In addition, further research could explore the effects of changes in self and peer-assessment criteria on student outcomes over time.

This research encountered two significant challenges in data processing, which led to the development of a sophisticated Python dictionary containing the essential information for accurately loading each received file. To establish a correlation between the activities of different years, a thorough comparison of each year's activities and corresponding documentation was required. This involved manually reviewing the Excel files to ensure consistency in column nomenclature, identifying any empty rows or columns at the

beginning of the worksheet, and determining which worksheet needed to be processed. If the data had a consistent format, the second part would not have been necessary. This consistency is ensured when using a Learning Management System (LMS) like Moodle to extract the data. This LMS is the one adopted in the Faculty of Science and Technology of NOVA (FCT) and adheres to the Shareable Content Object Reference Model (SCORM) standards and it allows the development of plugins to customize pages and automate teaching-related administrative tasks, as stated in Subsection 2.2.2.

Regarding the evaluation of Google Colaboratory hardware resources, the analysis in this research was limited, as it was not the primary focus of the project. While efforts were made to ensure consistency by running the same code across different configurations, minor variations, such as debug print statements, may have affected some of the results. Ideally, the hardware comparison should have employed simpler methods, rather than complex routines that test multiple algorithms with varying hyperparameters. Furthermore, both the classification and regression results were obtained using the same framework, leading to identical execution times, which prevents a meaningful comparison of how each algorithm performs in these different contexts. A more detailed exploration of whether a regressor requires more or less computational time compared to the same algorithm in a classifier role would have added value to this aspect of the study.

6

Conclusion and Future Work

This research investigated several aspects to address the research questions, such as the impact of gender on course selection at FCT (Research Question 1). It was demonstrated by Chi-squared tests that gender does indeed influence course selection. This suggests that societal norms and gender-based perceptions of specific fields can impact these choices, highlighting the importance of further research to enhance educational outcomes.

The relation between assessment points and final grades in terms of performance (RQ2) was studied. Activity points had the strongest correlation with final grades, but self and peer assessment (SPA) had the worst correlation. Furthermore, the fluctuating correlations of SPA throughout different years suggest a shift in the weighting of these points in the final grade. Also, the negative correlation of bonification points with final grade was verified, as expected. These points are awarded only for students with lower grades, so it makes sense.

This dissertation investigated the variation in student performance based on demographic factors and different versions of the course (Research Question 3). In all academic years, male students exhibited a higher rate of failure compared to their female counterparts, while having a larger enrollment for all available datasets. For example, for 2024, about 80% of the students who failed CTCT were male but only 65% of the students enrolled were male. The significant variation in weekly performance among different thematic areas underscores the need of comprehending the multiple factors that influence student achievement. In 2024, it was observed that students pursuing a degree in geological engineering (LEG) had lower performance in advanced spreadsheet activities. This emphasizes the need for targeted interventions to assist students who are facing difficulties in essential skill areas.

The clustering analysis (Research Question 4) demonstrated that student performance data can be utilized to identify trends within the CTCT program. The study analyzed both the overall assessment points and weekly activity data using the K-Means and Hierarchical Clustering techniques. Two clusters consistently exhibited the most significant differentiation, particularly when examining data from the initial two weeks of this

program. The Silhouette Scores of Hierarchical Clustering were greater, but the Davies-Bouldin Index values were lower, suggesting better cluster compactness and separation. The K-Means algorithm exhibited greater variability and had higher success rates when using the entire assessment points as features.

The machine learning models used in this study demonstrated the predictive power of early student performance data, with the Random Forest and Support Vector Machine models consistently outperforming the others. CatBoost also exceeded other ensemble algorithms when it came to predicting a student's final grade. However, the absence of failing student data limits the predictive models' scope, highlighting the need for more robust frameworks in future research.

Technologically, this study showed the value of Google Colaboratory for educational data mining. Research suggests that mid-range GPUs, such as the L4 and T4, offer a cost-effective balance of performance, making them suited for large-scale educational analysis.

Datasets with variable formats and structures were a major issue during the study. To load and process data for meaningful analysis, each dataset needed rigorous review and specific treatment. Lack of uniformity increased data processing complexity and time and introduced potential errors that could impair result reliability.

In addition to the knowledge and data gathered during this project, a web application was created that can be used to store CTCT materials and collect quiz answers. It was specifically modeled in terms of users for the roles that are present in this course. The VBA (Virtual Basic Applications) code developed can also be used in subsequent course editions to help teachers correct CTCT Excel activities.

Future Research Directions

Future research could investigate a more exhaustive study of code execution durations and expenses in Google Colaboratory, particularly across different computational resources. Executing the same code multiple times would enable the evaluation of fluctuations in performance and expenses, providing a more profound understanding of resource optimization.

An improvement suggested for this research is to forecast the `FinalGrade` by initially categorizing students as either passing or failing and then employing a distinct model to predict the final grade for those who pass. Although this strategy was not put into practice because of time limitations, it has the potential to produce more useful predictions, truly forecasting the grade of the student with results from the first two weeks.

This study showcases the capacity of educational data mining to reveal useful insights, even when working with incomplete data. It emphasizes the significance of improving data collection and analysis techniques. Through the creation and application of customized plugins, it will become feasible to assess student involvement and achievement instantly, facilitating more focused and prompt interventions. Taking advantage of

faculty's technology skills, Moodle's open-source platform is a good option to develop customized solutions for every discipline. This integration would facilitate the process of gathering and analyzing data, enabling teachers to focus on comprehending the outcomes and implementing data-driven enhancements to curriculum design and teaching methods. Integrating Moodle into the data mining workflow at FCT can greatly enhance the productivity, accuracy, and impact of educational analytics.

Moreover, the integration of Artificial Intelligence (AI) and Learning Analytics (LA) in education, particularly in higher education (HE), holds significant potential for reshaping the learning experience. AI can drive more personalized learning environments, where students receive tailored content, feedback, and support. Nevertheless, challenges such as ethical dilemmas, security concerns, and the need for skilled professionals must be considered. Future research can address these issues by creating more transparent, resource-efficient, and privacy-conscious predictive models that enhance student success while maintaining ethical integrity.

By advancing data-driven educational practices, this dissertation contributes to the wider field of AI in education, particularly in engineering and technical education. The findings not only provide insights into the CTCT course and its demographic variations but also offer a foundation for refining teaching methodologies, improving student engagement, and optimizing educational outcomes in line with the vision of Society 5.0.

This research also highlights the importance of transversal skills for engineers and scientists, emphasizing the role they play in enhancing the soft skills required in STEM fields. It is essential to foster these skills throughout technical courses, as they are essential for effective communication, problem-solving, and collaboration in real-world scenarios. Further efforts in educational data mining can contribute to a deeper understanding of how transversal skills can be integrated and enhanced in engineering education, fostering a more holistic approach to student development.

The application of educational data mining methods to engineering education, as seen in this research, is just the beginning. As more data becomes available, the potential for improving educational outcomes through data-driven insights will continue to expand. The journey towards optimizing engineering education through data mining, technology integration, and soft skill enhancement is an ongoing process that will shape the future of education in the context of a rapidly evolving technological landscape.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] C. Akturk, T. Talan, and C. C. Cerasi. "Education 4.0 and University 4.0 from Society 5.0 Perspective". In: *2022 12th International Conference on Advanced Computer Information Technologies (ACIT)*. Ruzomberok, Slovakia: IEEE, 2022-09, pp. 577–582. DOI: 10.1109/ACIT54803.2022.9913099. URL: https://ieeexplore.ieee.org/document/9913099/ (cit. on pp. xviii, 1).

[3] C. Narvaez Rojas et al. "Society 5.0: A Japanese Concept for a Superintelligent Society". In: *Sustainability* 13.12 (2021-06), p. 6567. DOI: 10.3390/su13126567. URL: https://www.mdpi.com/2071-1050/13/12/6567 (cit. on pp. xviii, 1).

[4] D. Wang and et al. "A problem solving oriented intelligent tutoring system to improve students' acquisition of basic computer skills". In: *Comput. Educ.* 3 (2015), pp. – (cit. on p. 6).

[5] P. Manickam et al. "Artificial Intelligence (AI) and Internet of Medical Things (IoMT) Assisted Biomedical Systems for Intelligent Healthcare". en. In: *Biosensors* 12.8 (2022-07), p. 562. ISSN: 2079-6374. DOI: 10.3390/bios12080562. URL: https://www.mdpi.com/2079-6374/12/8/562 (visited on 2023-07-07) (cit. on pp. 6, 8, 11).

[6] X. Gong. "Educational Artificial Intelligence (EAI) Connotation, Key Technology and Application Trend". In: IEEE, 2021. DOI: 10.1109/ICAA53760.2021.00046 (cit. on pp. 6, 8, 9).

[7] J. McCarthy. "WHAT IS ARTIFICIAL INTELLIGENCE?" en. In: (2007-11) (cit. on p. 6).

[8] European Commission. Joint Research Centre. *AI watch, defining artificial intelligence 2.0: towards an operational definition and taxonomy for the AI landscape.* en. LU: Publications Office, 2021. URL: https://data.europa.eu/doi/10.2760/019901 (visited on 2024-05-08) (cit. on pp. 6–8, 11, 12).

[9] S. Legg and M. Hutter. *A Collection of Definitions of Intelligence*. en. Tech. rep. arXiv:0706.3639. arXiv:0706.3639 [cs]. arXiv, 2007-06. URL: http://arxiv.org/abs/0706.3639 (visited on 2024-05-08) (cit. on p. 6).

[10] ENISA. *ARTIFICIAL INTELLIGENCE AND CYBERSECURITY RESEARCH ENISA Research and Innovation Brief*. ENISA Research and Innovation Brief. EDITORS Corina Pascu (ENISA), Marco Barros Lourenco (ENISA) AUTHORS Dr. Stavros NTALAMPIRAS, University of Milan, I; Dr. Gianluca MISURACA, Co-Founder and VP, Inspiring Futures, ES; Dr. Pierre Rossel, President at Inspiring Futures CH. European Union Agency for Cybersecurity (ENISA), 2023. URL: https://www.enisa.europa.eu/publications/artificial-intelligence-and-cybersecurity-research (visited on 2023-06-11) (cit. on pp. 6–8, 11–13).

[11] M. Chassignol et al. "Artificial Intelligence trends in education: a narrative overview". en. In: *Procedia Computer Science* 136 (2018), pp. 16–24. ISSN: 18770509. DOI: 10.1016/j.procs.2018.08.233. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050918315382 (visited on 2023-07-04) (cit. on pp. 7, 9).

[12] European Union Agency for Cybersecurity. *A multilayer framework for good cybersecurity practices for AI :: security and resilience for smart health services and infrastructures.* en. LU: Publications Office, 2023. URL: https://data.europa.eu/doi/10.2824/588830 (visited on 2023-07-03) (cit. on pp. 7, 8).

[13] L. Chen, P. Chen, and Z. Lin. "Artificial Intelligence in Education: A Review". en. In: *IEEE Access* 8 (2020), pp. 75264–75278. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2988510. URL: https://ieeexplore.ieee.org/document/9069875/ (visited on 2023-07-03) (cit. on pp. 7, 9).

[14] F. AlShaikh and N. Hewahi. "AI and Machine Learning Techniques in the Development of Intelligent Tutoring System: A Review". en. In: *2021 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*. Zallaq, Bahrain: IEEE, 2021-09, pp. 403–410. ISBN: 978-1-66544-032-5. DOI: 10.1109/3ICT53449.2021.9582029. URL: https://ieeexplore.ieee.org/document/9582029/ (visited on 2023-07-07) (cit. on pp. 7, 9, 10, 12, 13).

[15] W. Ma and et al. "Intelligent Tutoring Systems and Learning Outcomes: A Meta-Analysis". In: *J. Educ. Psychol.* – (2014), pp. – (cit. on pp. 8, 9).

[16] H. Zhou. "A study of technical support for artificial intelligence systems applied to knowledge management systems". en. In: *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*. Shenyang, China: IEEE, 2022-01, pp. 921–924. ISBN: 978-1-66544-276-3. DOI: 10.1109/ICPECA53709.2022.9718930. URL: https://ieeexplore.ieee.org/document/9718930/ (visited on 2023-05-03) (cit. on pp. 8, 19, 20).

[17] S. Dallah and H. Odeh. "ICT-Based Knowledge Management in Education:A Systematic Review". en. In: *International Journal of Information Technology and Language Studies (IJITLS)* 6.2 (2022) (cit. on pp. 8, 9, 13, 19).

[18] T. Lu et al. "A Framework of AI-based Intelligent Adaptive Tutoring System". en. In: *2021 16th International Conference on Computer Science & Education (ICCSE)*. Lancaster, United Kingdom: IEEE, 2021-08, pp. 726–731. ISBN: 978-1-66541-468-5. DOI: 10.1109/ICCSE51940.2021.9569273. URL: https://ieeexplore.ieee.org/document/9569273/ (visited on 2023-07-07) (cit. on p. 9).

[19] F. Afsahhosseini. "Technology in Tourism". In: *Proceedings of 8th ITSA Biennial Conference*. Islamic Development Bank (IsDB) Post-doctoral Scholar in ICT4D, RS & GIS Research Center, Sultan Qaboos University. Islamic Development Bank (IsDB). Muscat, Oman, 2020, p. 368 (cit. on pp. 9, 17).

[20] M. F. De Menezes, J. F. D. M. Netto, and A. M. M. Lopes. "Educational Data Mining: Analysis Based On an Intelligent Tutoring System for Teaching Algebra". en. In: *2022 IEEE Frontiers in Education Conference (FIE)*. Uppsala, Sweden: IEEE, 2022-10, pp. 1–7. ISBN: 978-1-66546-244-0. DOI: 10.1109/FIE56618.2022.9962484. URL: https://ieeexplore.ieee.org/document/9962484/ (visited on 2023-07-08) (cit. on pp. 9, 10, 22–24).

[21] R. Manhica, A. Santos, and J. Cravino. "The use of artificial intelligence in learning management systems in the context of higher education : Systematic literature review". en. In: *2022 17th Iberian Conference on Information Systems and Technologies (CISTI)*. Madrid, Spain: IEEE, 2022-06, pp. 1–6. ISBN: 978-989-33-3436-2. DOI: 10.23919/CISTI54924.2022.9820205. URL: https://ieeexplore.ieee.org/document/9820205/ (visited on 2023-07-12) (cit. on pp. 9, 10, 23).

[22] Y. B. Ampadu. "Handling Big Data in Education: A Review of Educational Data Mining Techniques for Specific Educational Problems". en. In: *AI, Computer Science and Robotics Technology* 2 (2023-04). ISSN: 2754-6292. DOI: 10.5772/acrt.17. URL: https://www.intechopen.com/journals/1/articles/185 (visited on 2023-07-13) (cit. on pp. 9, 10, 22–24).

[23] M. Blumenstein. "Synergies of Learning Analytics and Learning Design: A Systematic Review of Student Outcomes". en. In: *Journal of Learning Analytics* 7.3 (2020-12), pp. 13–32. ISSN: 19297750. DOI: 10.18608/jla.2020.73.3. URL: https://www.learning-analytics.info/index.php/JLA/article/view/7004 (visited on 2023-07-12) (cit. on p. 9).

[24] J. Calderon-Valenzuela, K. Payihuanca-Mamani, and N. Bedregal-Alpaca. "Educational Data Mining to Identify the Patterns of Use made by the University Professors of the Moodle Platform". en. In: *International Journal of Advanced Computer Science and Applications* 13.1 (2022). ISSN: 21565570, 2158107X. DOI: 10.14569/IJACSA.2022.0130140. URL: http://thesai.org/Publications/ViewPaper?Volume=13

&Issue=1&Code=IJACSA&SerialNo=40 (visited on 2024-03-26) (cit. on pp. 10, 20, 21).

[25] R. Bodily et al. "The design, development, and implementation of student-facing learning analytics dashboards". en. In: *Journal of Computing in Higher Education* 30.3 (2018-12), pp. 572–598. ISSN: 1042-1726, 1867-1233. DOI: 10.1007/s12528-018-9186-0. URL: http://link.springer.com/10.1007/s12528-018-9186-0 (visited on 2023-07-12) (cit. on p. 10).

[26] L. P. Macfadyen, L. Lockyer, and B. Rienties. "Learning Design and Learning Analytics: Snapshot 2020". en. In: *Journal of Learning Analytics* 7.3 (2020-12), pp. 6–12. ISSN: 19297750. DOI: 10.18608/jla.2020.73.2. URL: https://www.learning-analytics.info/index.php/JLA/article/view/7389 (visited on 2023-07-12) (cit. on p. 10).

[27] M. Yağcı. "Educational data mining: prediction of students' academic performance using machine learning algorithms". en. In: *Smart Learning Environments* 9.1 (2022-12), p. 11. ISSN: 2196-7091. DOI: 10.1186/s40561-022-00192-z. URL: https://slejournal.springeropen.com/articles/10.1186/s40561-022-00192-z (visited on 2023-07-13) (cit. on pp. 10, 11).

[28] F. Saleem et al. "Intelligent Decision Support System for Predicting Student's E-Learning Performance Using Ensemble Machine Learning". en. In: *Mathematics* 9.17 (2021-08), p. 2078. ISSN: 2227-7390. DOI: 10.3390/math9172078. URL: https://www.mdpi.com/2227-7390/9/17/2078 (visited on 2023-07-04) (cit. on pp. 10, 11, 13).

[29] I. Koyuncu and S. Gelbal. "Comparison of Data Mining Classification Algorithms on Educational Data under Different Conditions". en. In: *Egitimde ve Psikolojide Olçme ve Degerlendirme Dergisi* 11.4 (2020-12), pp. 325–345. ISSN: 1309-6575. DOI: 10.21031/epod.696664. URL: http://dergipark.org.tr/en/doi/10.21031/epod.696664 (visited on 2024-04-13) (cit. on pp. 10, 11).

[30] N. Pal and O. Dahiya. "Role of Learning Management System for Evaluating Students' progress in Learning Environment". en. In: *2022 5th International Conference on Contemporary Computing and Informatics (IC3I)*. Uttar Pradesh, India: IEEE, 2022-12, pp. 1800–1806. ISBN: 9798350398267. DOI: 10.1109/IC3I56241.2022.10072794. URL: https://ieeexplore.ieee.org/document/10072794/ (visited on 2024-04-13) (cit. on pp. 13, 21, 24).

[31] A. Jovic, K. Brkic, and N. Bogunovic. "An overview of free software tools for general data mining". en. In: *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija, Croatia: IEEE, 2014-05, pp. 1112–1117. ISBN: 978-953-233-077-9 978-953-233-081-6. DOI: 10.1109/MIPRO.2014.6859735. URL: http://ieeexplore.ieee.org/document/6859735/ (visited on 2024-05-16) (cit. on pp. 14–17).

[32] M. Hall et al. "The WEKA data mining software: an update". en. In: *ACM SIGKDD Explorations Newsletter* 11.1 (2009-11), pp. 10–18. ISSN: 1931-0145, 1931-0153. DOI: 10.1145/1656274.1656278. URL: https://dl.acm.org/doi/10.1145/1656274.1656278 (visited on 2024-05-13) (cit. on pp. 14–16).

[33] P. Kotak and H. Modi. "Enhancing the Data Mining Tool WEKA". en. In: *2020 5th International Conference on Computing, Communication and Security (ICCCS)*. Patna, India: IEEE, 2020-10, pp. 1–6. ISBN: 978-1-72819-180-5. DOI: 10.1109/ICCCS49678.2020.9276870. URL: https://ieeexplore.ieee.org/document/9276870/ (visited on 2024-05-13) (cit. on pp. 14, 16).

[34] I. Stancin and A. Jovic. "An overview and comparison of free Python libraries for data mining and big data analysis". en. In: *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija, Croatia: IEEE, 2019-05, pp. 977–982. ISBN: 978-953-233-098-4. DOI: 10.23919/MIPRO.2019.8757088. URL: https://ieeexplore.ieee.org/document/8757088/ (visited on 2024-05-13) (cit. on pp. 14–16).

[35] F. A. Ibrahim and O. A. Shiba. "Data Mining: WEKA Software (an Overview)". en. In: *Data Mining* 3 (2019) (cit. on p. 15).

[36] G. Research. *Google Colaboratory: A Platform for Deep Learning and Data Science*. https://colab.research.google.com/. Google Colaboratory provides access to powerful computational resources, including GPUs like the `T4` and `L4`, optimized for machine learning workloads. 2024 (cit. on pp. 17, 64, 169).

[37] T. pandas development team. *pandas: powerful Python data analysis toolkit*. https://pandas.pydata.org/. Accessed: 2024-05-16. 2024 (cit. on p. 17).

[38] T. N. development team. *NumPy*. https://numpy.org/. Accessed: 2024-05-16. 2024 (cit. on p. 17).

[39] T. M. development team. *Matplotlib*. https://matplotlib.org/. Accessed: 2024-05-16. 2024 (cit. on p. 17).

[40] T. S. development team. *Seaborn: statistical data visualization*. https://seaborn.pydata.org/. Accessed: 2024-05-16. 2024 (cit. on p. 17).

[41] T. S. development team. *SciPy*. https://scipy.org/. Accessed: 2024-05-16. 2024 (cit. on p. 17).

[42] T. scikit-learn development team. *scikit-learn: machine learning in Python*. https://scikit-learn.org/. Accessed: 2024-05-16. 2024 (cit. on p. 17).

[43] Yulistia, Ermatita, and R. F. Malik. "Knowledge Transfer Model for Private Higher Education Knowledge Management System". en. In: *2019 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS)*. Jakarta, Indonesia: IEEE, 2019-10, pp. 191–194. ISBN: 978-1-72812-930-3. DOI: 10.1109/ICIMCIS481

81.2019.8985229. URL: https://ieeexplore.ieee.org/document/8985229/ (visited on 2023-05-04) (cit. on pp. 17, 19, 20).

[44] R. B. C. Alves and P. Pinheiro. "Factors Influencing Tacit Knowledge Sharing in Research Groups in Higher Education Institutions". en. In: *Administrative Sciences* 12.3 (2022-07), p. 89. ISSN: 2076-3387. DOI: 10.3390/admsci12030089. URL: https://www.mdpi.com/2076-3387/12/3/89 (visited on 2023-06-15) (cit. on p. 17).

[45] T. H. Davenport, L. Prusak, and L. Prusak. *Working Knowledge: How Organizations Manage What They Know*. USA: Harvard Business School Press, 1997. ISBN: 0875846556 (cit. on pp. 17, 18).

[46] I. Nonaka. "A Dynamic Theory of Organizational Knowledge Creation". en. In: *Organization Science* 5.1 (1994), pp. 14–37. URL: http://www.jstor.org/stable/2635068 (cit. on pp. 17–19).

[47] M. Pinto. "Knowledge management in higher education institutions: A framework to improve collaboration". en. In: *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)*. Barcelona, Spain: IEEE, 2014-06, pp. 1–4. ISBN: 978-989-98434-3-1. DOI: 10.1109/CISTI.2014.6876876. URL: https://ieeexplore.ieee.org/document/6876876 (visited on 2023-06-17) (cit. on pp. 17, 19).

[48] N. Istyanto and M. Nasrullah. "VIEWOF~1.PDF". en. In: *Journal of Information Systems and Informatics* 4.3 (2022), pp. 753–771. ISSN: 2656-4882 (cit. on pp. 18, 19).

[49] M. K. Anam et al. "Development of Knowledge Management System to Improve the Performance of the New Student Admission System for Higher Education". en. In: *JISA(Jurnal Informatika dan Sains)* 5.2 (2022-12), pp. 181–186. ISSN: 2614-8404, 2776-3234. DOI: 10.31326/jisa.v5i2.1443. URL: https://trilogi.ac.id/journal/ks/index.php/JISA/article/view/1443 (visited on 2023-05-03) (cit. on pp. 19, 20).

[50] I. Becerra-Fernandez and D. Leidner, eds. *Knowledge Management: An Evolutionary View*. Vol. 12. Advances in Management Information Systems. Library of Congress Cataloging-in-Publication Data. Armonk, NY: M.E. Sharpe, 2008. ISBN: 978-0-7656-1637-1 (cit. on pp. 19, 20).

[51] D. L. Reid. "Learning Management Systems: The Game Changer for Traditional Teaching and Learning at Adult and Higher Education Institutions". en. In: *Global Journal of Human-Social Science* (2019-07), pp. 1–14. ISSN: 2249-460X, 0975-587X. DOI: 10.34257/GJHSSGVOL19IS6PG1. URL: https://socialscienceresearch.org/index.php/GJHSS/article/view/2907 (visited on 2023-06-17) (cit. on pp. 20, 21).

[52] H. Coates, R. James, and G. Baldwin. "A CRITICAL EXAMINATION OF THE EFFECTS OF LEARNING MANAGEMENT SYSTEMS ON UNIVERSITY TEACHING AND LEARNING". en. In: *Tertiary Education and Management* 11 (2005) (cit. on p. 21).

[53] A. Pardo and G. Siemens. "Ethical and privacy principles for learning analytics: Ethical and privacy principles". en. In: *British Journal of Educational Technology* 45.3 (2014-05), pp. 438–450. ISSN: 00071013. DOI: 10.1111/bjet.12152. URL: https://onlinelibrary.wiley.com/doi/10.1111/bjet.12152 (visited on 2023-07-01) (cit. on p. 21).

[54] M. Dougiamas. *Pedagogy*. English. 2006-11. URL: https://docs.moodle.org/404/en/Pedagogy (visited on 2024-05-11) (cit. on p. 22).

[55] D. Distante et al. "MILA: A SCORM-Compliant Interactive Learning Analytics Tool for Moodle". en. In: *2020 IEEE 20th International Conference on Advanced Learning Technologies (ICALT)*. Tartu, Estonia: IEEE, 2020-07, pp. 169–171. ISBN: 978-1-72816-090-0. DOI: 10.1109/ICALT49669.2020.00056. URL: https://ieeexplore.ieee.org/document/9155907/ (visited on 2024-03-26) (cit. on p. 22).

[56] B. Centenaro et al. "Systematic Mapping of Moodle Dashboards Focused on Learning Analytics Tasks". en. In: *2021 XVI Latin American Conference on Learning Technologies (LACLO)*. Arequipa, Peru: IEEE, 2021-10, pp. 60–66. ISBN: 978-1-66542-358-8. DOI: 10.1109/LACLO54177.2021.00013. URL: https://ieeexplore.ieee.org/document/9725214/ (visited on 2024-05-11) (cit. on p. 22).

[57] A. Al-Ajlan and H. Zedan. "Why Moodle". en. In: *2008 12th IEEE International Workshop on Future Trends of Distributed Computing Systems*. Kunming, China: IEEE, 2008, pp. 58–64. ISBN: 978-0-7695-3377-3. DOI: 10.1109/FTDCS.2008.22. URL: http://ieeexplore.ieee.org/document/4683115/ (visited on 2024-05-11) (cit. on p. 22).

[58] A. C. Ikegwu, H. F. Nweke, and C. V. Anikwe. "Recent trends in computational intelligence for educational big data analysis". en. In: *Iran Journal of Computer Science* 7.1 (2024-03), pp. 103–129. ISSN: 2520-8438, 2520-8446. DOI: 10.1007/s42044-023-00158-5. URL: https://link.springer.com/10.1007/s42044-023-00158-5 (visited on 2024-04-13) (cit. on p. 23).

[59] N. Sghir, A. Adadi, and M. Lahmer. "Recent advances in Predictive Learning Analytics: A decade systematic review (2012–2022)". en. In: *Education and Information Technologies* 28.7 (2023-07), pp. 8299–8333. ISSN: 1360-2357, 1573-7608. DOI: 10.1007/s10639-022-11536-0. URL: https://link.springer.com/10.1007/s10639-022-11536-0 (visited on 2024-04-13) (cit. on p. 23).

[60] D. Hooshyar et al. "Clustering Algorithms in an Educational Context: An Automatic Comparative Approach". en. In: *IEEE Access* 8 (2020), pp. 146994–147014. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3014948. URL: https://ieeexplore.ieee.org/document/9162041/ (visited on 2024-04-13) (cit. on p. 24).

[61] P. Chapman et al. "The CRISP-DM Process Model". en. In: *Discussion Paper* (1999-03) (cit. on pp. 24–26).

[62] N. F. Oliveira. "ETL for Data Science? A Case Study." en. PhD thesis. Lisbon, Portugal: ISCTE-IUL - Lisbon University Institute, 2021-11 (cit. on p. 24).

[63] M. Kaluža, M. Kalanj, and B. Vukelić. "A comparison of back-end frameworks for web application development". en. In: *Zbornik Veleučilišta u Rijeci* 7.1 (2019), pp. 317–332. ISSN: 18491723, 18481299. DOI: 10.31784/zvr.7.1.10. URL: https://hrcak.srce.hr/index.php?show=clanak&id_clanak_jezik=321176 (visited on 2023-06-18) (cit. on p. 175).

[64] N. Bakshi et al. "Spring Framework vs Django Framework: A Comparative Study". In: *International Research Journal of Engineering and Technology (IRJET)* 07 (06 2020), pp. 6232–6236. URL: www.irjet.net (cit. on p. 175).

[65] J. Plekhanova. "Evaluating web development frameworks:" en. In: (2009-09) (cit. on p. 175).

A

Data Details, Preparation and Initial Analysis

A.1 Detailed Data Description

This section offers an exhaustive overview of the datasets utilized for the analysis of the CTCT curricular unit spanning academic years from 2015/16 to 2023/24. These datasets are rich in attributes that reflect student performance, participation, and overall academic progress.

Categorical Attributes

The datasets encompass a variety of categorical attributes which include:

- **`Sex`**: This attribute distinguishes between Female (0) and Male (1).
- **`freq`**: This attendance marker categorizes students as S (Suficiente - Enough), A (Ausente - Absent), and I (Insuficiente - Not enough). An additional category, represented by "*", appears in three instances in the 2024 dataset but is excluded from analysis due to its undefined nature.
- **`SS`**: Special status indicator for students, its usage and significance have varied across different academic years and is not considered in this study.
- **`Commuting`**: A qualitative measure assessing the distance students travel from their residence to the university. The evolution of this attribute is as follows:
  - **2015/16**: Initially labeled as "Deslocado" (Commuting) or not, specifics not fully delineated.
  - **2016/17 to 2018/19**: Differentiated into several categories, ranging from "Não Deslocado" (Non-commuting), "Deslocado - Perto" (Nearby Commuting), to "Deslocado - Ilhas" (Islands Commuting).
  - **2021/22**: The year was marked by a transition to online delivery, leading to the absence of commuting data.

  - **2022/23 to 2023/24**: In these years, the attribute was refined to include specific locations such as Almada, nearby areas like Seixal, Lisbon districts, and further subdivisions including bordering municipalities and districts, extending to the islands.
  - **2023/24**: Discontinued due to organizational changes at FCT.

- **`Course`**: Represents the university degree program in which the student is enrolled.

Given by CTCT coordination, two batches of datasets with the properties shown in Table A.1 were received. The CTCT data from the academic years 2015/16 to 2018/19 were initially used, followed by the years 2021/22 to 2023/24. The first batch was processed using a different notebook each, but the introduction of new datasets from 2021/22 to 2023/24 implied the creation of more notebooks to process the data. The first batch of datasets was also updated with attributes relating to the final grade and class attendance when the second batch was received. This highlighted the need for some adjustments to the existing scripts. The iterative updates became cumbersome, as even minor changes affected all seven notebooks previously developed. To streamline this process and enhance efficiency, the scripts were reorganized into two primary notebooks—one for data preparation and another for data understanding.

Table A.1: Attributes summary for the available datasets.

Year	Filename	Attributes
2016	CTCT201516_Notas.xlsx	`Numero`, `Turma`, `Sigla_curso`, `Sexo`, `EE`, `PTP4`, `PTP5`, `freq2016`, `NotaFinalDecPauta`, `NotaFinalClip`, ... (total 134 attributes)
2017	CTCT201617_Notas.xlsx	`Numero`, `Turma`, `Sexo`, `EDESLOCADO`, `EE`, `PTP4`, `PTP5`, `Freq201617`, `Val_Tot_Pauta`, `NotaFinal`, ... (total 134 attributes)
2018	CTCT201718_Notas.xlsx	`Numero`, `Turma`, `Sigla_curso`, `Sexo`, `DESLOCADO`, `PTP4`, `PTP5`, `Freq201718`, `Val_Tot_Pauta`, `NotaFinal`, ... (total 140 attributes)
2019	CTCT201819_Notas.xlsx	`Numero`, `Turma`, `Sigla_curso`, `Sexo`, `EE`, `PTP3`, `PTP4`, `PTP5`, `Val_Tot_Pauta`, `NotaFinal`, ... (total 141 attributes)
2022	CTCT202122_Notas.xlsx	`Numero`, `Turma`, `Sigla_curso`, `Sexo`, `EE`, `PrtTP2`, `PrtTP3`, `Freq2122`, `Valor3`, `NotaFinal`, ... (total 91 attributes)

Continued on next page

Table A.1: (continued)

Year	Filename	Attributes
2023	CTCT202223_Notas.xlsx	`Numero`, `Turma`, `Sigla_curso`, `Sexo`, `EE`, `PrtTP4`, `PrtTP5`, `Freq202223`, `VFinalPauta`, `NFinal_Pauta`, ... (total 102 attributes)
2024	CTCT202324_Notas.xlsx	`Numero`, `Turma`, `Sigla_curso`, `Sexo`, `EE`, `PrtTP3`, `PrtTP4`, `Freq202324`, `VFinalPauta`, `NFinal_Pauta`, ... (total 105 attributes)

Table A.2 presents a detailed list of university degree initials as captured under the `Course` attribute across various datasets. Each entry in the table specifies the initials, the full degree title in Portuguese, its English translation, and the academic years for which the data is available. This table serves as a quick reference to identify the academic offerings over the years and their representation in the datasets.

Table A.2: University degrees at NOVA to understand `Course` categories.

Initials	Degree (Portuguese and English)	Datasets
MIEI	Mestrado Integrado (MI) em Engenharia Informática Integrated Master (IM) in Computer Engineering	2016, 7, 8, 9
MIEEC	MI em Engenharia Eletrotécnica e de Computadores IM in Electrical and Computer Engineering	2016, 7, 8, 9
LBq	Licenciatura (L.) em Bioquímica Bachelor (B.) in Biochemistry	2016-9, 22-24
MIEMc	MI em Engenharia Mecânica IM in Mechanical Engineering	2016-9
MIEQB	MI em Engenharia Química e Bioquímica IM in Chemical and Biochemical Engineering	2016-9
LBCM	L. em Biologia Celular e Molecular B. in Cellular and Molecular Biology	2016-9, 22-24
MIEC	MI em Engenharia Civil IM in Civil Engineering	2016-9
MIEA	MI em Engenharia do Ambiente IM in Environmental Engineering	2016-9

Continued on next page

Table A.2: (continued)

Initials	Degree (Portuguese and English)	Datasets
MIEB	MI em Engenharia Biomédica IM in Biomedical Engineering	2016-9
MIEGI	MI em Engenharia e Gestão Industrial IM in Industrial Engineering and Management	2016-9
MIEMN	MI em Engenharia de Micro e Nanotecnologias IM in Micro and Nanotechnologies Engineering	2016-9
LM	L. em Matemática B. in Mathematics	2016-9, 22-24
MIEF	MI em Engenharia Física IM in Physics Engineering	2016-9
LQA	L. em Química Aplicada B. in Applied Chemistry	2016-9, 22-24
MIEMat	MI em Engenharia de Materiais IM in Materials Engineering	2016-9
LEG	L. em Engenharia Geológica B. in Geological Engineering	2016-9, 22-24
LMAGR	L. em Matemática Aplicada à Gestão do Risco B. in Mathematics Applied to Risk Management	2019, 22-24
LEI	L. em Engenharia Informática B. in Computer Engineering	2022-24
LEEC	L. em Engenharia Eletrotécnica e de Computadores B. in Electrical and Computer Engineering	2022-24
LEQB	L. em Engenharia Química e Bioquímica B. in Chemical and Biochemical Engineering	2022-24
LEMc	L. em Engenharia Mecânica B. in Mechanical Engineering	2022-24
LEC	L. em Engenharia Civil B. in Civil Engineering	2022-24
LEA	L. em Engenharia do Ambiente B. in Environmental Engineering	2022-24

Continued on next page

Table A.2: (continued)

Initials	Degree (Portuguese and English)	Datasets
LEB	L. em Engenharia Biomédica B. in Biomedical Engineering	2022-24
LEGI	L. em Engenharia e Gestão Industrial B. in Industrial Engineering and Management	2022-24
LEMN	L. em Engenharia de Micro e Nanotecnologias B. in Micro and Nanotechnologies Engineering	2022-24
LEF	L. em Engenharia Física B. in Physics Engineering	2022-24
LEMt	L. em Engenharia de Materiais B. in Materials Engineering	2022-24
LTAI	L. Agro-Industrial B. in Agro-Industrial Technology	2024

Because the integrated master changed to bachelor in 2022, in the processing phase it was converted the integrated master names to the bachelor ones, so the courses can be compared across different years. CTCT course it's a first year discipline both in the bachelor and the integrated master.

Datasets Parameters

Table A.3 outlines the key parameters for the datasets corresponding to different academic years, and that are included in the dictionary with dataset characteristics, `dataset_params`, included in the begining of `data_preparation` notebook. The table includes details such as the filename of the raw `..xlsx` file, the range of the last column considered (`Last_col_range`), the column representing attendance frequency (`freq`), and the column representing the final grade (`FinalGrade`. Additional parameters such as rows to skip, columns to skip, and the value representing a failed grade are consistent across multiple datasets and thus are described separately.

Activity Dictionary

This activity dictionary it's one of the fields of the `dataset_params` dictionary with information necessary to process all the loaded datasets and include the following fields:

- **Activity ID:** A integer that represents the unique identifier for each activity based on a excel file created using the characterization of activities information given by CTCT coordination.

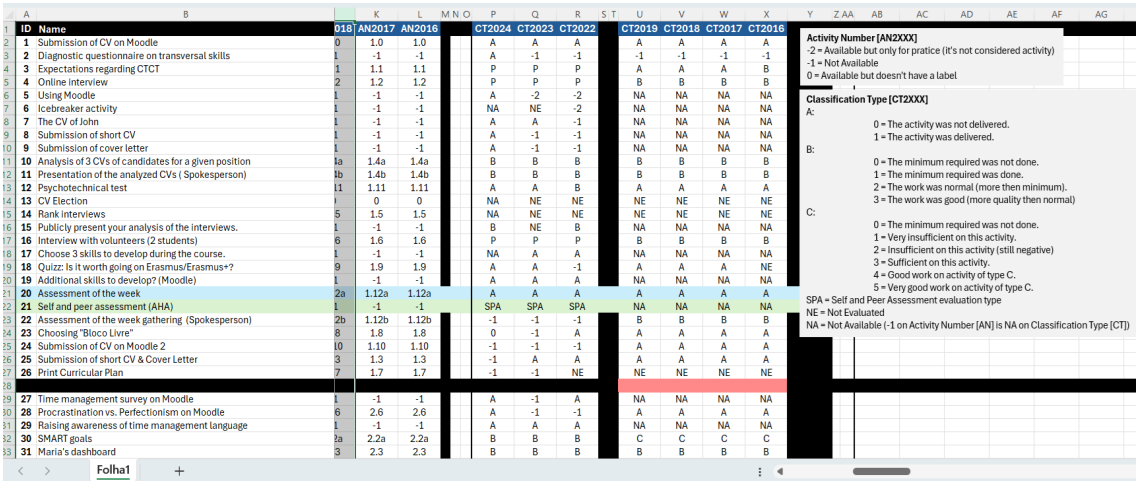
Table A.3: Dataset parameters for the available datasets.

Year	Filename	Last_col_range	Freq_col	FinalGrade
2016	CTCT201516_Notas.xlsx	135	freq2016	NotaFinalDecPauta
2017	CTCT201617_Notas.xlsx	136	Freq201617	Val_Tot_Pauta
2018	CTCT201718_Notas.xlsx	142	Freq201718	Val_Tot_Pauta
2019	CTCT201819_Notas.xlsx	143	missing	Val_Tot_Pauta
2022	CTCT202122_Notas.xlsx	93	Freq2122	Valor3
2023	CTCT202223_Notas.xlsx	104	Freq202223	VFinalPauta
2024	CTCT202324_Notas.xlsx	107	Freq202324	VFinalPauta

- **Activity Name:** A string with the activity name.
- **Type of Evaluation:** A string that can be **NE** for not evaluated, **A** for type A activity (binary classification), **B** for type B (0, 1, 2 or 3 of classification), **C** for type C activity (classification of 0, 1, 2, 3, 4 or 5), or **P** for participation type of activity.
- **Individual or Group:** Indicates whether the activity is individual (indicated as a flag 1) or group-based (indicated as 0).
- **Written or Oral:** Indicates whether the activity is written (if 1 on this field) or oral (indicated as 0).
- **Mandatory or Optional:** Indicates whether the activity is for all or for only some limited number of volunteers (indicated as 1 if it's just for some students)
- **Homework:** The last integer of the activity dictionary indicated if a activity is homework (represented with 1 as a flag) or a class activity (represented by zero).
- **Thematic:** The last field of the activity dictionary is a string with the thematic associated with the activity or an empty string for self and peer assessment and week evaluation types of activities. 'Employability-focused planning', 'Time management', 'Teamwork', 'Leadership', 'Advanced Excel use', 'Information Searching', 'Communication in science and technology', 'Ethics and deontology' were the thematics used.

The process of synthesizing the data used to create this dictionary included filling an Microsoft Excel Worksheet file named `Activities Evolution` with details about the 147 different activities, comparing multiple files with activity characterization by hand, and analyzing the received Excel files containing CTCT data to note any differences. This file associates the classification type of the activity for the corresponding edition (columns `CT2024` to `CT2016` of the produced file) with each activity's unique `ID` and activity number (columns `AN2024` to `AN2016`) for each dataset, as shown in Figure A.1. This

`dataset_params` is a necessary tool for all of the analysis and separation tasks because it enables the development of the majority of the lists found in all_lists dictionaries.



ID	Name	018	AN2017	AN2016		CT2024	CT2023	CT2022		CT2019	CT2018	CT2017	CT2016
1	Submission of CV on Moodle	0	1.0	1.0		A	A	A		A	A	A	A
2	Diagnostic questionnaire on transversal skills	1	-1	-1		A	-1	-1		-1	-1	-1	-1
3	Expectations regarding CTCT	1	1.1	1.1		P	P	P		A	A	A	B
4	Online interview	2	1.2	1.2		P	P	P		B	B	B	B
5	Using Moodle	1	-1	-1		A	-2	-2		NA	NA	NA	NA
6	Icebreaker activity	1	-1	-1		NA	NE	-2		NA	NA	NA	NA
7	The CV of John	1	-1	-1		A	A	-1		NA	NA	NA	NA
8	Submission of short CV	1	-1	-1		A	-1	-1		NA	NA	NA	NA
9	Submission of cover letter	1	-1	-1		A	-1	-1		NA	NA	NA	NA
10	Analysis of 3 CVs of candidates for a given position	1a	1.4a	1.4a		B	B	B		B	B	B	B
11	Presentation of the analyzed CVs (Spokesperson)	1b	1.4b	1.4b		B	B	B		B	B	B	B
12	Psychotechnical test	11	1.11	1.11		A	A	B		A	A	A	A
13	CV Election	I	0	0		NA	NE	NE		NE	NE	NE	NE
14	Rank Interviews	5	1.5	1.5		NA	NE	NE		NE	NE	NE	NE
15	Publicly present your analysis of the interviews.	1	-1	-1		B	NE	B		NA	NA	NA	NA
16	Interview with volunteers (2 students)	6	1.6	1.6		P	P	P		B	B	B	B
17	Choose 3 skills to develop during the course.	1	-1	-1		NA	A	A		NA	NA	NA	NA
18	Quizz: Is it worth going on Erasmus/Erasmus+?	9	1.9	1.9		A	A	-1		A	A	A	NE
19	Additional skills to develop? (Moodle)	1	-1	-1		A	A	A		NA	NA	NA	NA
20	Assessment of the week	2a	1.12a	1.12a		A	A	A		A	A	A	A
21	Self and peer assessment (AHA)	1	-1	-1		SPA	SPA	SPA		NA	NA	NA	NA
22	Assessment of the week gathering (Spokesperson)	2b	1.12b	1.12b		-1	-1	-1		B	B	B	B
23	Choosing "Bloco Livre"	8	1.8	1.8		0	-1	A		A	A	A	A
24	Submission of CV on Moodle 2	10	1.10	1.10		-1	-1	-1		A	A	A	A
25	Submission of short CV & Cover Letter	3	1.3	1.3		-1	A	A		A	A	A	A
26	Print Curricular Plan	7	1.7	1.7		-1	-1	NE		NE	NE	NE	NE
27	Time management survey on Moodle	1	-1	-1		A	-1	A		NA	NA	NA	NA
28	Procrastination vs. Perfectionism on Moodle	6	2.6	2.6		A	-1	-1		A	A	A	A
29	Raising awareness of time management language	1	-1	-1		A	A	A		NA	NA	NA	NA
30	SMART goals	2a	2.2a	2.2a		B	B	B		C	C	C	C
31	Maria's dashboard	3	2.3	2.3		B	B	B		B	B	B	B

Activity Number [AN2XXX]
-2 = Available but only for pratice (it's not considered activity)
-1 = Not Available
0 = Available but doesn't have a label

Classification Type [CT2XXX]
A:
0 = The activity was not delivered.
1 = The activity was delivered.
B:
0 = The minimum required was not done.
1 = The minimum required was done.
2 = The work was normal (more then minimum).
3 = The work was good (more quality then normal)
C:
0 = The minimum required was not done.
1 = Very insufficient on this activity.
2 = Insufficient on this activity (still negative)
3 = Sufficient on this activity.
4 = Good work on activity of type C.
5 = Very good work on activity of type C.
SPA = Self and Peer Assessment evaluation type
NE = Not Evaluated
NA = Not Available (-1 on Activity Number [AN] is NA on Classification Type [CT])

Figure A.1: Developed Microsoft Excel Worksheet `Activities Evolution`, that relates each activity with an ID, a name, a description, the activity column name in each dataset and the type of evaluation.

A.2 Data Preparation Notebook

The data preparation Jupyter notebook to preprocesses the data (`data_preparation`) handles missing values, corrects categorical data, and creates various subsets of the data for different types of analysis, and it include several steps as was shown in Table 3.1.

The notebook initiates by loading several fundamental libraries to support data manipulation, numerical computing, data visualization, and file operations. Pandas is employed for data manipulation and analysis via powerful DataFrame structures. NumPy supports numerical computing with extensive capabilities for handling large, multi-dimensional arrays and matrices. For data visualization, Matplotlib provides tools for creating static, animated, and interactive visualizations, while Seaborn extends these capabilities to include statistical data visualization. Pickle is used for the serialization and deserialization of Python object structures, enabling the preservation and restoration of model states. File operations are facilitated by OS and Shutil, which provide comprehensive file management functionalities essential for data handling and model storage. Integration with Google Colaboratory is streamlined through Google Colab, which allows for efficient file uploads and downloads.

The parameters for each dataset are defined, specifying the filename, rows to skip, columns to skip, last column range, commuting information, failed grade value (the student's that fail in each of CTCT were represented with different characters on the `FinalGrade` and not a continuous number as the students that failed), frequency column (that corresponds to the attendance of a student), grade column, and activity dictionary, as described in Subsection A.2. These parameters were essential to make the `all_lists`

and `all_lists_id` dictionaries, that have all the necessary lists to process the produced dataframes in several ways.

Table A.4 shows the methods coded in `data_preparation.ipynb` file, including the `process_activities` and `process_activities2` that create specific pivot tables that represent many aspects of the evolution of CTCT activities evolution along the studied editions. The resultant pivot_table from `process_activities` method was visualized as the heatmap shown in Figure 4.1, for more details about the types of activities in the several editions of this curricular unit.

Table A.4: Methods used in the data preparation notebook.

Method	Description
process_activities	Aggregates activity details from dataset parameters into a DataFrame and generates a pivot table for analysis.
process_activities2	Extends `process_activities` by providing a detailed summary DataFrame grouped by Evaluation Type and Year.
load_and_process_data	Handles initial data loading, renaming columns, categorizing data types, and dropping irrelevant columns.
generate_basic_lists	Generates lists categorizing the dataset features for structured processing and accessibility.
generate_weekly_and_participation_lists	Generates organized lists of activities that correspond to each of CTCT week.
generate_activity_type_lists	Categorizes activities to streamline process management and analytics based on activity type, using the information from the dataset_params dictionary.
process_combined_datasets	Processes the combined datasets, generating and printing all relevant lists dynamically for each year.
create_weekly_and_total_datasets	Generates weekly totals and combines them with the original dataset.

Continued on next page

Table A.4: (continued)

Method	Description
print_totals_description	Provides a descriptive summary of total metrics (the created columns from datasets.
check_missing_values_per_column	Identifies and reports missing data points per column. As a similar code to Code Snippet B.5.
fill_missing_values_appropriately	Applies appropriate methods to impute missing values, but it was not applied because of dataset specifics.
drop_column_ss	Creates a separate dataset without special status students column.
verify_no_ss_column	Ensures `SS` column are retained post-processing.
handle_outliers	Implements outlier detection but doesn't apply any change because all the outliers are zero, which is expected in students that fail the discipline.
detect_changes	Compares datasets pre- and post-corrections to validate data manipulations and ensure expected transformations.
verify_and_correct_activities	Confirms and adjusts activity characterizations to align with dataset specifications and integrity.
update_column_names	Standardizes column names across multiple datasets to ensure uniformity and avoid data misinterpretation, using the `ID` used for each activity in `dataset_params` dictionary. The code is shown in Code Snippet B.3.
update_all_lists	Updates all reference lists within the dataset scripts, reflecting changes in the data structure or content. The code is shown in Code Snippet B.3.

Continued on next page

Table A.4: (continued)

Method	Description
remove_duplicate_columns	Eliminates redundant columns from the dataset to prevent data redundancy and enhance processing efficiency.
split_datasets_by_missing_values	Segregates datasets based on the quantity of missing values, allowing targeted data processing strategies.
combine_course_names	Consolidates diverse course naming conventions into a unified format, enhancing data consistency.
standardize_sex_column	Normalizes data entries in the 'Sex' column to be represented by 'Female' and 'Male' instead of '0' and '1'.
process_datasets_for_passed_and_failed	Processes datasets to create approved and repproved datasets, and combines them into a complete dataset with a 'Passed' column.
save_processed_data	Archives the processed datasets in specified formats to ensure the persistence and reusability of data transformations.
download_the_zip_files	Facilitates the download of all processed data, encapsulated in zip files for convenient distribution and access.

Loading and Processing Data

Besides that study of the characterization of activities, the datasets with the parameters refered in Table A.3 were loaded using the method shown in Code Snippet A.1.

Code Snippet A.1: Method to load and process data.

```
def load_and_process_data(year, params):
    """Load and process the dataset according to specified parameters."""
    filename = params['filename']
    skiprows = params['skiprows']
    cols2skip = params['cols2skip']
    last_col_range = params['last_col_range']

    # Define columns to use based on columns to skip
```

```
    cols = [i for i in range(last_col_range) if i not in cols2kip]

    # Load dataset
    dataset = pd.read_excel(filename, sheet_name=0, usecols=cols, skiprows=skiprows)
    print("DATASET for {}".format(year))
    print(dataset.dtypes)
     # Ensure all column data are strings before replacing
    dataset = dataset.applymap(lambda x: str(x) if not pd.isnull(x) else x)

    # Rename columns to a standardized format
    dataset.columns = dataset.columns.str.replace('Numero', 'ID').str.replace('Turma', '
        ↪ Class').str.replace('Sigla_curso', 'Course').str.replace('Sexo', 'Sex').str.
        ↪ replace(params['commuting'], 'Commuting').str.replace('EE', 'SS').str.replace('
        ↪ AHA', 'SPA').str.replace('Bump', 'BonCoord').str.replace('Prt','PP').str.
        ↪ replace('PPTP','PTP').str.replace(params['freq_col'], 'freq').str.replace(
        ↪ params['grade_col'], 'FinalGrade').str.replace(params['grade_integer_col'], '
        ↪ FinalGradeInteger')

    # Set ID as index
    dataset = dataset.set_index("ID")
    if (year == '2016'):
      dataset = dataset.drop(["3_2 TPC","3_4 TPC"], axis='columns') # Drop irrelevant
          ↪ columns (only exist in 2016 dataset)

    # Handle non-numeric entries before conversion
    dataset['FinalGrade'] = dataset['FinalGrade'].replace([params['failed_grade_value'], '
        ↪ *'], np.nan)

    # Replace commas with periods and convert to float
    dataset['FinalGrade'] = dataset['FinalGrade'].str.replace(',', '.').astype(float,
        ↪ errors='raise')

    # Change data types for easier analysis
    if 'Class' in dataset.columns:
        dataset['Class'] = dataset['Class'].astype('category')
    if 'Course' in dataset.columns:
        dataset['Course'] = dataset['Course'].astype('category')
        if 'EG' in dataset['Course'].cat.categories:
            dataset['Course'] = dataset['Course'].cat.remove_categories(['EG'])
            dataset["Course"] = dataset["Course"].fillna('LEG')
    if 'Sex' in dataset.columns:
        dataset['Sex'] = dataset['Sex'].astype('category')
    if 'Commuting' in dataset.columns:
        dataset['Commuting'] = dataset['Commuting'].astype('category')
        dataset['Commuting'] = dataset['Commuting'].cat.add_categories(['NA'])
        dataset['Commuting'] = dataset['Commuting'].fillna('NA')
        if year == '2017':
            dataset['Commuting'] = dataset['Commuting'].cat.add_categories(['1', '2', '3', '
                ↪ 4', '5'])
```

```
48            dataset['Commuting'] = dataset['Commuting'].cat.remove_categories(['DESLOCADO -
                  ↪ PERTO'])
49            dataset['Commuting'] = dataset['Commuting'].fillna('1')
50            dataset['Commuting'] = dataset['Commuting'].cat.remove_categories(['DESLOCADO'])
                  ↪
51            dataset['Commuting'] = dataset['Commuting'].fillna('2')
52            dataset['Commuting'] = dataset['Commuting'].cat.remove_categories(['DESLOCADO -
                  ↪ ILHAS'])
53            dataset['Commuting'] = dataset['Commuting'].fillna('3')
54            dataset['Commuting'] = dataset['Commuting'].cat.remove_categories(['????
                  ↪ INSCRITO ÀPOSTERIORI'])
55            dataset['Commuting'] = dataset['Commuting'].fillna('NA')
56        if year == '2018':
57          dataset['Commuting'] = dataset['Commuting'].cat.add_categories(['5'])
58        if year == '2019':
59          dataset['Commuting'] = dataset['Commuting'].cat.add_categories(['5'])
60        if year == '2024':
61          dataset = dataset.drop('Commuting', axis=1)
62    if 'SS' in dataset.columns:
63        dataset['SS'] = dataset['SS'].astype('category')
64    if 'freq' in dataset.columns:
65        dataset['freq'] = dataset['freq'].astype('category')
66
67    # Check if 'FinalGradeInteger' is in columns before converting
68    if 'FinalGradeInteger' in dataset.columns:
69        # Replace non-numeric values in 'FinalGradeInteger' and convert
70        dataset['FinalGradeInteger'] = pd.to_numeric(dataset['FinalGradeInteger'], errors='
              ↪ coerce')
71
72        # Filter datasets for passed and failed students based on the failed grade value
73        dataset_passed = dataset[dataset['FinalGradeInteger'].notna()].copy()
74        dataset_failed = dataset[dataset['FinalGradeInteger'].isna()].copy()
75
76        # Convert data types after filtering
77        dataset_passed['FinalGradeInteger'] = dataset_passed['FinalGradeInteger'].astype('
              ↪ int64')
78    else:
79        print(f"'FinalGradeInteger' column is missing in the dataset for {year}.")
80        dataset_passed = dataset.copy()
81        dataset_failed = pd.DataFrame()
82
83    return dataset_passed, dataset_failed
```

Combining and Processing Datasets

Following loading and initial processing, using the code outlined in Code Snippet A.2, the dataset containing students who passed and failed CTCT is combined to create a unified structure for analysis. It is then cleaned as indicated in Code Snippet A.3 to make sure the final dataset is free of any rows or columns that are empty, optimizing the data

structure for analysis. Because the processing procedures related to the grade columns were different, the processing was carried out separately for the students who passed and failed.

Code Snippet A.2: Combine datasets for different years.

```
def combine_and_process_datasets(datasets):
    combined_datasets = {}
    for year, data_types in datasets.items():
        combined_dataset = pd.concat([data_types['passed'], data_types['failed']])
        combined_datasets[year] = combined_dataset
    return combined_datasets
```

Code Snippet A.3: Clean datasets by removing empty rows and columns.

```
def clean_datasets(datasets):
    cleaned_datasets = {}
    for year, dataset in datasets.items():
        dataset_cleaned = dataset.dropna(axis=1, how='all')
        dataset_cleaned = dataset_cleaned.dropna(axis=0, how='all')
        cleaned_datasets[year] = dataset_cleaned
    return cleaned_datasets
```

Generating Lists

In order to facilitate subsequent analysis steps, various lists were generated to categorize and separate different types of features from the dataset. These lists were part of a Python dictionary, named `all_lists`, designed to assist in the processing of the dataset. The consolidated table below (Table A.5) describes the structure of the lists, offering a comprehensive overview of column names and their respective descriptions for different dataset features, participation points by week, and activity types.

Table A.5: Comprehensive structure of lists generated for the dataset, stored in the `all_lists` Python dictionary.

List Name	Description
`listAllColumns`	Includes all names of all columns present in the dataset.
`listCategorical`	Includes the names of the columns with categorical data such as 'Course', 'Class', and 'Sex'.
`listGrades`	List of grade-related column names like 'FinalGrade' and 'FinalGradeInteger'.

Continued on next page

Table A.5: (continued)

List Name	Description
`listAsyncColumns`	List of activity names regarding activities done in asynchronous classes for students who couldn't attend presencial classes (denoted by suffix '_TE').
`listBonColumns`	Columns starting with "B", generally indicating bonus points or special grading criteria.
`listSPAColumns`	Columns containing "SPA", indicating self and peer assessment evaluation points decided among student groups.
`listPPointsColumns`	Columns starting with "P", usually relating to participation points or other performance metrics.
`listSyncColumns`	Synchronous columns correspond to the activities performed during normal classes (corresponding to activity points).
`listWeek1, 2, 3, 4, and 5`	Each list contains activities for each week in a specific CTCT edition, corresponding to the column names in the dataset (including synchronous and asynchronous activities).
`listWeekSync`	Dictionary of lists of column names (activity names) for each week, similar to `listWeek1` to `listWeek5`, but excluding asynchronous activities.
`listPPWeek1, 2, 3, 4, and 5`	Lists with column names detailing participation points for practical classes, indicating the names of the columns for each class in each edition of the discipline.
`listPP_TPs`	Lists of column names for participation points specifically tied to theoretical classes.
`EvTypeA, B, C, SPA, NE`	Lists categorizing activities by specific types of evaluation: type A (binary grade); type B (0, 1, 2, or 3 values per activity); type C (integer range from 0 to 5 activity points); type SPA (self and peer assessment); type NE (not evaluated).
`WrittenA, OralA`	Distinctions between written and oral activities.

Continued on next page

Table A.5: (continued)

List Name	Description
`JFSA, WithoutJFS`	Indicates whether activities include "Just-For-Some" students (volunteers) or if the activity is for all.
`GroupA, IndivA`	Separates group activities from individual ones.
`ClassA, HomeworkA`	Differentiates between in-class activities and homework assignments.

Data Cleaning and Feature Creation

In this step, categorical data correction and handling of missing values are done. Categorical data is transformed into a format adequate for analysis, and missing values are either filled in with appropriate values or eliminated. With the exception of `FinalGrade`, which was changed to `float` (as displayed in line 8 of Code Snippet **??**), all numerical attributes in the dataset have been adjusted to the type `int64`. Several issues that were initially discovered with some categorical attributes have also been fixed using this code. In the same method, the columns are renamed to English in addition to loading the original Excel files with data to be processed.

Furthermore, students who attended all of their classes in asynchronous mode are excluded because the majority of datasets do not include students who attended asynchonous classes, and even those whose information is available are not representative of the CTCT number of students.

New columns are created to aggregate different types of points students earned throughout the CTCT discipline, namely the sum for each student of the number of activity, self and peer assessment, participation and bonification points (`Total_AP`, `Total_SPA`, `Total_PP` and `Total_Bon`, repectively). The code to generate these features is shown in Code Snippet A.4, that describes the processing phase 2 title of notebook for general data processing.

Code Snippet A.4: Processing datasets and generating the engineered features with the somatory of points.

```
1 for year, dataset in datasets.items():
2     print(year)
3
4     # Extract the relevant lists from all_lists
5     listSyncColumns = all_lists[year].get('basic_lists').get('listSyncColumns')
6     listPPointsColumns = all_lists[year].get('basic_lists').get('listPPointsColumns')
7     listSPAColumns = all_lists[year].get('basic_lists').get('listSPAColumns')
8     listBonColumns = all_lists[year].get('basic_lists').get('listBonColumns')
```

```
    integerColumns = all_lists[year].get('basic_lists').get('listSyncNumericColumns').copy
        ↪ ()

    # Check if 'FinalGradeInteger' is in the dataset columns before processing
    if 'FinalGradeInteger' not in integerColumns:
        print(f"'FinalGradeInteger' is missing in the dataset for {year}.")
        continue

    integerColumns.remove('FinalGrade')

    # Ensure all integer columns are present in the dataset before applying pd.to_numeric
    missing_columns = [col for col in integerColumns if col not in dataset.columns]
    if missing_columns:
        print(f"Columns {missing_columns} are missing in the dataset for {year}.")
        continue

    datasetNumeric[year] = dataset[integerColumns].apply(pd.to_numeric, errors='coerce').
        ↪ fillna(0).astype('int64')

    # Join the 'FinalGrade' column back to the dataset
    if 'FinalGrade' in dataset.columns:
        datasetNumeric[year] = datasetNumeric[year].join(dataset['FinalGrade'])
    else:
        print(f"'FinalGrade' is missing in the dataset for {year}.")
        continue

    datasetAsyncClassesOnly = datasets[year].dropna(subset=all_lists[year].get('
        ↪ basic_lists').get('listAsyncColumns'), how='all').copy()
    display(datasetNumeric[year].describe())
    print("Number of Students only on Sync Classes = {}".format(len(datasetNumeric[year]))
        ↪ )

    # Calculate 'Total_AP'
    if all_lists[year].get('basic_lists').get('listSyncColumns'):
        datasetNumeric[year]['Total_AP'] = datasetNumeric[year][all_lists[year].get('
            ↪ basic_lists').get('listSyncColumns')[0]]
        for i in range(1, len(all_lists[year].get('basic_lists').get('listSyncColumns'))):
            datasetNumeric[year]['Total_AP'] += datasetNumeric[year][all_lists[year].get('
                ↪ basic_lists').get('listSyncColumns')[i]]

    # Calculate 'Total_PP'
    if all_lists[year].get('basic_lists').get('listPPointsColumns'):
        datasetNumeric[year]['Total_PP'] = datasetNumeric[year][all_lists[year].get('
            ↪ basic_lists').get('listPPointsColumns')[0]]
        for i in range(1, len(all_lists[year].get('basic_lists').get('listPPointsColumns'))
            ↪ ):
            datasetNumeric[year]['Total_PP'] += datasetNumeric[year][all_lists[year].get('
                ↪ basic_lists').get('listPPointsColumns')[i]]

    # Calculate 'Total_SPA'
```

```
    if all_lists[year].get('basic_lists').get('listSPAColumns'):
        datasetNumeric[year]['Total_SPA'] = datasetNumeric[year][all_lists[year].get('
            ↪ basic_lists').get('listSPAColumns')[0]]
        for i in range(1, len(all_lists[year].get('basic_lists').get('listSPAColumns'))):
            datasetNumeric[year]['Total_SPA'] += datasetNumeric[year][all_lists[year].get('
                ↪ basic_lists').get('listSPAColumns')[i]]

    # Calculate 'Total_Bon'
    if all_lists[year].get('basic_lists').get('listBonColumns'):
        datasetNumeric[year]['Total_Bon'] = datasetNumeric[year][all_lists[year].get('
            ↪ basic_lists').get('listBonColumns')[0]]
        for i in range(1, len(all_lists[year].get('basic_lists').get('listBonColumns'))):
            datasetNumeric[year]['Total_Bon'] += datasetNumeric[year][all_lists[year].get('
                ↪ basic_lists').get('listBonColumns')[i]]
```

Regarding feature engineering, weekly aggregated points (e.g., Week1_AP, Week2_PP) were also created and a column was made to indicate if a student passed (`Passed` = 1) or failed this course (which was maintained as an integer, even though it was categorical).

After all missing values on integer columns are replaced with zero as shown in line Code Snippet A.4, the code provided in Code Snippet A.5 is used to identify missing values in the dataset.

Code Snippet A.5: Check missing values per dataset column.

```
def check_missing_values_per_column(dataset_complete):
    for year, dataset in dataset_complete.items():
        print(f"Missing values in dataset for {year}:")
        print(dataset.isnull().sum()[dataset.isnull().sum() > 0])

check_missing_values_per_column(dataset_complete)
```

The code shown in Code Snippet A.6 was created to fill missing category data for the mode and float and integer missing data for the mean of a specific column, as is typical practice. However, it was not used because, with the exception of columns SS, `FinalGradeInteger`, and `FinalGrade`, most of the columns included complete data. It was decided not to use the special status column in the project analysis and to only use the final grade columns when absolutely necessary to anticipate these values and to train models with only the passed students. In this specific scenario, the missing data corresponds to all students who failed the discipline; hence, the value is unknown, but it is known to be less than 10. Therefore, the final grade's mean would not accurately reflect these absent samples.

Code Snippet A.6: Method to fill missing values with mode or mean, depending on the column Python type.

```
def fill_missing_values_appropriately(dataset_complete):
    filled_datasets = {}
    for year, dataset in dataset_complete.items():
```

```
        for column in dataset.columns:
            if dataset[column].dtype.name == 'category':
                mode_value = dataset[column].mode()[0]
                dataset[column].fillna(mode_value, inplace=True)
            elif dataset[column].dtype.name in ['float64', 'int64']:
                mean_value = dataset[column].mean()
                dataset[column].fillna(mean_value, inplace=True)
        filled_datasets[year] = dataset
        print(f"Filled missing values in dataset for {year}.")
    return filled_datasets

filled_datasets = fill_missing_values_appropriately(dataset_complete)
```

Handling Outliers

Outliers are controlled by capping values at the first and 99th percentiles. This technique corrects extreme values while retaining the majority of the data distribution. The code in Code Snippet A.7 was used to check the outliers, but it was verified that the considered outliers are the students that had zero points, which corresponds to students that did not try to make the discipline (part of the students that have missing data for final grades).

Code Snippet A.7: Method for outlier identification and counting using interquartile range.

```
def handle_outliers(data_dict):
    outlier_info = {}
    for year, df in data_dict.items():
        outlier_info[year] = {}
        columns = ['Total_AP', 'Total_PP', 'Total_SPA', 'Total_Bon']
        for col in columns:
            Q1 = df[col].quantile(0.25)
            Q3 = df[col].quantile(0.75)
            IQR = Q3 - Q1
            lower_bound = Q1 - 1.5 * IQR
            upper_bound = Q3 + 1.5 * IQR
            lower_cap = df[col].quantile(0.01)
            upper_cap = df[col].quantile(0.99)
            df[col] = np.where(df[col] < lower_bound, lower_cap, df[col])
            df[col] = np.where(df[col] > upper_bound, upper_cap, df[col])
            outlier_info[year][col] = {
                'lower_bound': lower_bound,
                'upper_bound': upper_bound,
                'lower_cap': lower_cap,
                'upper_cap': upper_cap,
                'outliers_lower': (df[col] == lower_cap).sum(),
                'outliers_upper': (df[col] == upper_cap).sum()
            }
    return data_dict, outlier_info
```

```
25
26 dataset_corrected, outlier_details = handle_outliers(dataset_complete_no_ss)
```

Verifying and Correcting Activities

The notebook verifies and corrects activities based on their types, ensuring that all columns are correctly characterized and updating lists as needed, using the code given in Code Snippet A.8. It compares if a column's maximum value match the activity typology indicated on the CTCT course activity plans.

Code Snippet A.8: Verify and correct activities, comparing the expected typology of activities with the results from the datasets received.

```
1  def verify_and_correct_activities(dataset_complete, all_lists):
2      for year, dataset in dataset_complete.items():
3          print(f"\nYear: {year}\n")
4          basic_list = all_lists[year]['basic_lists']
5          activity_types = all_lists[year]['activity_type_lists']
6          weekly_attrib_list = all_lists[year]['weekly_and_participation_lists']
7          created_columns = all_lists[year]['created_columns']
8          excluded_columns = [item for sublist in created_columns.values() for item in
               ↪ sublist]
9          listEvTypeA = activity_types['EvTypeA']
10         listEvTypeB = activity_types['EvTypeB']
11         listEvTypeC = activity_types['EvTypeC']
12         listEvNE = activity_types['EvNE']
13         listCategorical = basic_list['listCategorical']
14         listBonColumns = basic_list['listBonColumns']
15         listSPAColumns = basic_list['listSPAColumns']
16         listPPointsColumns = basic_list['listPPointsColumns']
17         listGrades = basic_list['listGrades']
18         listSyncColumns_withoutIdentification = [
19             col for col in dataset.columns
20             if 'ID' not in col and not col.startswith('Unnamed') and col not in
                   ↪ excluded_columns
21         ]
22
23         print("Activities of type A, B and C that are not in the dataset:")
24         print("Type A missing:", [i for i in listEvTypeA if i not in
               ↪ listSyncColumns_withoutIdentification])
25         print("Type B missing:", [i for i in listEvTypeB if i not in
               ↪ listSyncColumns_withoutIdentification])
26         print("Type C missing:", [i for i in listEvTypeC if i not in
               ↪ listSyncColumns_withoutIdentification])
27         uncharacterized_activities = [
28             i for i in listSyncColumns_withoutIdentification
29             if i not in (listEvTypeA + listEvTypeB + listEvTypeC + listEvNE +
                   ↪ listCategorical + listBonColumns + listSPAColumns + listPPointsColumns +
                   ↪  listGrades + excluded_columns)
30         ]
```

```
        print("\nActivities that are in the dataset but not characterized:",
            ↪ uncharacterized_activities)
        print("\nMax values for potentially mischaracterized columns:")
        for activity in uncharacterized_activities:
            if activity in dataset.columns:
                if pd.api.types.is_categorical_dtype(dataset[activity]):
                    if dataset[activity].cat.ordered:
                        print(f"{activity}: {dataset[activity].max()}")
                    else:
                        print(f"{activity}: Cannot compute max, unordered categorical data")
                else:
                    print(f"{activity}: {dataset[activity].max()}")

verify_and_correct_activities(dataset_complete, all_lists)
```

This structured and thorough approach ensures that the datasets are clean and well-organized, allowing for reliable and comprehensive data analysis. However, some preparation steps, such as scaling and encoding categorical variables, are not included in this notebook because they were decided to be implemented in the same notebook as the training, testing, and evaluation of a specific model for a specific analysis.

A.3 Data Understanding

The `data_understanding` Jupyter notebook generates appropriate statistics and graphical representations, providing insights into the dataset, as it's possible to see in the `data_understanding` notebook file steps, which are presented in Table A.6. This section highlights the important processes taken within the notebook that are required for comprehending and preparing educational datasets spanning numerous academic years, from 2015/16 to 2023/24.

Table A.6: Steps in `data_understanding` Google Colaboratory Notebook.

Step	Description
Importing Libraries	Import essential libraries for data manipulation (pandas, NumPy), visualization (Matplotlib, Seaborn), machine learning (sklearn), and other utilities.
Data Loading	Load datasets from .feather format and pickle files, ensuring efficient data processing.
Descriptive Statistics	Generate basic statistics for each dataset to understand data distribution and variable characteristics.

Continued on next page

Table A.6: (continued)

Step	Description
Analysis of Passed and Failed Students	Segregated data into passed and failed groups to identify factors influencing performance.
Correlation Analysis	Conduct correlation analysis using heatmaps to identify relationships between educational attributes.
Label Encoding	Convert categorical labels into numerical labels using `LabelEncoder` for machine learning compatibility.
Feature Selection	Use techniques like Variance Threshold and Mutual Information to identify influential attributes.
Hierarchical Clustering	Apply hierarchical clustering to explore data structure and group similar variables.
Statistical Testing	Conduct various statistical tests such as Mann-Whitney U test, Chi-squared tests, and Shapiro-Wilk tests to validate assumptions or inferences.
Results Handling	Systematically presenting test results, ensuring comprehensive documentation and interpretability.
Visualization	Use various visualization techniques to enhance data understanding.
Data Sufficiency Verification	Verify datasets have sufficient non-NA values and unique values for statistical tests.
Saving Results	Saving visualizations and results into Word documents and other formats for documentation and sharing.
Custom Print Functions	Define custom functions for printing and displaying outputs, capturing them into Word documents if needed.
Saving Results	Saving all visualizations and results into a zip file for documentation and sharing.

In addition to the data_understanding notebook, Data_Analysis_Report Jupyter notebook file was developed to improve some of the previous notebook and to do some preliminary feature selection and clustering analysis. It imports and runs methods from an external Python file, incorporating the findings of the data_understanding notebooks with enhanced charts, as well as some methods for clustering, feature selection, and prediction to obtain a general overview of the data before generating more focused notebooks to address fourth and fifth research questions. Although it was thought that this

method would be preferable than showing the results to those who are not interested in the produced code, it was discovered that this strategy required significantly more time to develop than simply running the methods on the notebook, for debugging purposes.

Used Libraries

The `data_understanding` Jupyter file notebook incorporates a variety of Python libraries to facilitate comprehensive data analysis and visualization within the Google Colaboratory environment. The core libraries, Pandas and NumPy, are used for data manipulation and numerical computing and Pandas provides robust tools for data manipulation through DataFrame structures, enabling efficient data cleaning and analysis. NumPy enhances these capabilities with support for large, multi-dimensional arrays and matrices, essential for complex numerical computations.

Visualization of data is achieved through libraries such as Matplotlib and Seaborn. Matplotlib offers extensive functionalities for creating a range of visualizations from static to interactive plots, while Seaborn builds on Matplotlib by providing a high-level interface for drawing attractive and informative statistical graphics. These tools are indispensable for exploring the underlying patterns and insights within the data.

For data serialization, the Pickle module is employed to serialize and deserialize Python object structures, ensuring the persistence and transferability of data states. Operating system dependent functionalities, such as file handling, are managed by the OS library, with the Shutil library providing high-level operations like file copying and removal. These operations are vital for managing the data workflow and storage efficiently.

In the realm of machine learning, preprocessing and model evaluation tools from Scikit-learn, such as LabelEncoder and SimpleImputer, play a pivotal role. LabelEncoder assists in converting categorical labels into a numeric format, while SimpleImputer facilitates handling missing values through various imputation techniques. Feature selection is conducted using methods like VarianceThreshold and mutual information, which help in reducing dimensionality and focusing on the most impactful variables.

Statistical analysis is enhanced by libraries such as SciPy, which offers functions for conducting a range of statistical tests and hierarchical clustering. Fastcluster, optionally installed via pip, provides a performance enhancement for hierarchical clustering methods, which is beneficial for handling large datasets efficiently.

Integration with Google Colaboratory is seamlessly maintained through the Google Colab files library, which supports file uploads and downloads directly within the notebook environment. Additionally, the Statsmodels library provides comprehensive options for statistical modeling, crucial for in-depth data analysis and interpretation.

For interactive data analysis, IPyWidgets offers a range of widgets that enhance user interaction within Jupyter notebooks, facilitating dynamic data exploration. Date and time manipulations are efficiently handled by the datetime library, which is essential for analyses involving time-series data.

Overall, the integration of these diverse libraries ensures that the `data_understanding` Jupyter notebook file is well-equipped to perform detailed data analysis and visualization, supporting robust educational data analysis within an interactive cloud-based environment.

Statistical Analysis

The notebook contains a full statistical analysis. This involves providing descriptive statistics that reveal the dataset's central tendency, dispersion, and distribution shape. Correlation analysis is also carried out, using correlation heatmaps to determine the strength and direction of correlations between variables. A concentrated examination of passed and failed students is conducted to separate the data into different groups, showing differences and similarities that may influence performance outcomes.

Pairplots and multivariate analysis are used to demonstrate complex interactions between several factors, which are useful for identifying potential implications on student performance. Figures A.2, A.3, and A.4 show these graphical representations.

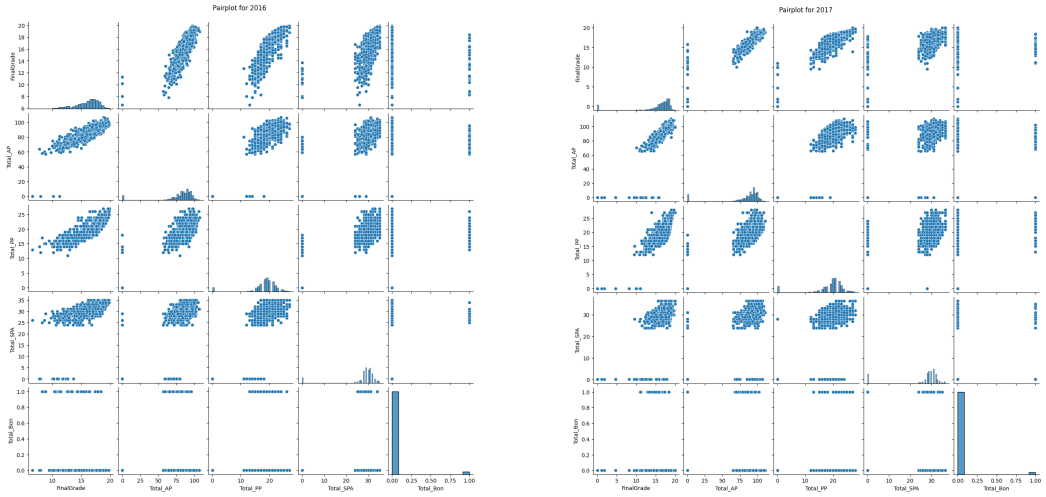


Figure A.2: Pairplots between `Total_AP`, `Total_PP`, `Total_SPA`, `Total_Bon`, and `FinalGrade` attributes for the years 2016 and 2017.

Statistical Tests

Statistical tests are included in the notebook to properly validate the analyses. Mann-Whitney U tests compare distributions of two independent samples and are effective for determining grade differences across demographic groupings. Chi-squared tests determine the independence of categorical variables, which aids in identifying connections between demographic characteristics. Normality tests, notably the Shapiro-Wilk test, evaluate the normality of data distributions, which is an important step in determining the appropriateness of parametric testing.

Prior to executing statistical tests, the notebook validates that the data is enough. Functions are built to ensure that the data contains a sufficient number of non-NA (not

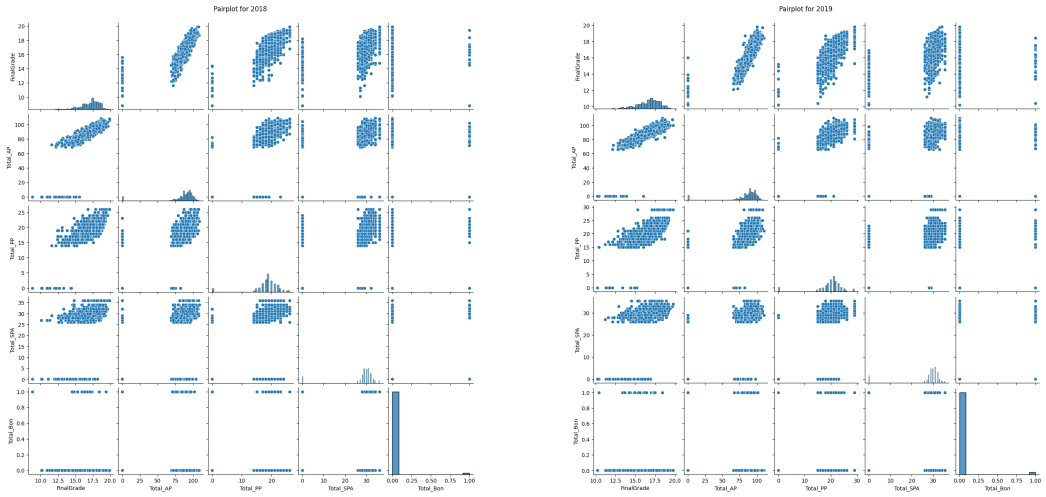


Figure A.3: Pairplots between `FinalGrade`, `Total_AP`, `Total_PP`, `Total_SPA`, and `Total_Bon` attributes for the years 2018 and 2019.

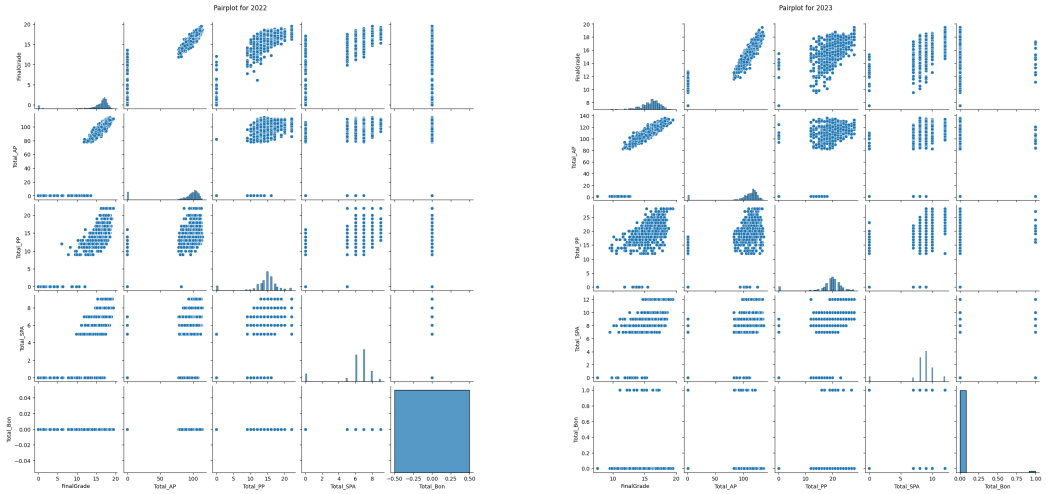


Figure A.4: Pairplots between `FinalGrade`, `Total_AP`, `Total_PP`, `Total_SPA`, and `Total_Bon` for the years 2022 and 2023.

available) values and unique values to support rigorous statistical tests. The data is additionally preprocessed to ensure that each categorical bin has the minimal predicted frequency count, which is critical for the validity of Chi-squared tests.

Results Handling and Visualization

The results of these analyses are presented and documented in a methodical manner. Descriptive and comparative analysis filters and isolates particular data features to enable targeted statistical and visual studies. Mann-Whitney U and Chi-squared analyses offer comprehensive insights into performance discrepancies and potential biases that are essential for well-informed decision-making in educational initiatives.

The notebook uses a variety of graphical representations to help in data visualization. While box plots depict the final grade distribution by student sex and course to provide insights into data variation and outliers, histograms are used to display the distribution of

single variables. In order to provide a visual representation of possible links, heatmaps are created to show the correlation between various variables. Plotting correlations between two variables using a scatter plot makes it easier to spot patterns or trends. Pie charts are also used to show how categorical data is distributed, giving proportions and categories an easy-to-understand visual representation.

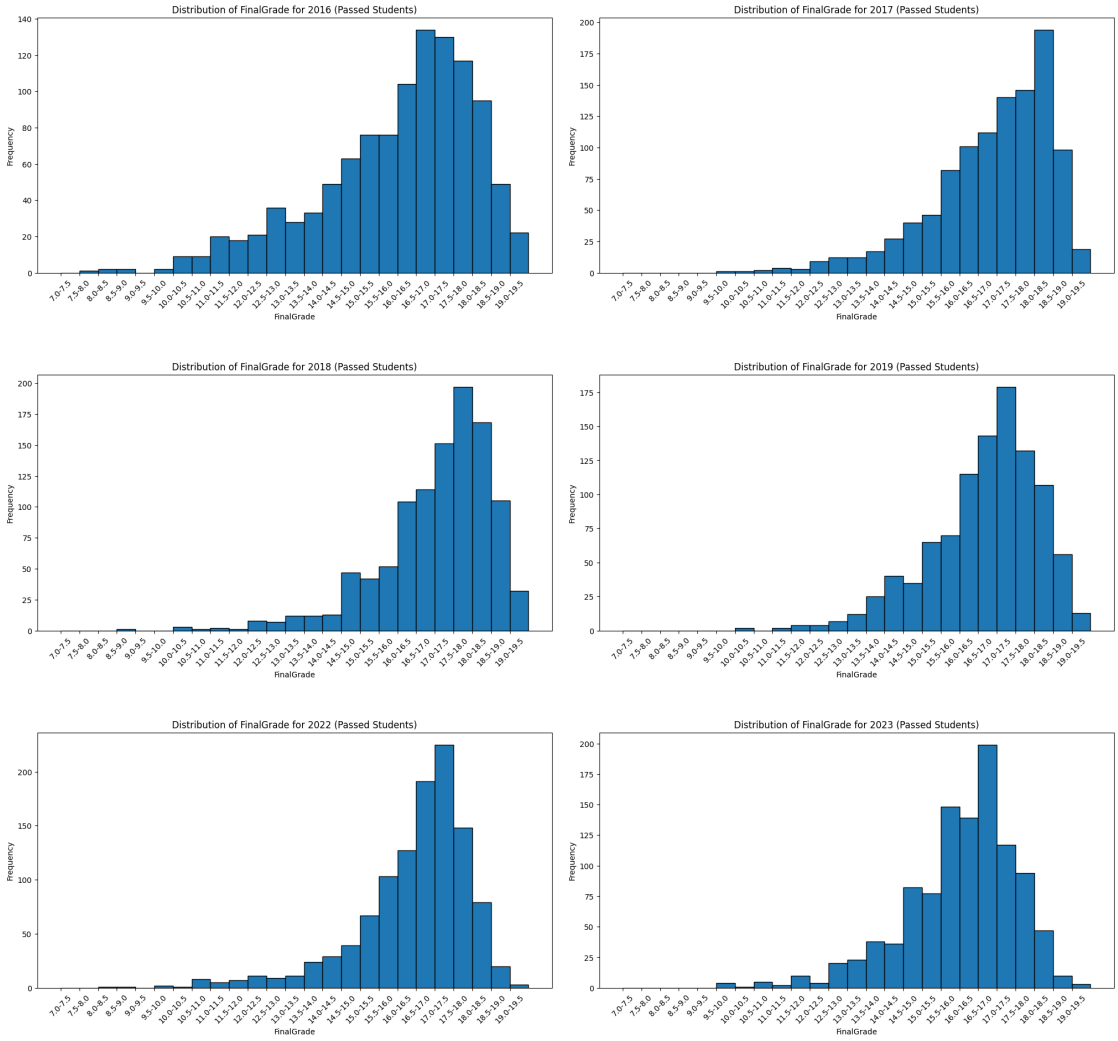


Figure A.5: Histograms representing the distribution of students final grades for the students that didn't reprove (`FinalGrade`).

In addition to the standard visual outputs, the notebook has produced further graphical representations that extend beyond those displayed in Chapter 4. Specifically, Figures A.6, A.7, A.8, and A.9 depicts the distribution of activity points and participation points, respectively. These charts represent the aggregated scores from all activities across various editions, offering a detailed visual analysis of the data collected.

These visualizations complement the statistical analyses with visual evidence that can be more intuitively understood, aiding in deeper insights and more informed decision-making.

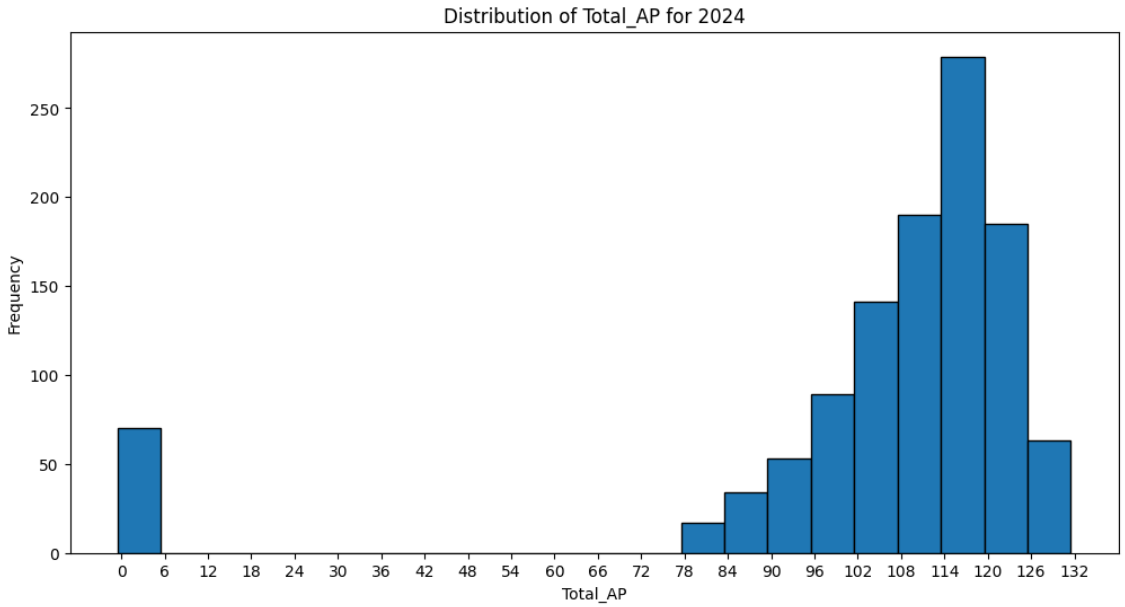


Figure A.6: Histograms representing the distribution of the total value of activity points (`Total_AP`) for 2023-24 school year.

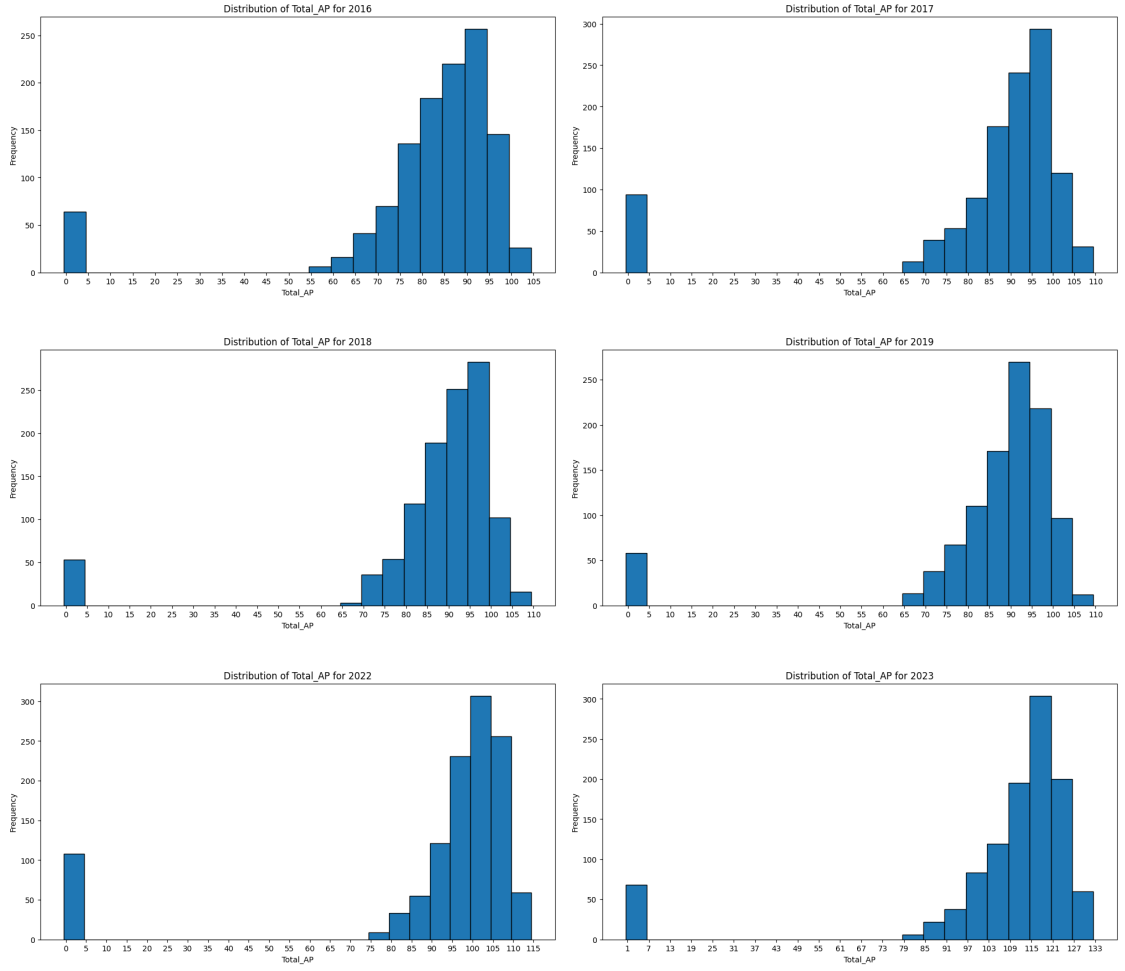


Figure A.7: Histograms representing the distribution of the total value of activity points (`Total_AP`).

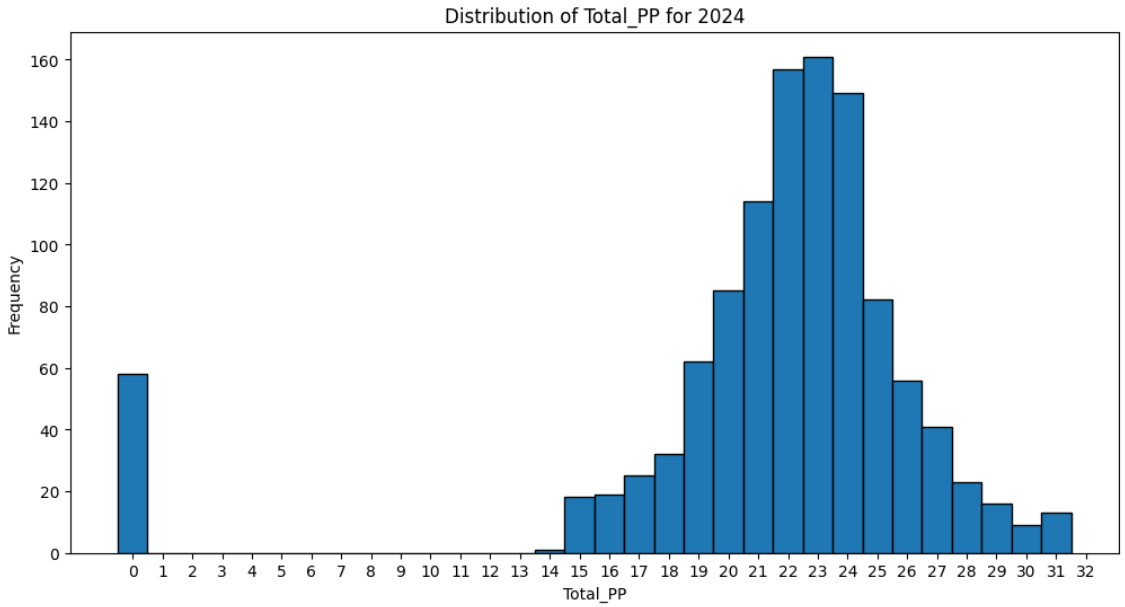


Figure A.8: Histograms representing the distribution of the total value of participation points (`Total_PP`) for 2023-24 school year.

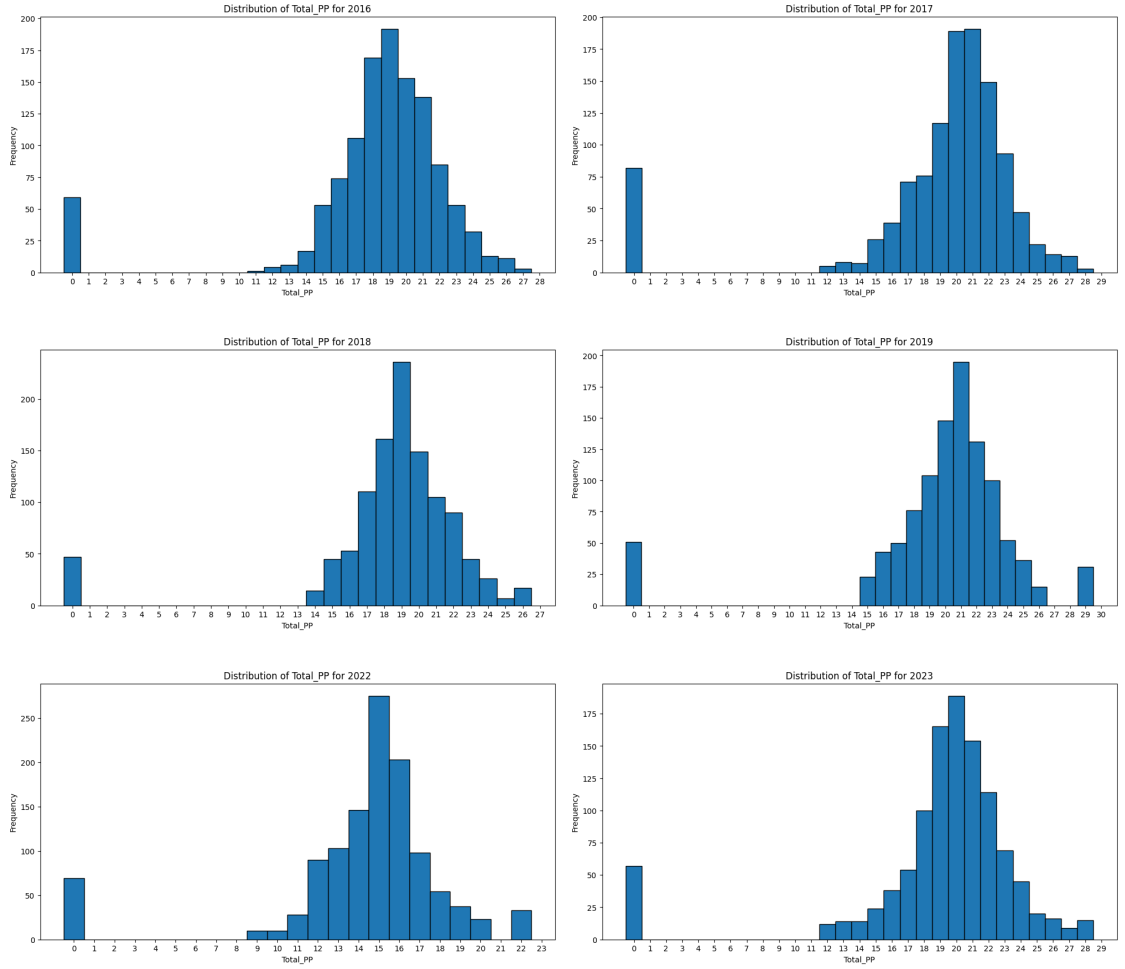


Figure A.9: Histograms representing the distribution of the total value of participation points (`Total_PP`).

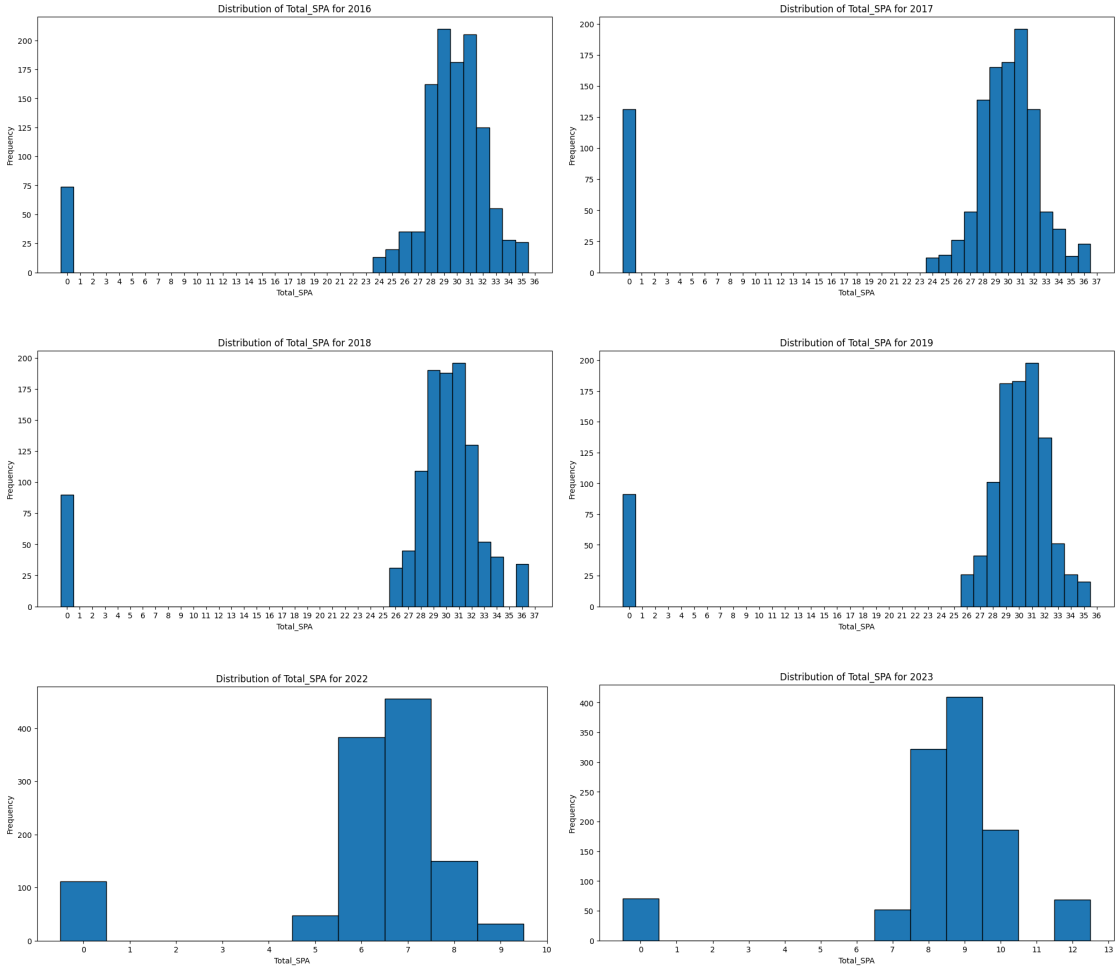


Figure A.10: Histograms representing the distribution of the total value of self and peer assessment points (`Total_SPA`).

Answer to Research Questions

Analysis of gender on university degree selection

To understand how gender impacts course selection, to be able to answer the first research question, the `data_understanding` notebook analyzed the distribution of male and female students across various courses over multiple academic years. Gender distributions were visualized using bar charts and pie charts for each year, revealing distinct patterns. The Chi-squared test was employed to determine if the observed distributions of gender within each course were independent of the courses chosen. The results indicated significant differences in course selection between male and female students. Specifically, the p-values obtained from the Chi-squared tests were extremely small, suggesting a strong association between gender and course choice. These findings highlight the presence of gender-specific preferences or tendencies in course selection at the NOVA School of Science and Technology.

Examination of the relationship between each assessment point's overall value

The steps employed to address the third research question are outlined in this subsection: What are the interrelationships and correlation strengths between Activity Points, Participation Points, Self and Peer Assessment Points and Final Grade?

The relationships between activity points (AP), participation points (PP), and self and peer assessment Points (SPA) were explored through correlation analysis across different academic years. Heat maps of correlation matrices were generated to visualize these relationships. The analyses revealed varying degrees of correlation among AP, PP, and SPA points across the years. For instance, certain years exhibited strong correlations, indicating that students who performed well in one type of assessment tended to perform well in others. Additionally, the correlation of these points with the Final Grade was examined. The findings showed that all three types of points had significant positive correlations with the Final Grade, suggesting that higher participation and engagement in activities and assessments were associated with better final grades. This analysis provides valuable insights into how different assessment components contribute to overall student performance in the CTCT discipline.

Examination of differences in student performance as a function of several factors

The steps employed to address the third research question are outlined in this subsection: How do average points represent student success in the CTCT discipline across editions, student degree (`Course`), gender (`Sex`), weeks, and themes?

In general, female students tend to score higher in AP and SPA across multiple years, according to the description of data grouped by sex. This could be an indication of consistent engagement and evaluative success in comparison to their male counterparts. In addition to the information provided in Subsection 4.2.3.1, Table A.7 summarizes the average of all values of each assessment point for each sex for each of the seven curricular unit editions that were examined.

Table A.7: Summary of the average and standard deviation of the total of points of each type of assessment.

Year	Gender	Mean_AP	Mean_PP	Mean_SPA	Standard Deviation
2016	Female	83.43 (0, 107)	18.84 (0, 27)	29.05 (0, 35)	19.32, 4.67, 6.94
	Male	79.52 (0, 105)	17.94 (0, 27)	27.43 (0, 35)	22.33, 4.99, 7.86
2017	Female	88.41 (0, 109)	19.76 (0, 28)	27.95 (0, 36.49)	22.10, 5.06, 9.29
	Male	81.64 (0, 111)	18.39 (0, 28)	25.97 (0, 36.49)	28.18, 6.16, 10.02
2018	Female	89.44 (0, 105)	19.04 (0, 26)	29.89 (0, 36)	16.17, 3.85, 5.92

Continued on next page

Table A.7: (continued)

Year	Gender	Mean_AP	Mean_PP	Mean_SPA	Standard Deviation
	Male	85.11 (0, 109)	18.16 (0, 26)	26.71 (0, 36)	22.86, 4.83, 9.49
2019	Female	86.69 (0, 107)	20.35 (0, 29)	28.55 (0, 35.46)	21.31, 4.92, 8.28
	Male	84.37 (0, 110)	19.48 (0, 29)	27.06 (0, 35.46)	22.53, 5.40, 8.89
2022	Female	94.46 (0, 113)	14.75 (0, 22)	6.44 (0, 9)	25.02, 3.90, 1.90
	Male	88.63 (0, 114)	14.03 (0, 22)	5.93 (0, 9)	31.91, 4.42, 2.24
2023	Female	109.42 (1, 135)	19.37 (0, 28)	8.69 (0, 12)	25.41, 4.96, 2.25
	Male	105.65 (1, 136)	18.85 (0, 28)	8.24 (0, 12)	30.48, 5.39, 2.53
2024	Female	108.19 (0, 129)	22.24 (0, 31)	8.87 (0, 12)	23.84, 4.87, 2.16
	Male	102.21 (0, 131)	20.94 (0, 31)	8.10 (0, 12)	31.16, 6.25, 2.64

The analysis of gender-based performance in FCT faculty, as presented in Table A.8, reveals noteworthy patterns in a range of science and engineering fields. Notably, during a number of years, female students have regularly outscored male students in Electrical and Computer Engineering (LEEC), indicating a significant gender performance gap. In contrast, there are not much gender variations in performance on CTCT discipline for degrees like biochemistry (LBq) and informatics engineering (LEI).

Table A.8: Mean grade by gender and course.

Attribute	Gender	2016	2017	2018	2019	2022	2023	2024
Course: Mechanical Engineering (LEMc)								
Gender	Female	85.2%	89.7%	81.5%	87.7%	85.9%	78.6%	85.7%
Proportion	Male	14.8%	10.3%	18.5%	12.3%	14.1%	21.4%	14.3%
	Female	16.6	17.3	17.6	16.7	17.0	16.0	15.6
Mean Grade	Course	15.6	17.0	17.0	16.5	15.9	15.9	15.2
	Male	15.4	16.6	16.9	16.4	16.4	15.8	15.2
Course: Civil Engineering (LEC)								
Gender	Female	89.7%	89.1%	82.3%	78.4%	78.9%	69.8%	69.2%

Continued on next page

Table A.8: (continued)

Attribute	Gender	2016	2017	2018	2019	2022	2023	2024
Proportion	Male	10.3%	10.9%	17.7%	21.6%	21.1%	30.2%	30.8%
	Female	13.9	17.3	17.1	16.2	16.3	15.3	14.9
Mean Grade	Course	14.5	14.7	16.2	16.0	13.4	15.0	14.8
	Male	14.6	15.9	15.9	16.0	15.6	14.9	14.8
Course: Informatic Engineering (LEI)								
Gender	Female	80.9%	85.2%	91.2%	86.0%	87.6%	89.3%	86.1%
Proportion	Male	19.1%	14.8%	8.8%	14.0%	12.4%	10.7%	13.9%
	Female	16.5	17.9	17.3	16.7	16.9	16.7	15.5
Mean Grade	Course	16.2	15.0	17.1	16.8	15.7	16.2	15.5
	Male	16.1	16.7	17.1	16.8	16.7	16.1	15.5
Course: Electrical and Computer Engineering (LEEC)								
Gender	Female	88.6%	90.6%	85.5%	85.0%	92.4%	92.7%	89.4%
Proportion	Male	11.5%	9.4%	14.5%	15.0%	7.6%	7.3%	10.6%
	Female	17.0	17.0	17.4	16.7	16.3	16.3	15.2
Mean Grade	Course	15.9	16.0	16.9	16.5	15.1	15.7	14.8
	Male	15.7	16.7	16.9	16.5	16.2	15.6	14.8
Course: Biochemistry (LBq)								
Gender	Female	61.3%	66.7%	59.8%	67.5%	74.2%	70.4%	61.5%
Proportion	Male	38.7%	33.3%	40.2%	32.5%	25.8%	29.6%	38.5%
	Female	16.6	17.3	17.5	16.9	17.0	16.3	15.8
Mean Grade	Course	16.3	16.6	17.3	16.7	15.6	16.1	15.5
	Male	15.8	16.9	17.1	16.4	16.6	15.6	15.0
Course: Chemical and Biochemical Engineering (LEQB)								
Gender	Female	70.0%	69.7%	52.1%	70.8%	65.9%	59.7%	76.1%

Continued on next page

Table A.8: (continued)

Attribute	Gender	2016	2017	2018	2019	2022	2023	2024
Proportion	Male	30.0%	30.3%	48.0%	29.2%	34.2%	40.3%	23.9%
	Female	16.3	17.3	16.4	16.7	16.9	15.7	15.3
Mean Grade	Course	16.4	17.0	16.6	16.6	15.7	15.7	15.4
	Male	16.6	17.1	16.7	16.4	16.2	15.7	16.0
Course: Cellular and Molecular Biology (LBCM)								
Gender	Female	75.3%	63.5%	69.0%	71.9%	66.7%	71.0%	69.6%
Proportion	Male	24.7%	36.5%	31.0%	28.1%	33.3%	29.0%	30.4%
	Female	17.1	17.4	17.8	17.3	17.0	16.5	15.5
Mean Grade	Course	17.0	17.6	17.9	17.2	16.8	16.5	15.5
	Male	16.5	17.9	18.1	17.0	17.0	16.3	15.6
Course: Applied Chemistry (LQA)								
Gender	Female	60.0%	66.7%	77.8%	75.0%	56.8%	74.2%	56.3%
Proportion	Male	40.0%	33.3%	22.2%	25.0%	43.2%	25.8%	43.8%
	Female	16.8	17.6	17.0	16.6	16.6	15.9	15.6
Mean Grade	Course	16.6	16.9	17.2	16.4	16.9	16.1	15.3
	Male	16.3	16.5	17.6	16.0	17.2	17.0	15.1
Course: Environmental Engineering (LEA)								
Gender	Female	50.0%	60.0%	58.2%	52.6%	58.8%	54.7%	50.9%
Proportion	Male	50.0%	40.0%	41.8%	47.4%	41.2%	45.3%	49.1%
	Female	15.8	16.4	16.9	15.3	15.3	15.7	14.5
Mean Grade	Course	15.2	14.8	16.7	15.7	14.9	15.7	14.4
	Male	14.7	16.1	16.6	16.1	15.1	15.8	14.2
Course: Industrial Engineering and Management (LEGI)								
Gender	Female	57.6%	61.8%	57.4%	54.6%	60.9%	56.9%	66.2%

Continued on next page

Table A.8: (continued)

Attribute	Gender	2016	2017	2018	2019	2022	2023	2024
Proportion	Male	42.4%	38.2%	42.6%	45.5%	39.1%	43.1%	33.9%
	Female	16.4	16.9	17.0	16.6	16.6	16.3	15.6
Mean Grade	Course	15.8	15.0	16.5	16.4	15.0	16.1	15.0
	Male	15.4	16.7	16.1	16.2	15.8	15.9	14.7
Course: Biomedical Engineering (LEB)								
Gender	Female	71.2%	66.7%	64.1%	72.6%	77.3%	63.5%	68.3%
Proportion	Male	28.8%	33.3%	35.9%	27.4%	22.7%	36.5%	31.7%
	Female	16.9	17.2	17.3	17.3	16.7	16.1	15.9
Mean Grade	Course	16.7	16.6	17.4	17.2	16.3	16.2	15.9
	Male	16.2	17.8	17.6	17.1	17.0	16.4	16.0
Course: Physics Engineering (LEF)								
Gender	Female	76.0%	57.1%	69.2%	70.8%	61.3%	51.7%	87.5%
Proportion	Male	24.0%	42.9%	30.8%	29.2%	38.7%	48.3%	12.5%
	Female	16.6	18.0	18.1	17.5	17.0	16.2	14.8
Mean Grade	Course	16.2	15.5	17.6	17.3	15.9	16.3	15.8
	Male	16.1	16.9	17.4	17.2	16.1	16.3	15.9
Course: Micro and Nanotechnologies Engineering (LEMN)								
Gender	Female	73.7%	59.3%	52.7%	60.3%	63.3%	74.2%	71.7%
Proportion	Male	26.3%	40.7%	47.3%	39.7%	36.7%	25.8%	28.3%
	Female	15.8	16.7	16.7	16.9	16.4	15.3	14.8
Mean Grade	Course	15.8	15.9	17.1	16.8	16.1	15.6	14.4
	Male	15.8	16.8	17.5	16.7	16.6	15.7	14.2
Course: Mathematics (LM)								
Gender	Female	57.1%	68.8%	54.5%	60.0%	53.8%	57.7%	63.6%

Continued on next page

Table A.8: (continued)

Attribute	Gender	2016	2017	2018	2019	2022	2023	2024
Proportion	Male	42.9%	31.3%	45.5%	40.0%	46.2%	42.3%	36.4%
	Female	16.2	17.2	17.3	17.2	17.5	16.7	16.4
Mean Grade	Course	16.3	15.0	17.1	17.4	16.1	16.5	16.5
	Male	16.5	17.1	16.9	17.6	16.9	16.4	16.9
Course: Geological Engineering (LEG)								
Gender	Female	87.5%	64.0%	52.4%	75.0%	57.1%	68.4%	55.6%
Proportion	Male	12.5%	36.0%	47.6%	25.0%	42.9%	31.6%	44.4%
	Female	14.5	15.9	16.9	15.0	15.4	16.3	14.7
Mean Grade	Course	15.1	12.6	16.8	15.5	14.9	15.6	14.5
	Male	15.2	16.2	16.8	15.6	15.5	15.2	14.2
Course: Materials Engineering (LEMt)								
Gender	Female	64.0%	57.1%	80.8%	68.2%	84.0%	65.2%	69.6%
Proportion	Male	36.0%	42.9%	19.2%	31.8%	16.0%	34.8%	30.4%
	Female	16.7	17.5	16.3	16.6	16.3	16.2	15.2
Mean Grade	Course	15.1	15.0	16.7	16.5	15.8	15.9	14.9
	Male	14.3	17.6	16.8	16.4	15.7	15.8	14.7
Course: Mathematics Applied to Risk Management (LMAGR)								
Gender	Female	-	-	-	58.8%	52.0%	52.0%	56.5%
Proportion	Male	-	-	-	41.2%	48.0%	48.0%	43.5%
	Female	-	-	-	16.6	17.5	15.3	15.1
Mean Grade	Course	-	-	-	16.3	16.9	15.4	15.4
	Male	-	-	-	16.0	16.3	15.4	15.7
Course: Agro-Industrial Technology (LTAI)								
Gender	Female	-	-	-	-	-	-	60.6%

Continued on next page

Table A.8: (continued)

Attribute	Gender	2016	2017	2018	2019	2022	2023	2024
Proportion	Male	-	-	-	-	-	-	39.4%
	Female	-	-	-	-	-	-	14.9
Mean Grade	Course	-	-	-	-	-	-	15.1
	Male	-	-	-	-	-	-	15.3

As seen in Subsection 4.2.3.1, a portion of the statistical analysis uses ANOVA to examine the impact of course enrollment, gender, and their interaction on students' final grades. The `data_understanding.ipynb` notebook's implementation of this analysis is described in full in Code Snippet A.9. The dataset for each academic year is processed to compute and show the results of the ANOVA, making sure that any errors in computation are handled and do not interfere with the analysis of other datasets.

Code Snippet A.9: ANOVA analysis Python code.

```
for year, dataset in dataset_complete_updated.items():
    try:
        print("-----------------------------------------------")
        print(year)
        model = ols('FinalGrade ~ C(Course) + C(Sex) + C(Course):C(Sex)', data=dataset).fit
            ↪ ()
        anova_table = sm.stats.anova_lm(model, typ=2) # Type 2 ANOVA DataFrame
        print(anova_table)
    except Exception as e:
        print(f"Error processing year {year}: {e}")
```

ANOVA Type 2 is employed in this study. Unlike Type 1 ANOVA, which assesses each variable's influence in the order in which it is entered into the model, Type 2 ANOVA evaluates each variable's major effects after the others have been accounted for, making it ideal for unbalanced designs or models containing interactions. The results in Table A.9 show the statistical importance of course, gender, and their interaction over academic years.

Table A.9: ANOVA results for study of the impact of course enrollment, gender, and their interaction on students' final grades.

Year	Sum of Squares	Degrees of Freedom	F-Statistic	p-value
2016	271.46, 72.66, 73.96	15, 1, 15	4.35, 17.47, 1.19	5.64×10^{-8}, 3.15×10^{-5}, 0.276
2017	869.71, 93.50, 465.59	15, 1, 15	3.01, 4.85, 1.61	9.3×10^{-5}, 0.0279, 0.0647
2018	156.83, 10.60, 55.92	15, 1, 15	4.98, 5.06, 1.78	1.49×10^{-9}, 0.0248, 0.0332
2019	158.73, 3.53, 20.86	16, 1, 16	4.90, 1.75, 0.64	8.78×10^{-10}, 0.187, 0.849
2022	652.77, 57.43, 78.01	16, 1, 16	2.83, 3.99, 0.34	1.52×10^{-4}, 0.046, 0.993
2023	119.08, 6.93, 37.15	16, 1, 16	3.33, 3.10, 1.04	1.0×10^{-5}, 0.0785, 0.412
2024	213.46, 6.36, 42.35	17, 1, 17	5.22, 2.64, 1.03	3.73×10^{-11}, 0.1045, 0.416

It was also studied the average of the total of each type of point among CTCT enrolled in the different university degree, as it's possible to see in Figures A.11, A.12, and A.13.

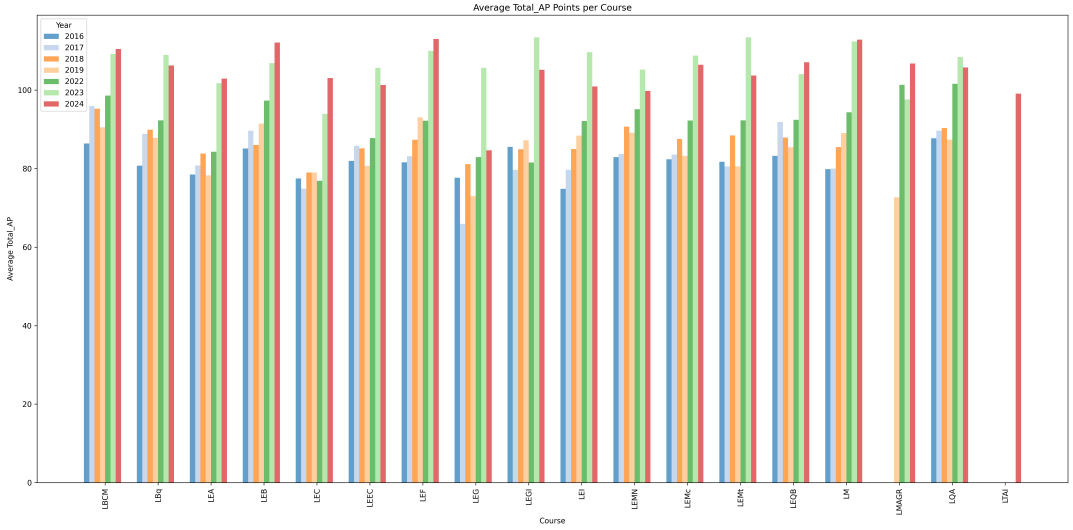


Figure A.11: Bar chart with courses performance in terms of `Total_AP` for all editions.

The activity results were also analyzed in accordance with the theme of the activity. Two distinct methods were used: the first included grouping activities results by week

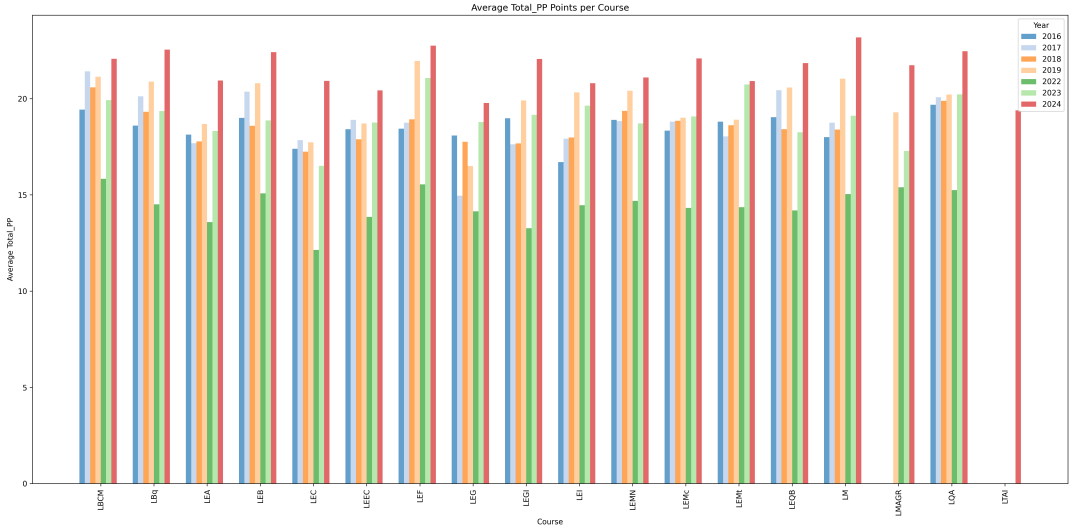


Figure A.12: Bar chart with courses performance in terms of `Total_PP` for all editions.

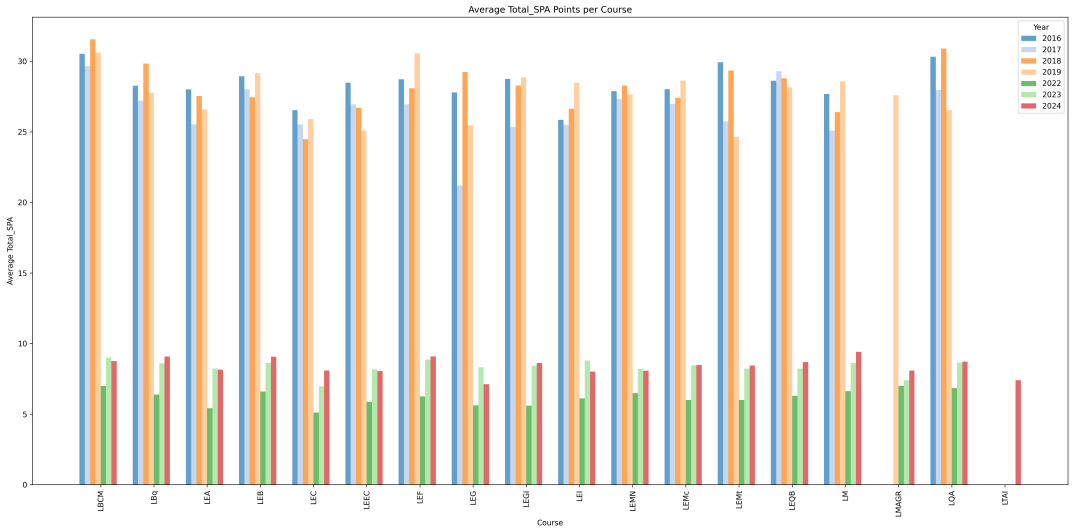


Figure A.13: Bar chart with courses performance in terms of `Total_SPA` for all editions.

using the lists specified in Table A.5, and the second involved grouping activities by theme using the activity_dictionary last field refered in Subsection A.3. Code Snippet A.10 displays the code used to compute the normalized average points per thematic area.

Code Snippet A.10: Calculate average normalized points by thematic area.

```
# Define the function to calculate average normalized points by thematic areas
def calculate_normalized_average_points_by_thematic_area(dataset, activity_dictionary):
    # Define the mapping from letters to maximum points
    max_points_mapping = {'A': 1, 'B': 3, 'C': 5}

    # Extract thematic areas and maximum points from the activity dictionary, skipping 'NE' activities
    thematic_areas = {activity: details[7] for activity, details in activity_dictionary.items()}
```

```
    max_points = {activity: max_points_mapping[details[2]] for activity, details in
        ↪ activity_dictionary.items() if details[2] in max_points_mapping} # Skip 'NE'
        ↪ activities

    # Initialize a dictionary to store average normalized points by thematic area for each
        ↪  student
    student_thematic_points = {thematic: [] for thematic in set(thematic_areas.values())}

    # Iterate over each student in the dataset
    for student_id, student_data in dataset.iterrows():
        # Initialize a dictionary to collect normalized points for each thematic area for
            ↪ the current student
        thematic_points = {thematic: [] for thematic in set(thematic_areas.values())}

        # Iterate over each activity in the activity dictionary
        for activity, details in activity_dictionary.items():
            activity_col = activity # Column name in the dataset for the activity

            # Check if the activity column exists in the student's data
            if activity_col in student_data.index:
                thematic = details[7] # Thematic area associated with the activity

                # Check if the thematic area is valid, the activity is not 'NE', and the
                    ↪ student's activity score is not NaN
                if thematic and details[2] in max_points_mapping and not pd.isna(
                    ↪ student_data[activity_col]):
                    max_point = max_points[activity] # Maximum points for the activity
                    # Normalize the student's score for the activity based on the maximum
                        ↪ points and append to the thematic list
                    normalized_score = student_data[activity_col] / max_point
                    thematic_points[thematic].append(normalized_score)

        # Calculate the mean normalized points for each thematic area for the current
            ↪ student
        for thematic, points in thematic_points.items():
            if points:
                # Calculate the mean of the normalized points if there are any points
                    ↪ collected
                student_thematic_points[thematic].append(np.mean(points))
            else:
                # Append NaN if there are no points for the thematic area
                student_thematic_points[thematic].append(np.nan)

    # Convert the dictionary to a DataFrame where each row corresponds to a student and
        ↪ each column to a thematic area
    return pd.DataFrame(student_thematic_points)
```

The Python function `calculate_normalized_average_points_by_thematic_area` illustrated in Code Snippet A.10 computes the average normalized points per thematic area based on the structured data provided in the `dataset` and the corresponding

`activity_dictionary`. Initially, the function establishes a mapping of maximum points for each activity based on a predefined scheme (`max_points_mapping`). It then extracts the thematic areas and applicable maximum points for activities, excluding those marked as 'NE'. An intermediate structure (`student_thematic_points`) is prepared to hold the average normalized points for each thematic area across all students. Within the function, for each student, it calculates the normalized score for each activity if valid scores are present and aggregates these scores by thematic area. Finally, the function returns a data frame where each row represents a student and each column corresponds to a thematic area, filled with the average normalized scores. This systematic approach aids in a detailed analysis of the performance of students across various thematic dimensions, aligning with the categorization detailed in Subsection A.3. Comparable techniques were created to display the average normalized scores by gender and by course.

Other Results

Analysis of Categorical Attributes

Table A.10 presents the results of the chi-squared tests used to examine the relationships between categorical attributes and binned numerical attributes that represent performance. Several important insights can be derived from this analysis. Firstly, it highlights the impact of sex on both course selection and student performance. Secondly, the analysis reveals the influence of frequency on various performance metrics. Lastly, it provides an understanding of longitudinal trends and educational strategies, offering potential directions for future academic planning and improvement.

Table A.10: Chi-squared test results between categorical attributes and the total of each type of assessment points binned (2016-2024)

Year	Variables	Statistic	P-value
2016	`Course` & `Sex`	285.298	6.30×10^{-52}
	`Course` & `freq`	49.316	0.0146
	`Class` & `Total_AP` (binned)	923.259	1.27×10^{-9}
	`Sex` & `Total_SPA` (binned)	81.923	1.77×10^{-12}
	`freq` & `Total_AP` (binned)	1124.656	9.20×10^{-210}
2017	`Course` & `Sex`	285.695	5.21×10^{-52}
	`Course` & `freq`	49.270	0.0147
	`Class` & `Total_AP` (binned)	993.379	3.46×10^{-17}

Continued on next page

Table A.10: (continued)

Year	Variables	Statistic	P-value
	Sex & Total_SPA (binned)	104.946	1.81×10^{-16}
	freq & Total_AP (binned)	966.970	3.58×10^{-178}
2018	Course & Sex	231.971	6.30×10^{-41}
	Course & freq	32.263	0.355
	Class & Total_AP (binned)	964.452	3.45×10^{-12}
	Sex & Total_SPA (binned)	76.434	2.50×10^{-12}
	freq & Total_AP (binned)	591.137	3.33×10^{-99}
2019	Course & Sex	258.037	1.16×10^{-45}
	Class & Sex	6.174	0.9999996
	Course & Total_AP (binned)	346.445	0.04707
	Course & Total_PP (binned)	255.979	0.01305
	Course & Total_SPA (binned)	232.174	0.000168
2022	Course & Sex	322.704	4.98×10^{-59}
	Course & freq	48.884	0.02847
	Sex & Total_AP (binned)	38.176	0.00367
	freq & Total_AP (binned)	715.607	3.09×10^{-127}
	freq & Total_PP (binned)	1047.879	2.61×10^{-204}
2023	Course & Sex	274.995	3.75×10^{-49}
	Course & Total_AP (binned)	361.077	0.01353
	Class & Total_AP (binned)	1287.417	7.47×10^{-48}
	freq & Total_SPA (binned)	728.923	3.87×10^{-150}
2024	Course & Sex	262.618	5.48×10^{-46}
	Class & Total_AP (binned)	932.715	6.38×10^{-23}
	Sex & Total_SPA (binned)	42.852	3.96×10^{-8}
	freq & Total_PP (binned)	1183.124	3.75×10^{-212}

To provide context on how the numerical attributes were split into different bins for

this analysis, it's presented Code Snippets A.11 and A.12.

Code Snippet A.11: Method to implement Chi-Squared statistical test to check the relation between categorical variables and binned numerical ones

```
def chi_squared_test_binned(dataset, cat_var, num_var, bins, doc=None):
  if cat_var in dataset.columns and num_var in dataset.columns:
    dataset['binned'] = pd.cut(dataset[num_var], bins)
    contingency_table = pd.crosstab(dataset[cat_var], dataset['binned'])
    chi2_stat, p_val, dof, _ = chi2_contingency(contingency_table)
    result = (
    f"Chi-squared test between {cat_var} and {num_var} (binned):\n"
    f"Chi-squared statistic: {chi2_stat}\n"
    f"Degrees of freedom: {dof}\n"
    f"P-value: {p_val}\n"
    )
    if doc:
      print_to_doc(doc, result)
    else:
      print(result)
  else:
    error_msg = f"One or both variables {cat_var} and {num_var} are not in the dataset."
    if doc:
      print_to_doc(doc, error_msg)
    else:
      print(error_msg)
```

The chi_squared_test_binned method represented in the Code Snippets A.11 performs a chi-squared test between a categorical variable and a binned numerical variable. The numerical variable is binned using the pd.cut function in pandas, which divides the data into discrete intervals. The calculate_bins Python method shown in the Code Snippets A.12 is used to determine the optimal number of bins based on the Interquartile Range (IQR) and the Freedman-Diaconis rule calculating one of the inputs to give to chi_squared_test_binned method.

Code Snippet A.12: Method used to determine the number of bins adequate to visualize a numerical variable distribution.

```
def calculate_bins(dataset, num_var):
  if num_var in dataset.columns:
    iqr = np.percentile(dataset[num_var], 75) - np.percentile(dataset[num_var], 25)
    bin_width = 2 * iqr / (len(dataset[num_var]) ** (1/3))
    data_range = dataset[num_var].max() - dataset[num_var].min()
    num_bins = int(np.ceil(data_range / bin_width))
    return num_bins
  else:
    return 10
```

Over the years, the data consistently show significant differences in course selection and performance outcomes based on sex, particularly evident in the very low p-values (indicating strong statistical significance) in multiple years. For example, in 2019 and 2024, the chi-squared tests between '`Course` and `Sex`' showed extremely significant differences,

suggesting that sex might play a crucial role in how courses are chosen or the performance within those courses. These results underline the need for gender-sensitive approaches in educational settings to ensure equitable learning environments.

The variable `freq` often showed significant interactions with other performance metrics like `Total_AP`, `Total_PP`, and `Total_SPA`, especially in the later years such as 2022 and 2023. The extremely low p-values in tests like the chi-squared test between '`freq` and `Total_PP`' in 2022 suggest that frequency (possibly of class attendance or engagement) has a profound impact on performance. This insight could be used to reinforce the importance of consistent engagement in courses to optimize student outcomes.

Across the seven years, there appears to be a pattern of significant associations between class-related variables and performance outcomes, particularly in terms of assessment points (APs, PPs, SPAs). The data from 2023, showing a significant chi-squared statistic between 'Class and `Total_AP`', highlights this trend. Such findings could guide educational strategists to tailor class structures and teaching methods that better align with improving or modifying assessment strategies, ultimately enhancing educational effectiveness based on empirical evidence.

B

Feature Selection, Clustering and Prediction Analysis

B.1 Feature Selection and Clustering Notebook

The first three research questions were studied using the code developed in the `data_understanding` Jupyter Notebook, with improvements made in the Data Analysis Report notebook. However, the `Feature_selection_and_clustering` Notebook was specifically designed to address the fourth research question and to select features and create datasets for subsequent prediction and categorization tasks. This notebook, which is described in this section and executes the steps presented in Table B.1, employs a combination of clustering techniques and predictive analysis to uncover significant patterns in student performance and prepare the data for further analysis.

Table B.1: Steps in Feature Selection and Clustering notebook.

Step	Description
Importing Libraries	Necessary libraries are imported for data manipulation, visualization, machine learning, and clustering.
Auxiliary Methods	Functions are defined for loading data, updating column names, checking missing values, feature selection, and filtering datasets.
Loading Data	Datasets and other files are loaded into the notebook, including feature lists, dataset parameters, and additional datasets.
Updating Column Names	Column names are updated based on activity dictionaries, ensuring consistency across datasets.

Continued on next page

Table B.1: (continued)

Step	Description
Data Combination	Datasets from different years are combined into a single DataFrame, with columns standardized and duplicates removed.
Checking Missing Values	Missing values across datasets are checked and reported. Filtering is applied to remove rows with missing or zero values in key columns.
Feature Selection	Various methods such as Variance Threshold, Mutual Information, Random Forest, and Decision Trees are applied to select important features for predicting target variables.
Clustering Analysis	Clustering methods like K-Means and Hierarchical Clustering are applied to the datasets, with optional PCA for dimensionality reduction. The Elbow Method is used to determine the optimal number of clusters.
Saving Processed Data	Processed datasets are saved in various formats, and the results of feature selection and clustering are documented in a Word document.
Final Outputs	The final results, including charts and processed datasets, are saved in ZIP files for download.

In this notebook, several libraries are used for different purposes, such as pandas, that is used for data manipulation and analysis, while numpy supports numerical operations on large multi-dimensional arrays and matrices. For visualization, pyplot of matplotlib is employed to create static, animated, and interactive visualizations. Machine learning tasks, including preprocessing, feature selection, classification, and evaluation, are handled by sklearn modules, such as VarianceThreshold, mutual_info_classif, RandomForestClassifier, LabelEncoder, OneHotEncoder, SimpleImputer, KMeans, and PCA (that performs Principal Component Analysis - PCA). Additionally, cluster.hierarchy of scipy is used to enhance hierarchical clustering analysis.

The notebook integrates with Google Colaboratory using colab.files of google for file upload and download. Document generation is facilitated by the docx library, allowing the creation and updating of Microsoft Word files. Date and time utilities are provided by datetime, and pickle is used for data serialization. The os, StringIO, and shutil modules offer functionality for file operations and string manipulation. Finally, IPython.display is employed for rendering objects within the notebook, enhancing the presentation of data and results.

Auxiliary Methods

A number of auxiliary methods, including `print_to_doc`, `custom_display`, and `save_document`, were used in this notebook to handle the output. These methods were also used in `data_understanding` Jupyter notebook and the python file named `data_analysis_functions` imported in `Data Analysis Report` Jupyter notebook. The code for these methods is presented in Code Snippet B.1. The output is captured by the `print_to_doc` method and appended to a Microsoft Word document. `custom_display` uses IPython's display function based on a global configuration or saves display outputs to a Word document. The current date and time are appended to the Word document that is saved with a specified filename by the `save_document` function. If the integer `output_to_docx` defined on the notebook is set to 1, the Python method `print` will be replaced by `print_to_doc` and the Python `display` method will be replaced by `custom_display`.

Code Snippet B.1: Auxiliary methods used in several notebooks to save Word documents presenting outputs.

```
# Custom print function to capture output to a Word document
def print_to_doc(*args, **kwargs):
    text = ' '.join(map(str, args))
    doc.add_paragraph(text)

# Custom display function to capture output to a Word document
def custom_display(*args, **kwargs):
    if output_to_docx == 1:
        for arg in args:
            if isinstance(arg, pd.DataFrame):
                # Convert DataFrame to a string representation for saving
                buf = StringIO()
                arg.to_string(buf)
                text = buf.getvalue()
            else:
                # Convert other types of arguments to string
                text = str(arg)
            doc.add_paragraph(text)
    else:
        # Display using IPython's display function
        ipy_display(*args, **kwargs)

# Function to save the document with the current date
def save_document(filename='feature_selection.docx'):
    current_date = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    doc.add_paragraph(f"Document saved on {current_date}")
    doc.save(filename)
    print(f"Document saved as {filename}")
```

A directory `inputs` can be used to load Feather files using the method shown in Code Snippet B.2 named `load_compose_feather_files`. It returns a dictionary of datasets indexed by the extracted numeric identification after filtering out files whose names are not numeric and making sure there are no duplicates with already-existing datasets.

Code Snippet B.2: Method used in `Feature_selection_and_clustering` Jupyter Notebook to load files.

```
def load_compose_feather_files(datasets, path='inputs'):
    # Get a list of Feather files in the directory with only numbers in the filename
    files = [f for f in os.listdir(path) if f.endswith('.feather') and f[:-8].isdigit()
        ↪ and not f.startswith('dataset_')]
    compose_datasets = {}

    # Load the compose datasets
    for file in files:
        # Extract the index from the file name
        index = int(file.replace('.feather', ''))
        # Check if the index is already in datasets to avoid duplication
        if index not in datasets:
            # Read the Feather file into a DataFrame
            compose_datasets[index] = pd.read_feather(os.path.join(path, file))

    return compose_datasets
```

The `update_column_names` and `update_all_lists` functions are used to update dataset column names based on an activity dictionary. `update_column_names` applies the mapping to the dataset columns, while `update_all_lists` reflects these updated column names in various lists within the `all_lists` dictionary, ensuring consistency across different lists and years. These methods were also used on `data_preparation` Jupyter notebook as shown in Table A.4. The code is shown in Code Snippet B.3.

Code Snippet B.3: Method used to preform operations when merging datasets from different years in `Feature_selection_and_clustering` and `data_preparation` Jupyter Notebook files.

```
# Function to update column names based on a mapping
def update_column_names(dataset, activity_dict, sync_columns):
    column_mapping = {}
    for col in sync_columns:
        if col in activity_dict:
            activity_info = activity_dict[col]
            activity_type = activity_info[2]
            if activity_type in ['A', 'B', 'C', 'SPA']:
                column_mapping[col] = str(activity_info[0])
            else:
                column_mapping[col] = col # Keep the original column if not evaluated
        else:
```

```
            column_mapping[col] = col # Keep the original column if not in the dictionary

    # Apply the mapping to the columns
    new_columns = [column_mapping.get(col, col) for col in dataset.columns]
    dataset.columns = new_columns
    return dataset, column_mapping

# Function to update all relevant lists with the new column names
def update_all_lists(all_lists, year, column_mapping):
    updated_lists = all_lists[str(year)].copy()

    # Update basic lists
    for list_name in updated_lists['basic_lists']:
        updated_lists['basic_lists'][list_name] = [column_mapping.get(col, col) for col in
            ↪ updated_lists['basic_lists'][list_name]]

    # Update weekly and participation lists
    for list_name in updated_lists['weekly_and_participation_lists']:
        if isinstance(updated_lists['weekly_and_participation_lists'][list_name], dict):
            updated_lists['weekly_and_participation_lists'][list_name] = {
                k: [column_mapping.get(col, col) for col in v]
                for k, v in updated_lists['weekly_and_participation_lists'][list_name].items
                    ↪ ()
            }
        else:
            updated_lists['weekly_and_participation_lists'][list_name] = [
                column_mapping.get(col, col) for col in updated_lists['
                    ↪ weekly_and_participation_lists'][list_name]
            ]

    # Update activity type lists
    for list_name in updated_lists['activity_type_lists']:
        updated_lists['activity_type_lists'][list_name] = [
            column_mapping.get(col, col) for col in updated_lists['activity_type_lists'][
                ↪ list_name]
        ]

    # Update created columns lists
    for list_name in updated_lists['created_columns']:
        updated_lists['created_columns'][list_name] = [
            column_mapping.get(col, col) for col in updated_lists['created_columns'][
                ↪ list_name]
        ]

    return updated_lists
```

Some auxiliary methods were developed to present the characteristics of activities to aid in interpreting the feature selection results. The `print_activity_details_for_year`, `print_activity_details_for_year_by_id`, and other two methods, `get_activity_index`

and `get_activity_name`, provide utilities for accessing and displaying activity details from the dataset parameters, as shown in Code Snippet B.4. These methods facilitate the retrieval of activity indices or names by their keys or IDs, and they can print detailed information about specific activities for a given year.

Code Snippet B.4: Methods used in `Feature_selection_and_clustering` Jupyter Notebook file to access and display activity details.

```
# Function to update column names based on a mapping
def update_column_names(dataset, activity_dict, sync_columns):
    column_mapping = {}
    for col in sync_columns:
        if col in activity_dict:
            activity_info = activity_dict[col]
            activity_type = activity_info[2]
            if activity_type in ['A', 'B', 'C', 'SPA']:
                column_mapping[col] = str(activity_info[0])
            else:
                column_mapping[col] = col # Keep the original column if not evaluated
        else:
            column_mapping[col] = col # Keep the original column if not in the dictionary

    # Apply the mapping to the columns
    new_columns = [column_mapping.get(col, col) for col in dataset.columns]
    dataset.columns = new_columns
    return dataset, column_mapping

# Function to update all relevant lists with the new column names
def update_all_lists(all_lists, year, column_mapping):
    updated_lists = all_lists[str(year)].copy()

    # Update basic lists
    for list_name in updated_lists['basic_lists']:
        updated_lists['basic_lists'][list_name] = [column_mapping.get(col, col) for col in
            ↪ updated_lists['basic_lists'][list_name]]

    # Update weekly and participation lists
    for list_name in updated_lists['weekly_and_participation_lists']:
        if isinstance(updated_lists['weekly_and_participation_lists'][list_name], dict):
            updated_lists['weekly_and_participation_lists'][list_name] = {
                k: [column_mapping.get(col, col) for col in v]
                for k, v in updated_lists['weekly_and_participation_lists'][list_name].items
                    ↪ ()
            }
        else:
            updated_lists['weekly_and_participation_lists'][list_name] = [
                column_mapping.get(col, col) for col in updated_lists['
                    ↪ weekly_and_participation_lists'][list_name]
            ]

```

```
    # Update activity type lists
    for list_name in updated_lists['activity_type_lists']:
        updated_lists['activity_type_lists'][list_name] = [
            column_mapping.get(col, col) for col in updated_lists['activity_type_lists'][
                ↪ list_name]
        ]

    # Update created columns lists
    for list_name in updated_lists['created_columns']:
        updated_lists['created_columns'][list_name] = [
            column_mapping.get(col, col) for col in updated_lists['created_columns'][
                ↪ list_name]
        ]

    return updated_lists
```

The `check_missing_values_per_column` method in the `data_preparation` Jupyter Notebook has practically the same code as `check_missing_values` in the Jupyter Notebook file explained in this section, `Feature_selection_and_clustering`. It checks and prints missing values in each dataset, helping to identify columns with incomplete data and the code is presented in Code Snippet B.5.

Code Snippet B.5: Method used in `Feature_selection_and_clustering` Jupyter Notebook file to present the number of missing values for each column with missing data on a dataset.

```
def check_missing_values(datasets):
    for year, dataset in datasets.items():
        print(f"Missing values for {year} dataset:")
        missing_values = dataset.isnull().sum()
        # Filter and print only columns with missing values
        print(missing_values[missing_values > 0])
```

Filtering methods include `filter_missing_and_zero_final_grade` shown in Code Snippet B.6 and the `filter_dataset_by_activities` method whose code is depicted in Code Snippet B.7. `filter_missing_and_zero_final_grade` removes rows with missing or zero values in the `FinalGrade` column, whereas `filter_dataset_by_activities` filters the dataset to retain only specified target columns, additional columns, and those related to known activities.

Code Snippet B.6: Method used in `Feature_selection_and_clustering` Jupyter Notebook file to select only the students that passed CTCT discipline.

```
def filter_missing_and_zero_final_grade(datasets):

    filtered_datasets = {}
    for year, dataset in datasets.items():
```

```
        # Drop rows with NaN in 'FinalGrade' and where 'FinalGrade' is 0
        filtered_dataset = dataset.dropna(subset=['FinalGrade'])
        filtered_dataset = filtered_dataset[filtered_dataset['FinalGrade'] != 0]
        filtered_datasets[year] = filtered_dataset
        # Print the results of the filtering process
        print(f"After filtering, {year} dataset has {len(filtered_dataset)} rows. Before
            ↪ had {len(dataset)}. Dropped {len(dataset) - len(filtered_dataset)}
            ↪ instances/students.")
    return filtered_datasets
```

Code Snippet B.7: Method used in `Feature_selection_and_clustering` Jupyter Notebook file to select only the students that passed CTCT discipline.

```
def filter_dataset_by_activities(dataset, known_activities, target_column='
    ↪ FinalGradeInteger', additional_columns=None):
    if additional_columns is None:
        additional_columns = []

    # Extract activity IDs from the known_activities list
    activity_ids = [activity_id for activity_id, _ in known_activities]

    # Create a list of columns to retain
    columns_to_retain = [target_column] + additional_columns + [col for col in dataset.
        ↪ columns if col in activity_ids]

    # Filter the dataset
    filtered_dataset = dataset[columns_to_retain]

    return filtered_dataset
```

The `filter_known_activities` function refines an activity dictionary by excluding activities marked as 'Unknown,' returning a list of activities with valid descriptions, as shown in Code Snippet B.8.

Code Snippet B.8: Method used in `Feature_selection_and_clustering` Jupyter Notebook file to filter known activites.

```
def filter_known_activities(activity_dict):
    known_activities = []
    for activities in activity_dict.values():
        for activity_id, activity_desc in activities:
            if activity_desc != 'Unknown':
                known_activities.append((activity_id, activity_desc))
    return known_activities
```

As demonstrated in Code Snippet B.9, `convert_feature_ids_to_activity_names` and `convert_feature_ids_to_activity_names_complete` methods convert feature IDs into activity names using dataset parameters. These techniques work with single years as well as many years, translating specific traits for each technique into more comprehensible labels for activities.

Code Snippet B.9: Methods used in `Feature_selection_and_clustering` Jupyter Notebook file to convert attributes list resulting from feature selection to understandable names.

```
def convert_feature_ids_to_activity_names(selected_features, dataset_params):
    def get_activity_name_by_id(activity_id, dataset_params, year):
        """
        Helper function to get the activity name given its ID, year, and dataset parameters.
        """
        for value in dataset_params[str(year)]['activity_dictionary'].values():
            if str(value[0]) == str(activity_id):
                return value[1]
        return None

    if isinstance(selected_features, dict):
        converted_features = {}
        for year, methods in selected_features.items():
            if str(year) not in dataset_params:
                print(f"Year {year} not found in dataset_params.")
                continue

            converted_features[year] = {}

            for method, features in methods.items():
                converted_names = []
                for feature in features:
                    if feature.isdigit():
                        activity_name = get_activity_name_by_id(int(feature), dataset_params,
                            int(year))
                        if activity_name:
                            converted_names.append(activity_name)
                        else:
                            converted_names.append(f"ID_{feature}_Not_Found")
                    else:
                        converted_names.append(feature)
                converted_features[year][method] = converted_names

        return converted_features

    elif isinstance(selected_features, pd.Index):
        converted_names = []
        year = list(dataset_params.keys())[0] # Assumes first year if multiple years exist
        for feature in selected_features:
            if feature.isdigit():
                activity_name = get_activity_name_by_id(int(feature), dataset_params, int(
                    year))
                if activity_name:
                    converted_names.append(activity_name)
                else:
                    converted_names.append(f"ID_{feature}_Not_Found")
```

```
            else:
                converted_names.append(feature)
        return converted_names

def convert_feature_ids_to_activity_names_complete(selected_features, dataset_params):
    activity_dictionaries = {year: params['activity_dictionary'] for year, params in
        ↪ dataset_params.items()}

    converted_features = {}

    for method, features in selected_features.items():
        converted_features[method] = []
        for feature in features:
            found = False
            for activities in activity_dictionaries.values():
                for value in activities.values():
                    if str(value[0]) == feature:
                        converted_features[method].append((feature, value[1])) # Append
                            ↪ feature ID and name
                        found = True
                        break
                if found:
                    break
            if not found:
                converted_features[method].append((feature, 'Unknown')) # Append feature ID
                    ↪ and 'Unknown' if not found

    return converted_features
```

Feature Selection

Exclusion of Columns from Feature Selection

According to the objectives of the analysis, some columns were left out of a particular study in order to guarantee that the feature selection procedure concentrates on the most pertinent data. Three sets of exclusion criteria were used, however the student's enrolled class, commuting information, special statuses (SS), and class frequencies (attendance) were all eliminated in all of the category columns. All listed columns are included in the first exclusion set, named `exclude_columns`, with the exception of the activity results for the first two weeks of each CTCT edition. With fewer exclusions, the second set, designated `exclude_columns_less_restricted`, includes all activity results; however, it excludes the created columns, participation points, and SPA related columns, which were also omitted from the first set. With just the excluded categorical columns remaining, the final set, `exclude_columns_not_restricted`, is the least restrictive. We were able to adjust the feature selection procedure to various degrees of data granularity by utilizing these various sets of exclusions, guaranteeing a balance between comprehensiveness and relevancy.

Feature Selection Methods

In this dissertation, several feature selection methods were employed to identify the most informative attributes that influence student performance outcomes, such as Variance Threshold, Decision Tree (DT), Random Forest (RF), and Mutual Information (MI) classifiers. Every approach has advantages and disadvantages, and each is selected according to how well it can handle particular features of educational data, as stated in this dissertation Chapter 3 in Table 3.2. The rationale behind using these methods lies in their ability to effectively reduce the number of features, thereby simplifying the models and reducing the risk of overfitting. The selected features from each method were compared and consolidated to identify the most relevant features across all methods. This comprehensive approach ensures that the final model utilizes only the most predictive features, leading to more reliable and interpretable results.

The Mutual Information (MI) approach assesses the dependency between two variables and is especially helpful for capturing nonlinear interactions, whereas the variance threshold method is used to remove features whose variance falls below a predefined threshold. The foundation of Variance Threshold is the idea that features with low variance have little effect on the model's ability to forecast the future (or classify some attribute using another). Higher MI values indicate more relevant features and this method was used to evaluate the amount of information each feature shares with the target variable, which in this example includes several student performance metrics.

The variance threshold method presented in Code Snippet B.10 was applied across multiple datasets corresponding to different academic years and excluding the features in `exclude_columns_less_restricted` referred in Subsubsection B.1. This method selects features by filtering out those with low variance, which are less likely to contribute meaningful information. After applying a variance threshold (that was set at 0.05), the method ranks the remaining features by their variance, ensuring that features with the highest variance, which might be more important for the model, are prioritized. The retained features were then converted from feature IDs to their corresponding activity names to facilitate interpretation.

Code Snippet B.10: Feature selection method that uses variance threshold to present the list of columns with variance in descending order.

```
def select_by_variance(dataset, exclude_columns, variance_threshold=0.0):
    # Select numerical columns, drop excluded columns, and fill missing values with 0
    numerical_cols = dataset.select_dtypes(include=[np.number]).drop(columns=
        ↪ exclude_columns, errors='ignore').fillna(0)

    # Initialize the VarianceThreshold with the specified threshold
    sel = VarianceThreshold(threshold=variance_threshold)

    # Apply the variance threshold to the selected numerical columns
    numerical_cols_var = sel.fit_transform(numerical_cols)
```

```
10
11      # Retrieve the variances of the selected columns
12      variances = sel.variances_
13
14      # Sort the indices of the variances in descending order
15      sorted_indices = variances.argsort()[::-1]
16
17      # Get the column names sorted by their variance
18      sorted_columns = numerical_cols.columns[sorted_indices]
19
20      return sorted_columns
```

A number of features in the 2016 sample showed considerable variance, indicating variations in student performance across different tasks. Notable among these were "Evaluation of synthesis groups A vs. A," "Individual Test: VB Function," "School Planning," "SMART goals," and "Moodle test Excel." The 2017 dataset revealed similar patterns, with significant variance observed in activities like "SMART goals," "School Planning," and "Individual Test: VB Function," along with additional tasks such as "Newscast Presentation" and "Leadership Paradigms (Spokesperson)." This pattern persisted in 2018, where features like "Evaluation of synthesis groups A vs. A," "Individual Test: VB Function," and "SMART goals" continued to display variability. By 2019, the activities "List of Insured Persons," "Moodle test Excel," and "Presentation of Oral Communication" also showed significant variance.

In the 2022 dataset, new features such as "Purchase of computer equipment (Solver)" and "Communication via audiovisual media - Slides" emerged with high variance. However, many feature IDs were not properly mapped to their corresponding activity names, as indicated by entries labeled `ID_Not_Found`, suggesting a conversion issue that needs addressing. The datasets for 2023 and 2024 exhibited a similar trend, retaining high-variance features like "SMART goals" and "Communication using audiovisual media - Slides." Despite this consistency, the conversion problem persisted, leaving several features unidentified in the final analysis.

The Random Forest (RF) classifier was employed due to its feature significance mechanism, which yields a score that represents the contribution of each feature to enhancing the model's predictive accuracy. This approach is advantageous due to its ability to manage huge data sets with a high degree of complexity and its resilience against overfitting. Decision Trees (DT), like RF, also offer a feature importance score. They are appropriate for preliminary feature relevance investigations, nevertheless, because they are easier to compute and take less time.

Multiple steps were involved in implementing attribute selection for predicting a specific feature, as demonstrated by Code Snippet B.11, which displays the code of the method called `feature_selection_single_dataframe`. This method takes five inputs: a data frame, a dictionary with different lists of the dataset's columns, a list of columns to

omit from the analysis, and the methods to use from the ones mentioned. It then returns a dictionary with the features selected for each method, arranged in descending order in terms of preference using a specific method.

Code Snippet B.11: Feature selection method to apply to a single dataframe.

```
def feature_selection_single_dataframe(df, all_lists, target_column, exclude_columns,
    ↪ methods=['MI', 'RF', 'DT']):
    def select_by_mutual_info(numerical_cols, target_encoded, random_state=2024):
        mi = mutual_info_classif(numerical_cols, target_encoded, random_state=random_state)
        mi_scores = pd.Series(mi, index=numerical_cols.columns)
        return mi_scores

    def select_by_random_forest(numerical_cols, target_encoded, random_state=2024):
        rf = RandomForestClassifier(random_state=random_state)
        rf.fit(numerical_cols, target_encoded)
        rf_scores = pd.Series(rf.feature_importances_, index=numerical_cols.columns)
        return rf_scores

    def select_by_decision_tree(numerical_cols, target_encoded, random_state=2024):
        dt = DecisionTreeClassifier(random_state=random_state)
        dt.fit(numerical_cols, target_encoded)
        dt_scores = pd.Series(dt.feature_importances_, index=numerical_cols.columns)
        return dt_scores

    print(f"Feature Selection Analysis for the dataset with target column: {target_column}
        ↪ ")

    if target_column not in df.columns:
        print(f"Skipping: Target column not found in dataset.")
        return {}

    # Ensure columns are excluded correctly
    columns_to_exclude = set(exclude_columns) | set([target_column])
    numerical_cols = df.select_dtypes(include=[np.number]).drop(columns=columns_to_exclude,
        ↪  errors='ignore')

    # Handle missing values
    imputer = SimpleImputer(strategy='constant', fill_value=0)
    numerical_cols = pd.DataFrame(imputer.fit_transform(numerical_cols), columns=
        ↪ numerical_cols.columns)

    selected_features = {}

    # Identify categorical columns
    categorical_cols = [col for col in df.columns if col in all_lists['basic_lists']['
        ↪ listCategorical']]
    categorical_cols = [col for col in categorical_cols if col not in columns_to_exclude]
        ↪ # Exclude categorical columns that should be excluded

```

```
    # Create a copy of the dataset to avoid modifying the original
    df_copy = df.copy()

    # Apply appropriate encoding for each method
    encoded_datasets = {}

    # OneHotEncoding for MI
    if 'MI' in methods and categorical_cols:
        enc = OneHotEncoder(sparse_output=False, drop='first')
        encoded_categorical = enc.fit_transform(df[categorical_cols].astype(str))
        encoded_categorical_df = pd.DataFrame(encoded_categorical, columns=enc.
            ↪ get_feature_names_out(categorical_cols))
        encoded_datasets['MI'] = numerical_cols.join(encoded_categorical_df)

    # LabelEncoding for RF and DT
    if 'RF' in methods or 'DT' in methods:
        le = LabelEncoder()
        for col in categorical_cols:
            df_copy[col] = le.fit_transform(df_copy[col].astype(str))

        if 'RF' in methods:
            encoded_datasets['RF'] = df_copy[numerical_cols.columns.tolist() +
                ↪ categorical_cols]

        if 'DT' in methods:
            encoded_datasets['DT'] = df_copy[numerical_cols.columns.tolist() +
                ↪ categorical_cols]

    # Handle target column discretization
    if pd.api.types.is_numeric_dtype(df[target_column]):
        # Discretize continuous target columns
        discretizer = KBinsDiscretizer(n_bins=4, encode='ordinal', strategy='uniform')
        target_encoded = discretizer.fit_transform(df[[target_column]]).flatten()
    else:
        target_encoded = df[target_column]

    for method in methods:
        if method == 'MI' and 'MI' in encoded_datasets:
            mi_scores = select_by_mutual_info(encoded_datasets['MI'], target_encoded)
            print(f"Mutual Information (MI) Score Statistics: {mi_scores.describe()}")
            important_features_mi = mi_scores.sort_values(ascending=False).index[mi_scores >
                ↪  mi_scores.median()]
            selected_features['MI'] = important_features_mi.tolist()
            print("Important features by MI:", important_features_mi)

        if method == 'RF' and 'RF' in encoded_datasets:
            rf_scores = select_by_random_forest(encoded_datasets['RF'], target_encoded)
            print(f"Random Forest (RF) Feature Importance Statistics: {rf_scores.describe()}
                ↪ ")
```

```
83            important_features_rf = rf_scores.sort_values(ascending=False).index[rf_scores >
                  ↪ rf_scores.mean()]
84            selected_features['RF'] = important_features_rf.tolist()
85            print("Important features by RF:", important_features_rf)
86
87        if method == 'DT' and 'DT' in encoded_datasets:
88            dt_scores = select_by_decision_tree(encoded_datasets['DT'], target_encoded)
89            print(f"Decision Tree (DT) Feature Importance Statistics: {dt_scores.describe()}
                  ↪ ")
90            important_features_dt = dt_scores.sort_values(ascending=False).index[dt_scores >
                  ↪ dt_scores.max() * 0.1]
91            selected_features['DT'] = important_features_dt.tolist()
92            print("Important features by DT:", important_features_dt)
93
94    print("-" * 50)
95
96    return selected_features
```

As shown in Code Snippet B.11, the first step of the feature selection for prediction was data preparation, where all datasets were processed to handle missing values and encode categorical variables appropriately. This involved appending zeros to non-numeric missing values and using either OneHotEncoding or LabelEncoding for categorical features, depending on the method used. The numerical features were then standardized and put through the selection process with MI method during the feature selection execution. In contrast, for the RF and DT methods, all preprocessed features were used to train the models and extract feature importance based on their contribution to model accuracy.

Lastly, the features chosen by each method were assessed according to their scores. In the MI method, features that scored higher than the median were deemed significant; in the RF method, features that contributed more than the average to the model's performance were chosen; and in the DT method, columns whose scores are higher than 10% of the maximum value of the obtained scores for this algorithm. These values were chosen in an attempt to approximate the same value of the chosen features by observing the values of the produced score. Furthermore, in cases where the target variable was continuous, it was discretized to facilitate the effective application of feature selection techniques.

The feature selection analysis for predicting final grades (`FinalGradeInteger`) using the `dataset_complete` data frame, and shown in Code Snippet B.11, identified several key features through different selection methods. Mutual Information (MI) selected a wide range of features including both academic performance metrics such as `Total_AP`, `Total_SPA`, and `Total_PP`, and specific activities such as SMART goals, Submission of CV on Moodle, Maria's dashboard, Parking, Wolis, 4x100m freestyle, Ethical dilemmas, Forced landing - individual, Analysis of 3 CVs of candidates, Assessment of the week, Leading is ..., and Psychotechnical test - Moodle. This suggests that a combination of different academic metrics and specific soft skills activities are relevant in predicting the final grades. Random Forest (RF) also identified several of the same features, with a focus

on activities such as Maria's dashboard, 4x100m freestyle, Analysis of 3 CVs of candidates, Forced landing - individual, and Leading is ..., along with features like participation points for theoretical classes (`PTP3`, `PTP1`) and the bonification column (`Bon`). The Decision Tree (DT) method highlighted a smaller, more specific set of features, such as 4x100m freestyle, Psychotechnical test - Moodle, and Sex, as well as participation and bonification points related columns such as `PP1_3`, `Bon`, and `Total_Bon`.

In addition to the code for a single dataset that is provided in Code Snippet B.11, two other comparable methods were developed and are displayed in Code Snippets B.12 and B.13, respectively, for the datasets for each of the years. The three code snippets use the same techniques to assess and prioritize features, and they all follow the same basic rationale for feature selection. Their application contexts vary, though, as each focuses on a different component of the datasets in order to find the most pertinent features.

The first presented method (`feature_selection_single_dataframe`) handles a single dataset at a time, and was created as the last to process the combined dataset with no missing data. This method systematically processes the data by excluding irrelevant columns, encoding categorical variables, and applying the selection methods to determine the most important features for predicting outcomes like passing status or final grades.

The second method (`feature_selection_analysis`) is shown in Code Snippet B.12 and extends this approach by applying the feature selection methods across multiple datasets, each representing a different academic year. This method allows for year-specific analysis, accounting for the unique characteristics and data distributions of each year. This method was employed to predict various outcomes, such as passing status (`Passed`), final grades (`FinalGrade`), total activity points (`Total_AP`), total participation points (`Total_PP`), and self and peer assessment points (`Total_SPA`). For each target variable, the most relevant features were identified using the MI, RF, and DT techniques, and were subsequently converted from feature IDs to meaningful activity names. This conversion facilitated a more intuitive interpretation of the results, making the findings actionable and easier to comprehend.

For most predicted columns, the analysis included all students, capturing the full range of student performance and participation. However, for the `FinalGrade` column, only students who passed the CTCT discipline with a grade of 10 or higher were considered in the analysis. This selective filtering was necessary to ensure that the feature selection process accurately reflected the characteristics and activities associated with successful course completion. Without this filtering, the analysis might have included data from students who did not pass, potentially skewing the results and reducing the predictive power of the model for this specific outcome. Therefore, while other predicted columns utilized data from all students, the analysis of `FinalGrade` focused exclusively on those who met the passing criteria.

Code Snippet B.12: Feature selection method to apply to the original preprocessed datasets (one for each year).

```
def feature_selection_analysis(datasets, all_lists, dataset_params, target_column, exclude_columns, methods=['MI', 'RF', 'DT']):
    # Same code as in feature_selection_single_dataframe omitted
    for year, dataset in datasets.items(): # code inside a FOR that runs original datasets
        print(f"Feature Selection Analysis for the dataset of year: {year} with target column: {target_column}")

        if target_column not in dataset.columns:
            print(f"Skipping year {year}: Target column not found in dataset.")
            continue

        # Ensure columns are excluded correctly
        columns_to_exclude = list(exclude_columns[str(year)]) + [target_column]
        numerical_cols = dataset.select_dtypes(include=[np.number]).drop(columns=columns_to_exclude, errors='ignore')

        non_numeric_columns = check_non_numeric_values(dataset, columns_to_exclude)

        if non_numeric_columns:
            print(f"Columns with non-numeric values (including NaN): {non_numeric_columns}")
            # Impute missing values
            imputer = SimpleImputer(strategy='constant',fill_value=0)
            numerical_cols = pd.DataFrame(imputer.fit_transform(numerical_cols), columns=numerical_cols.columns)
        else:
            print("All selected columns are numeric.")
        # Impute missing values in the target column if necessary
        if dataset[target_column].isnull().any():
            print(f"Imputing missing values in target column: {target_column}")
            target_imputer = SimpleImputer(strategy='constant',fill_value=0)
            dataset[target_column] = target_imputer.fit_transform(dataset[[target_column]]).flatten()
        # Identify categorical columns
        categorical_cols = [col for col in dataset.columns if col in all_lists[str(year)]['basic_lists']['listCategorical']]
        categorical_cols = [col for col in categorical_cols if col not in columns_to_exclude] # Exclude categorical columns that should be excluded
        # Create a copy of the dataset to avoid modifying the original
        dataset_copy = dataset.copy()
        # Apply appropriate encoding for each method
        encoded_datasets = {}
        # OneHotEncoding for MI
        if 'MI' in methods and categorical_cols:
            # Similar to method feature_selection_single_dataframe
        # LabelEncoding for RF and DT
        if 'RF' in methods or 'DT' in methods:
            # Similar to method feature_selection_single_dataframe
        # Handle target column discretization
```

```
        if pd.api.types.is_numeric_dtype(dataset[target_column]):
            # Similar to method feature_selection_single_dataframe
        selected_features[year] = {}
        for method in methods:
            # Similar to method feature_selection_single_dataframe
        print("-" * 50)

    return selected_features
```

The approach is further generalized to handle combined datasets spanning multiple years by the third method (`feature_selection_analysis_generic`), which is illustrated in Code Snippet B.13. This approach is especially helpful for detecting consistent and important features across time since it takes into consideration the different data quality and organization between years. The method used for a `dataset_complete` data frame only considered activities from 2024 to its execution (although it has data from all the available years), whereas this method checks which activity lists are for each year whose data frame dictionary entry is `combined_dataset` and checks which years are present in each of the combined datasets in combination.

Code Snippet B.13: Feature selection method to apply to the combined datasets with several years with different number of missing values.

```
def feature_selection_analysis_generic(datasets, all_lists, target_column, exclude_columns
    ↪ , methods=['MI', 'RF', 'DT']):
    # Same code as in feature_selection_single_dataframe omitted

    for key, dataset in datasets.items(): # for each preprocessed original dataset
        print(f"Feature Selection Analysis for the dataset with key: {key} with target
            ↪ column: {target_column}")
        if target_column not in dataset.columns:
            print(f"Skipping key {key}: Target column not found in dataset.")
            continue
        years = dataset['Year'].unique()
        # Ensure columns are excluded correctly
        columns_to_exclude = set(exclude_columns.get(str(years[0]), [])) | set([
            ↪ target_column])
        numerical_cols = dataset.select_dtypes(include=[np.number]).drop(columns=
            ↪ columns_to_exclude, errors='ignore')
        # Handle missing values
        imputer = SimpleImputer(strategy='constant',fill_value=0)
        numerical_cols = pd.DataFrame(imputer.fit_transform(numerical_cols), columns=
            ↪ numerical_cols.columns)
        selected_features[key] = {}

        # runs each of the years that a combined dataset as data about
        for year in years:
            year = str(year)
            # Identify categorical columns for the specific year
```

```
            if year in all_lists:
                categorical_cols = [col for col in dataset.columns if col in all_lists[year
                    ↪ ]['basic_lists']['listCategorical']]
            else:
                print(f"Year {year} not found in all_lists for key {key}. Skipping this year.
                    ↪ ")
                continue
            categorical_cols = [col for col in categorical_cols if col not in
                ↪ columns_to_exclude] # Exclude categorical columns that should be
                ↪ excluded
            # Create a copy of the dataset to avoid modifying the original
            dataset_copy = dataset.copy()
            # Apply appropriate encoding for each method
            encoded_datasets = {}
            # OneHotEncoding for MI
            if 'MI' in methods and categorical_cols:
                # Similar to method feature_selection_single_dataframe
            # LabelEncoding for RF and DT
            if 'RF' in methods or 'DT' in methods:
                # Similar to method feature_selection_single_dataframe
            # Handle target column discretization
            if pd.api.types.is_numeric_dtype(dataset[target_column]):
                # Similar to method feature_selection_single_dataframe
            for method in methods:
               # Similar to method feature_selection_single_dataframe
        print("-" * 50)

    return selected_features
```

The code snippets presented demonstrate a systematic approach to feature selection, with each method adapted to handle different dataset configurations—whether single datasets, yearly datasets, or combined datasets. Although the creation of a unified method to handle all types of datasets was considered, it was not implemented due to time constraints within the project.

To preserve the results of the feature selection process, the selected features and corresponding datasets are saved in Feather format. The `create_and_save_datasets` method automates this process, generating datasets for each feature selection scenario and saving them with filenames that reflect the method and target column used. This systematic approach ensures that the results are well-documented and easily accessible for future reference.

Code Snippet B.14: Method used in `Feature_selection_and_clustering` Jupyter Notebook file that creates and saves datasets with selected features for each method and year.

```
def create_and_save_datasets(selected_features, datasets, target_column):
    for year, methods in selected_features.items():
        for method, features in methods.items():
            dataset = datasets.get(int(year))
```

```
        if the dataset is None:
            print(f"Dataset for year {year} not found.")
            continue

        # Ensure the target column exists in the dataset
        if target_column not in dataset.columns:
            print(f"Skipping dataset for year {year} and method {method}: Target column
                ↪ '{target_column}' not found in dataset.")
            continue

        # Filter features that exist in the dataset
        valid_features = [feature for feature in features if feature in dataset.columns
            ↪ ]
        valid_features.append(target_column) # Ensure target column is the last column

        filtered_dataset = dataset[valid_features]
        file_name = f'dataset_{year}_{method}_{target_column}.feather'
        filtered_dataset.reset_index(drop=True).to_feather(f'output/feather_files/{
            ↪ file_name}')
        print(f"Dataset saved as '{file_name}' with features: {valid_features}")
```

The feature selection analysis offered a strong framework for locating the most instructive aspects in the educational data, but it has a major drawback in that the findings may not match up because the features are not scaled before the selection. However, when utilizing the first two weeks' results to estimate ultimate outcomes, the prediction results will demonstrate that, despite this constraint, the selected attributes increased the results.

Clustering Analysis

To find distinct patterns in student performance and provide an answer to the fourth research question of whether distinct clusters are obvious in student performances based on various types of evaluation points, hierarchical clustering and K-means clustering (KMC) were used to analyze the data and this was achieved by creating two separate Google Colaboratory notebooks. K-means clustering (KMC) is often referenced in EDM literature reviews and typically models circular clusters, providing a contrast with Hierarchical Clustering, which captures different data aspects.

The initial notebook, an enhancement of the `data_understanding` notebook named `Data_Analysis_Report`, offers a comprehensive overview of clustering outcomes. This notebook focuses on various engineered features, including total points and total AP and PP per week, applying these clustering techniques across seven datasets.

Clustering Methods and Setup

The second notebook, `Feature_selection_and_clustering`, is tailored specifically for this research question and assists in feature selection for a prediction-focused query and it's the focus of this section of this Appendix. It compares the effects of using featured

characteristics with totals versus only the results from the first two weeks of activities. This notebook provides a detailed annual analysis and displays the clustering results for the complete dataset, encompassing all years. Additionally, it includes a comparison of employing two components in Principal Component Analysis (PCA) reduction, as shown in Snippet B.15, against using the selected clustering features directly without reduction. It was confirmed that both approaches yielded identical outcomes in terms of the Silhouette Score, Davies-Bouldin Index (DBI), and Calinski-Harabasz (CH) Index.

Code Snippet B.15: Display PCA components and their contribution to the Principal Components.

```
def display_pca_components(pca, feature_names):
    components_df = pd.DataFrame(pca.components_, columns=feature_names, index=[f'PC{i+1}'
        ↪ for i in range(pca.n_components_)])
    explained_variance = pca.explained_variance_ratio_
    print("PCA Components and their contribution to the Principal Components:")
    print(components_df)
    print("\nExplained variance ratio per Principal Component:")
    for i, variance in enumerate(explained_variance):
        print(f"PC{i+1}: {variance:.4f}")
```

Before applying Principal Component Analysis (PCA) for dimensionality reduction, the data was scaled to ensure that each feature contributed equally to the clustering process. The PCA contribution tables that display the loading scores for each week's activity points (`Week1_AP` to `Week5_AP`) on the first two principal components (PC1 and PC2) are shown in Table B.2. These loading scores indicate the extent to which each week's activity points contribute to the variance captured by each principal component. The explained variance ratio per principal component indicates the percentage of the total variance captured by each principal component. For instance, in 2016, PC1 captures 65.50% of the variance, while PC2 captures 11.11%. This indicates that PC1 is the most significant component in terms of capturing the variance in the data, with PC2 providing additional but lesser information.

Table B.2: PCA Components, their contribution, and explained variance ratio to the Principal Components for each year.

Year	Principal Component	Week1_AP	Week2_AP	Week3_AP	Week4_AP	Week5_AP	Explained Variance (%)
2016	**PC1**	-0.43	-0.47	-0.48	-0.42	-0.43	65.5
	PC2	0.53	-0.03	-0.01	-0.80	0.30	11.1
2017	**PC1**	-0.44	-0.47	-0.47	-0.44	-0.42	73.8
	PC2	0.30	0.11	0.14	0.31	-0.89	8.6
2018	**PC1**	-0.46	-0.49	-0.48	-0.39	-0.41	54.6
	PC2	0.15	-0.05	0.02	-0.77	0.61	14.7
2019	**PC1**	-0.44	-0.46	-0.48	-0.43	-0.43	64.5
	PC2	0.62	-0.09	0.06	-0.76	0.17	10.5
2022	**PC1**	-0.571	-0.58	-0.58	-	-	85.8
	PC2	0.81	-0.27	-0.53	-	-	7.9
2023	**PC1**	-0.48	-0.49	-0.52	-0.51	-	74.4
	PC2	-0.82	0.55	0.14	0.10	-	10.5
2024	**PC1**	-0.50	-0.51	-0.51	-0.48	-	78.4
	PC2	-0.43	-0.38	0.03	0.82	-	9.3

To determine the optimal number of clusters for the K-means clustering algorithm (KMC), the Elbow Method was used, as illustrated in Code Snippet B.16. This method involves plotting the Within-Cluster Sum of Squares (WCSS) for different numbers of clusters and identifying the "elbow point" where the rate of decrease slows. The `random_state` parameter was set to ensure reproducibility, with values of 0 used in `Feature_selection_and_clustering` and 42 in `Data_Analysis_Report`. Additionally, the number of times the K-means algorithm is run with different centroid seeds was set to 10 in both cases.

Code Snippet B.16: Elbow Method plotting method for determining the optimal number of clusters.

```
def plot_elbow_method(X, year, max_clusters=10, save_path='Charts/Elbow'):
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    distortions = []
    for i in range(1, max_clusters + 1):
        kmeans = KMeans(n_clusters=i, random_state=0, n_init=10)
        kmeans.fit(X)
        distortions.append(kmeans.inertia_)

    plt.figure()
    plt.plot(range(1, max_clusters + 1), distortions, marker='o')
    plt.title(f'Elbow Method for Optimal Number of Clusters ({year})')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Distortion')
    plt.savefig(f'{save_path}/Elbow_Method_{year}.png')
    plt.close()
```

For check the optimal number of clusters for Hierarchichal Clustering it was used dendrogram as the ones generated by Code Snippet B.17 that was used in `Data_Analysis_Report`, using 'ward' linkage method estimated cluster membership by calculating the total sum of squared deviations from the mean of a cluster. One of the limitations of this analysis is that it was not tryed different linkage methods, but the Hierarchichal Clustering was not a priority, only a based for comparing with KMC.

Code Snippet B.17: Dendrogram plotting method for determining the optimal number of clusters.

```
# Function to plot Hierarchical Clustering Dendrogram
def plot_dendrogram(X, year):
    """Plots a dendrogram for hierarchical clustering."""
    Z = linkage(X, method='ward')
    fig, ax = plt.subplots()
    dendrogram(Z, ax=ax)
    ax.set_title(f'Dendrogram ({year})')
    ax.set_xlabel('Students')
    ax.set_ylabel('Euclidean distances')

    image_name = f'Dendrogram_{year}.png'
    save_and_show_plot(fig, image_name)
```

Figure 4.10 presents the charts generated by Code Snippets B.16 and B.17 regarding the 2024 dataset for features from `Week1_AP` to `Week5_AP` and Figure 4.11 shows the generated clusters.

Code Snippet B.18 was used to perform KMC with scaling for the more extensive clustering analysis performed in the `Feature_selection_and_clustering` file. This method has an option to choose to reduce the dimensionality with PCA (`use_pca=True`), for a more flexible approach. It also always display the performance metrics (Silhouette, DBI and CH Index) and show the components contributions when the reduction option is chosen. Code Snippet B.19 presents a similar method but for Hiearachichal Clustering and also plotting the correspondent dendograms.

Code Snippet B.18: Method that performs K-Means clustering with optional PCA.

```
def perform_kmeans_clustering(df, features, year, num_clusters, use_pca=True, save_path='Charts/Clustering'):
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    # Data preparation
    X = df[features].values
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    if use_pca:
        pca = PCA(n_components=2)
        X_pca = pca.fit_transform(X_scaled)
        display_pca_components(pca, features)

    # K-Means Clustering
    kmeans = KMeans(n_clusters=num_clusters, init='k-means++', random_state=0, n_init=10)
    y_kmeans = kmeans.fit_predict(X_scaled)

    # Evaluate clustering
    silhouette_avg = silhouette_score(X_scaled, y_kmeans)
    davies_bouldin = davies_bouldin_score(X_scaled, y_kmeans)
    calinski_harabasz = calinski_harabasz_score(X_scaled, y_kmeans)

    print(f"Silhouette Score for {num_clusters} clusters: {silhouette_avg:.3f}")
    print(f"Davies-Bouldin Index for {num_clusters} clusters: {davies_bouldin:.3f}")
    print(f"Calinski-Harabasz Index for {num_clusters} clusters: {calinski_harabasz:.3f}")

    # Plotting the clusters with PCA-reduced data if PCA is used
    if use_pca:
        centroids = kmeans.cluster_centers_
        centroids_pca = pca.transform(centroids)
        plt.figure()
        colors = ['red', 'green', 'blue', 'purple', 'orange', 'brown'] # Extend this list if needed
        for i in range(num_clusters):
            plt.scatter(X_pca[y_kmeans == i, 0], X_pca[y_kmeans == i, 1], color=colors[i % len(colors)], label=f'Cluster {i+1}')
        plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1], s=100, c='yellow', label='Centroids', marker='x')
```

```
        plt.title(f'PCA-Reduced K-Means Clustering of {num_clusters} Clusters ({year})')
        plt.xlabel('Principal Component 1')
        plt.ylabel('Principal Component 2')
        plt.legend()
        plt.savefig(f'{save_path}/PCA_KMeans_{num_clusters}_Clusters_{year}.png')
        plt.close()
    elif X_scaled.shape[1] == 2: # If no PCA and only 2 features, plot directly
        plt.figure()
        colors = ['red', 'green', 'blue', 'purple', 'orange', 'brown']
        for i in range(num_clusters):
            plt.scatter(X_scaled[y_kmeans == i, 0], X_scaled[y_kmeans == i, 1], color=
                ↪ colors[i % len(colors)], label=f'Cluster {i+1}')
        plt.title(f'K-Means Clustering of {num_clusters} Clusters ({year})')
        plt.xlabel('Feature 1')
        plt.ylabel('Feature 2')
        plt.legend()
        plt.savefig(f'{save_path}/KMeans_{num_clusters}_Clusters_{year}.png')
        plt.close()
```

Code Snippet B.19: Method to perform Hierarchical Clustering with optional PCA.

```
def perform_hierarchical_clustering(df, features, year, num_clusters, use_pca=True,
    ↪ save_path='output/Clustering'):
    if not os.path.exists(save_path):
        os.makedirs(save_path)

    X = df[features].values
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)

    if use_pca:
        pca = PCA(n_components=2)
        X_pca = pca.fit_transform(X_scaled)
        display_pca_components(pca, features)

    # Hierarchical Clustering
    hc = AgglomerativeClustering(n_clusters=num_clusters, linkage='ward')
    y_hc = hc.fit_predict(X_scaled)

    # Evaluate clustering
    silhouette_avg = silhouette_score(X_scaled, y_hc)
    davies_bouldin = davies_bouldin_score(X_scaled, y_hc)
    calinski_harabasz = calinski_harabasz_score(X_scaled, y_hc)

    print(f"Silhouette Score for {num_clusters} clusters: {silhouette_avg:.3f}")
    print(f"Davies-Bouldin Index for {num_clusters} clusters: {davies_bouldin:.3f}")
    print(f"Calinski-Harabasz Index for {num_clusters} clusters: {calinski_harabasz:.3f}")

    # Plotting the clusters with PCA-reduced data if PCA is used
    if use_pca:
```

```
        plt.figure()
        colors = ['red', 'blue', 'green', 'purple', 'orange', 'brown'] # Extend this list
            ↪ if needed
        for i in range(num_clusters):
            plt.scatter(X_pca[y_hc == i, 0], X_pca[y_hc == i, 1], color=colors[i % len(
                ↪ colors)], label=f'Cluster {i+1}')
        plt.title(f'PCA-Reduced Hierarchical Clustering of Students ({year})')
        plt.xlabel('Principal Component 1')
        plt.ylabel('Principal Component 2')
        plt.legend()
        plt.savefig(f'{save_path}/PCA_Hierarchical_Clusters_{year}.png')
        plt.close()

        # Plotting the Dendrogram without labels on the x-axis
        plt.figure()
        dendrogram = sch.dendrogram(sch.linkage(X_scaled, method='ward'), no_labels=True)
        plt.title(f'Dendrogram for Hierarchical Clustering ({year})')
        plt.xlabel('Students')
        plt.ylabel('Euclidean distances')
        plt.savefig(f'{save_path}/Dendrogram_{year}.png')
        plt.close()

        # Plotting centroids for Hierarchical Clustering
        cluster_centers = np.array([X_scaled[y_hc == i].mean(axis=0) for i in range(
            ↪ num_clusters)])
        centroids_pca = pca.transform(cluster_centers)
        plt.figure()
        for i in range(num_clusters):
            plt.scatter(X_pca[y_hc == i, 0], X_pca[y_hc == i, 1], color=colors[i % len(
                ↪ colors)], label=f'Cluster {i+1}')
        plt.scatter(centroids_pca[:, 0], centroids_pca[:, 1], s=100, c='yellow', label='
            ↪ Cluster Centers', marker='x')
        plt.title(f'PCA-Reduced Hierarchical Clustering with Cluster Centers ({year})')
        plt.xlabel('Principal Component 1')
        plt.ylabel('Principal Component 2')
        plt.legend()
        plt.savefig(f'{save_path}/PCA_Hierarchical_Centers_{year}.png')
        plt.close()
    elif X_scaled.shape[1] == 2: # If no PCA and only 2 features, plot directly
        plt.figure()
        colors = ['red', 'blue', 'green', 'purple', 'orange', 'brown']
        for i in range(num_clusters):
            plt.scatter(X_scaled[y_hc == i, 0], X_scaled[y_hc == i, 1], color=colors[i %
                ↪ len(colors)], label=f'Cluster {i+1}')
        plt.title(f'Hierarchical Clustering of Students ({year})')
        plt.xlabel('Feature 1')
        plt.ylabel('Feature 2')
        plt.legend()
        plt.savefig(f'{save_path}/Hierarchical_Clusters_{year}.png')
        plt.close()
```

Results of Clustering Analysis

The evaluation metrics obtained using Code Snippets B.18 and B.19 for all the available datasets and the one produced by merging the different datasets (referred to as 'All' in this table) are presented in Table B.3. It is evident that the merged dataset with no missing data produced considerably inferior results since numerous evaluation-related factors changed over time.

Table B.3: Results of clustering analysis when using `Total_AP`, `Total_PP`, `Total_SPA` and `Total_Bon` as features.

Year	Method	Clusters	Silhouette Score	DBI	CH Index
2016	KMC	2	0.809	0.322	1542.388
		3	0.836	0.279	2665.552
		4	0.419	0.625	2750.951
	Hierarchical	2	0.819	0.141	1512.526
		3	0.836	0.213	2578.661
		4	0.441	0.625	2673.296
2017	KMC	2	0.793	0.295	1440.347
		3	0.830	0.267	2353.784
		4	0.774	0.284	2942.418
	Hierarchical	2	0.791	0.282	1413.282
		3	0.830	0.273	2338.163
		4	0.774	0.291	2912.722
2018	KMC	2	0.785	0.481	859.538
		3	0.810	0.412	1160.181
		4	0.758	0.433	1560.501
	Hierarchical	2	0.780	0.518	814.967
		3	0.808	0.441	1146.982
		4	0.759	0.480	1536.102
	KMC	2	0.786	0.411	937.800
		3	0.816	0.307	1323.234

Continued on next page

Table B.3: (continued)

Year	Method	Clusters	Silhouette Score	DBI	CH Index
	Hierarchical	4	0.764	0.298	1715.257
		2	0.778	0.405	903.825
		3	0.811	0.360	1305.294
		4	0.765	0.379	1685.897
2022	KMC	2	0.782	0.486	2527.851
		3	0.746	0.770	1968.432
		4	0.357	0.952	1853.781
	Hierarchical	2	0.770	0.606	2385.140
		3	0.746	0.770	1968.432
		4	0.736	0.504	1843.474
2023	KMC	2	0.782	0.431	1122.895
		3	0.809	0.362	1700.979
		4	0.327	0.740	1623.344
	Hierarchical	2	0.777	0.490	1102.802
		3	0.805	0.411	1660.461
		4	0.319	0.811	1463.431
2024	KMC	2	0.764	0.393	1289.933
		3	0.756	0.363	2036.112
		4	0.426	0.690	2010.855
	Hierarchical	2	0.754	0.506	1258.797
		3	0.756	0.431	1967.977
		4	0.752	0.793	1786.779
All	KMC	2	0.662	0.492	4546.664
		3	0.698	0.428	6280.002
		4	0.616	0.498	13399.484
		2	0.660	0.469	4474.117

Continued on next page

Table B.3: (continued)

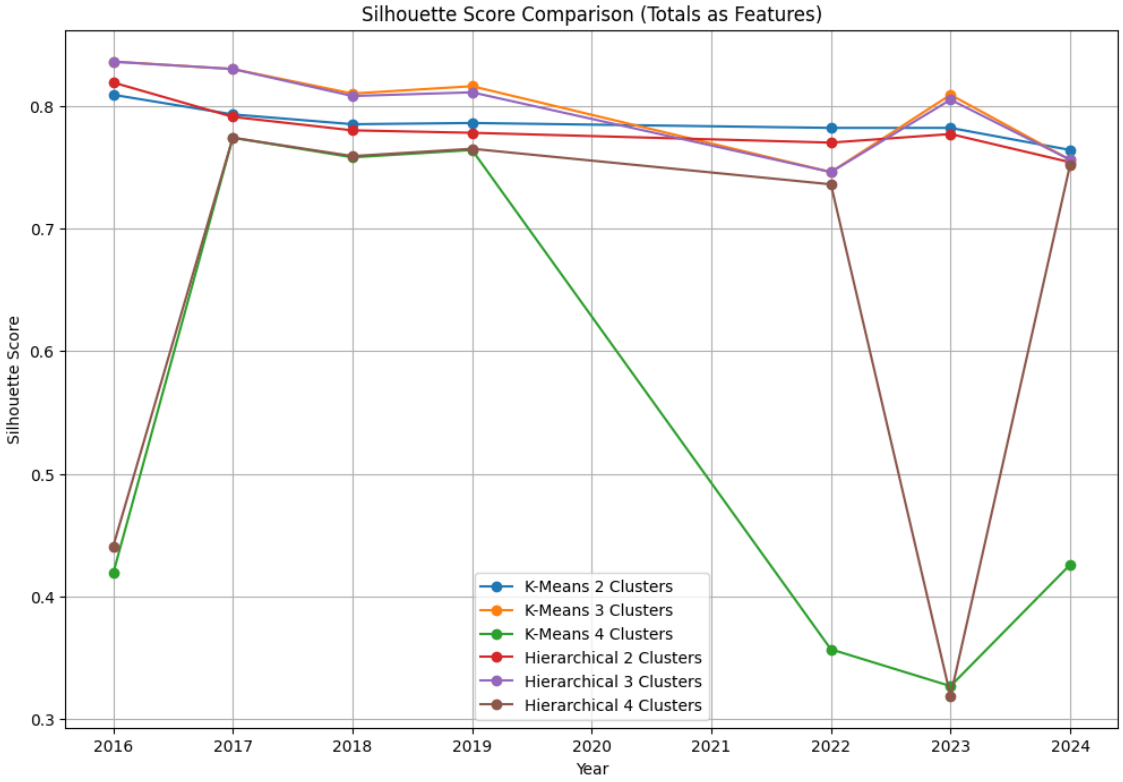
Year	Method	Clusters	Silhouette Score	DBI	CH Index
		3	0.699	0.426	6230.010
		4	0.615	0.503	13155.699

In certain cases, the number of clusters and clustering technique produced the best results, depending on the assessment measure that was employed. For instance, for the 2016 dataset, three clusters yielded the best results (i.e., the outcomes closest to 1) according to the Silhoutte Score, with a tie between the two clustering techniques. KMeans with four clusters was the best result, or the result in which this index is highest, according to the Calinski-Harabsz Index. The best result, however, according to the Davies-Bouldin Index, which corresponds to the lowest value, was Hierarchical clustering with two clusters. However, for some datasets, such as the 2022 dataset, all of these measures confirmed that KMeans with two clusters produced the best outcome.

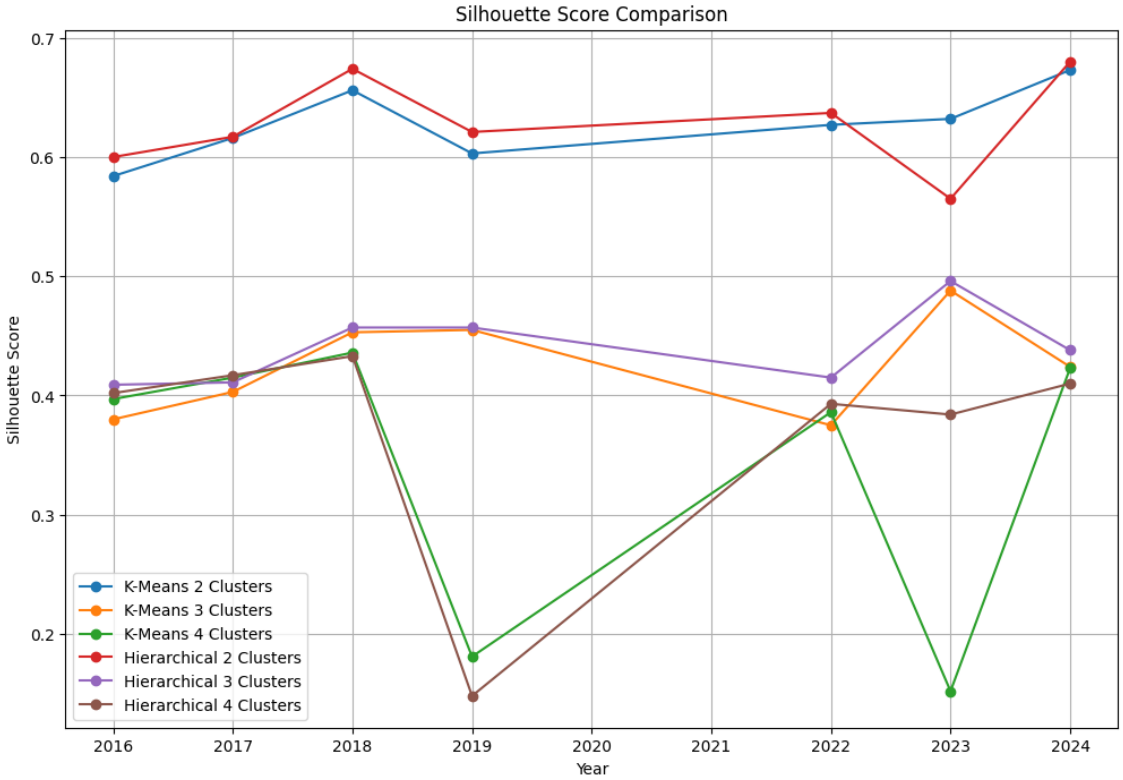
The activity scores of the first two weeks for each of the years with accessible datasets were also used to derive the aggregated student results, as shown in the charts in Figures B.1b, B.2b, and B.3b. The comparison of techniques and number of clusters for the Silhouette Index is displayed in Figure B.1, while Figure B.2 displays the same information for the DBI and Figure B.3b for the CH Index.

When collecting activity data from the first two weeks, clustering using two clusters typically produces the best results, as shown in Figure B.1. Depending on the dataset, either K-Means or hierarchical clustering is the best technique. K-Means, for example, showed atypical behavior in 2023 and performed best in that year. On the other hand, Figure B.1a illustrates that the best number of clusters becomes less obvious when total assessment points are used. Two clusters had better results for 2022 (the COVID-19 year) and obtained similar results for 2024, while three clusters fared better from 2016 to 2019 and in 2023.

It is clear from Figure B.2a that the best clustering strategy differs according to the dataset. The best results in 2016 came from hierarchical clustering with two clusters. In 2017, the results of all constructed clustering techniques were similar. K-Means with four clusters employing total features performed best in 2019, whereas K-Means with three clusters earned the best DBI in 2018. Between 2019 and 2022, no data is available. K-Means with two clusters produced the greatest outcomes in 2022. The best outcomes in 2023 and 2024 were achieved with K-Means with three clusters. The findings from the first two weeks were used as clustering characteristics in Figure B.2b, and hierarchical clustering with two clusters yielded the best results throughout, with the exception of 2023, when K-Means with two clusters outperformed.

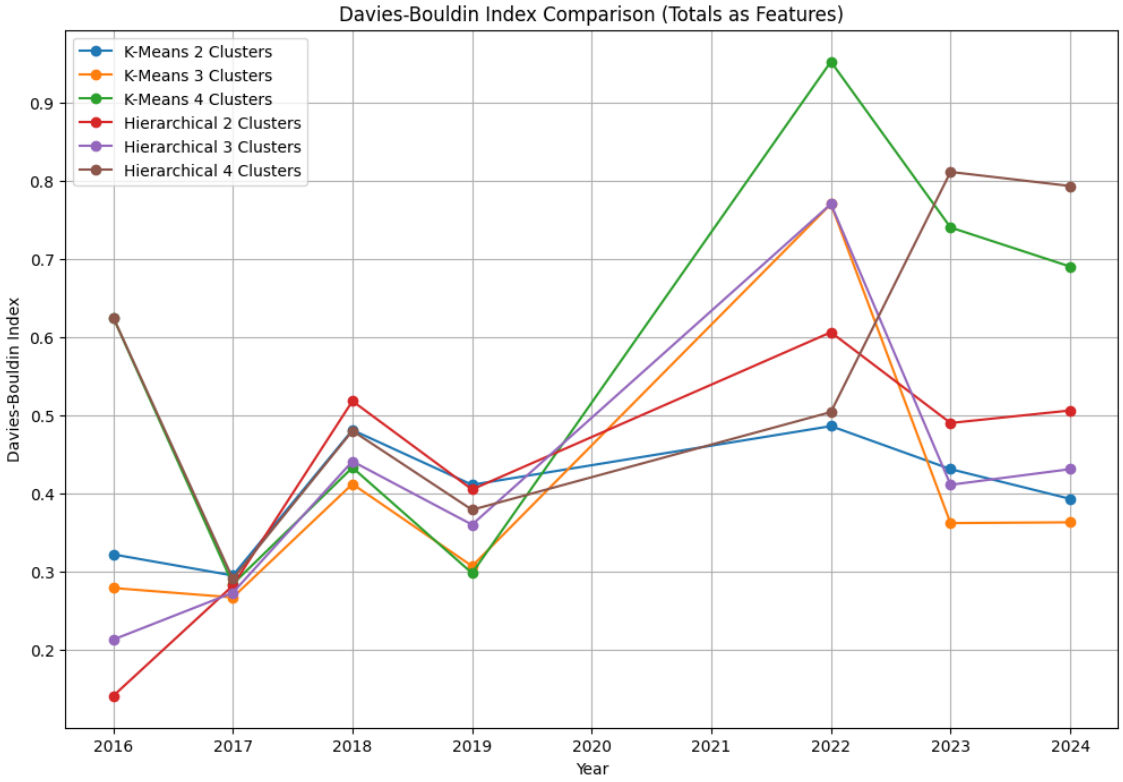


(a) Using total of assessment points.

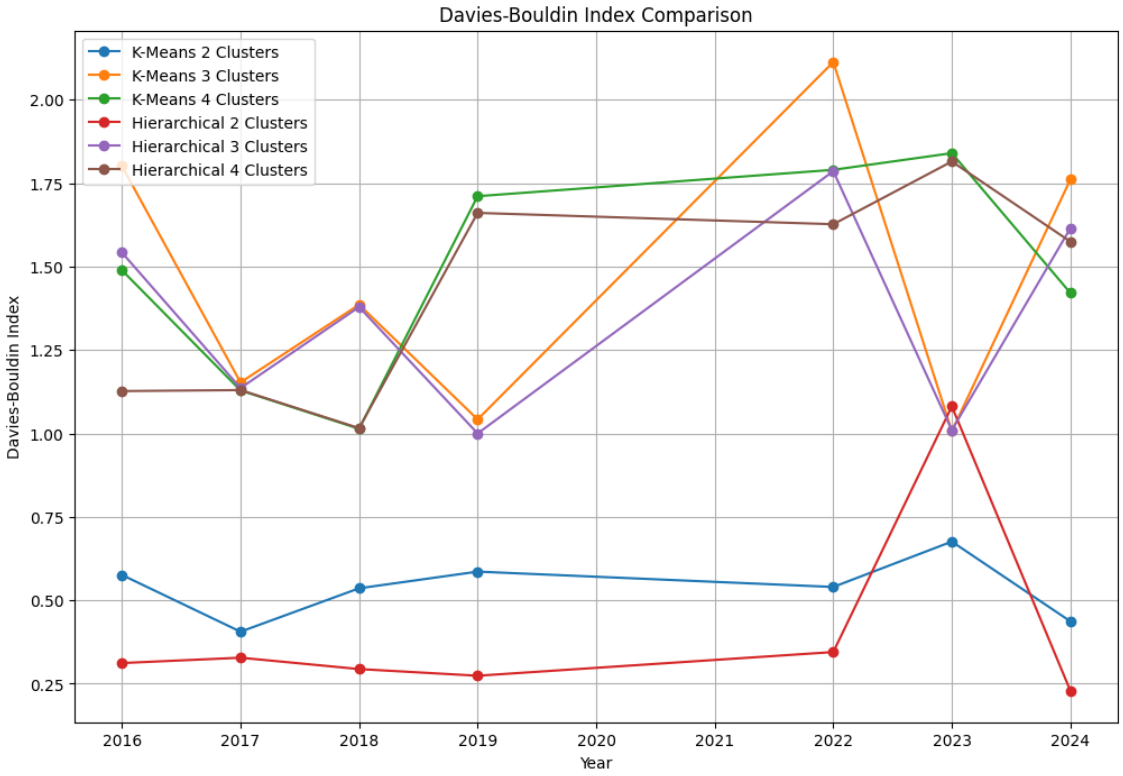


(b) Using results from the first two weeks.

Figure B.1: Silhouette Index comparison between methods and number of clusters using different feature sets.

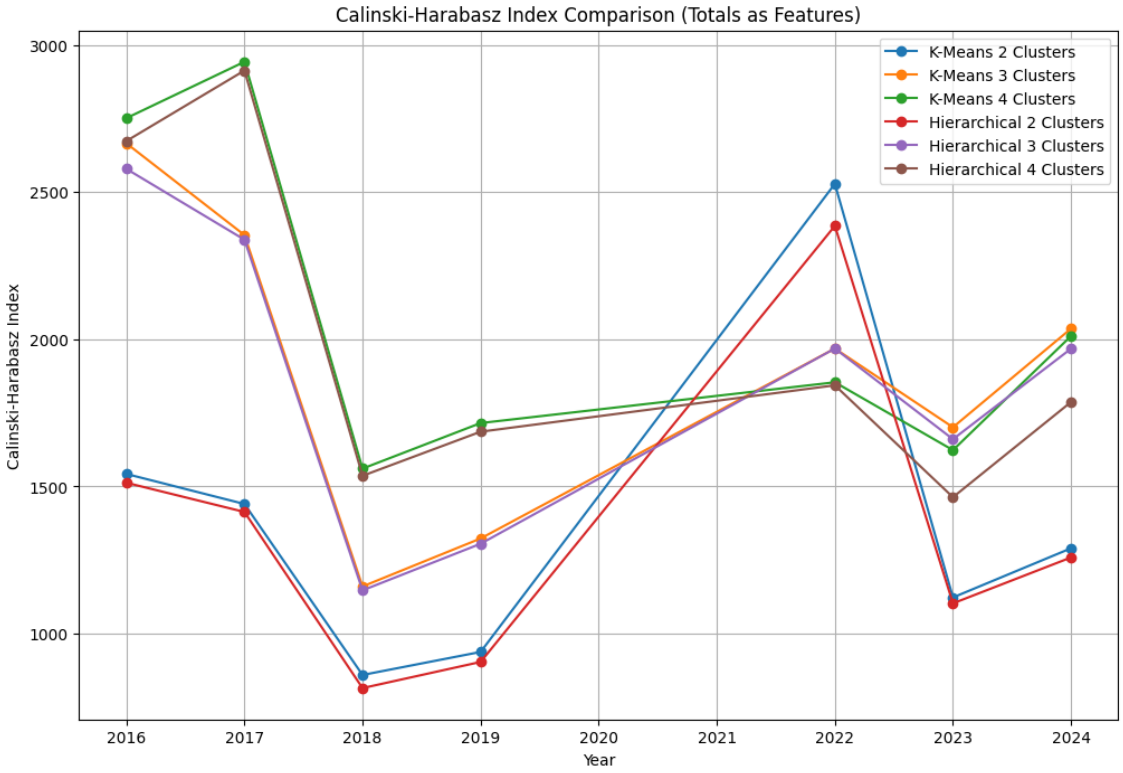


(a) Using total of assessment points.

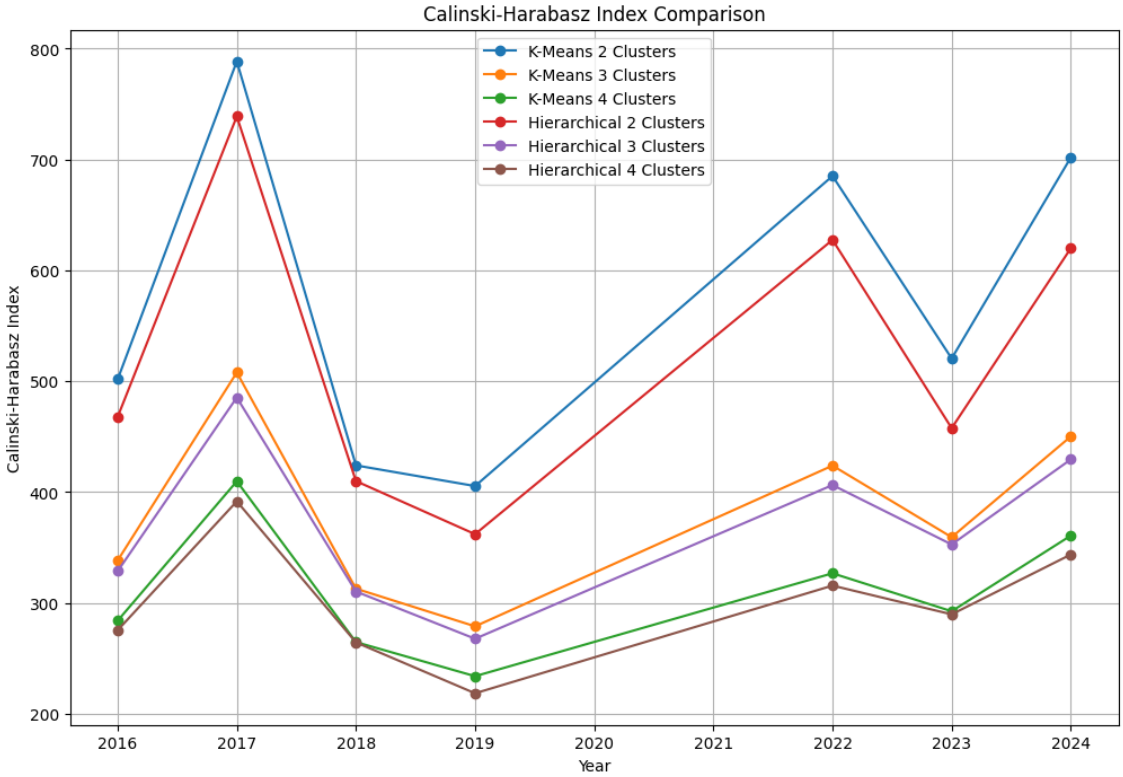


(b) Using results from the first two weeks.

Figure B.2: Davies-Bouldin Index comparison between methods and number of clusters using different feature sets.



(a) Using total of assessment points.



(b) Using results from the first two weeks.

Figure B.3: Calinski-Harabasz Index comparison between methods and number of clusters using different feature sets.

According to the results of the CH Index, which is displayed in Figure B.3, KMC with four clusters from 2016 to 2019; K-Means with three clusters in 2023 and 2024; and KMC with two clusters in 2022 (the COVID-19 year) perform better when complete assessment points are used. K-Means with two clusters was the most effective clustering method when using data from the first two weeks' activity. The performance and interpretation of the clustering results can be greatly impacted by the combination of features used, which emphasizes the significance of feature selection in clustering analysis.

B.2 Predictive Modeling Methods

To address the research questions related to predicting student performance across various evaluation points, several predictive modeling techniques were employed. These methods were integrated into a dedicated Google Colaboratory notebook that performs classification and regression tasks using advanced machine learning algorithms. The notebook focuses on predicting various outcomes such as `Total_AP`, `Total_PP`, `FinalGrade`, and `Passed`, employing a variety of approaches such as feature selection, imputation strategies, and ensemble methods. Table B.4 presents a detailed description of the primary functions used, highlighting their purposes, techniques, and approaches.

Table B.4: Methods used on `CTCT Data Mining - Prediction and Classification` Jupyter notebook.

Method Name: Description
`load_feather_files` : Loads the data files in Feather format and organizes the datasets. Techniques: Feather file reading.
`check_missing_values_in_single_df` : Checks and displays missing values for a specific dataset. Techniques: Missing values count.
`preprocess_data_with_indicators` : Prepares the data through imputation, scaling, and optionally PCA. Techniques: Imputation, Scaling, PCA (optional).
`get_model` : Returns classification or regression models depending on the task type. Techniques: Model selection. Models: Linear Regression, Logistic Regression, SVM, Random Forest, MLP, etc. Evaluation: Accuracy, F1-Score, MSE, R^2, MAE.
`get_test_scores` : Evaluates the models using different metrics based on the task. Techniques: Evaluation. Models: Not applicable. Evaluation: Accuracy, F1-Score, MSE, R^2, MAE.
`perform_grid_search_cv` : Performs cross-validation with Grid Search to find the best hyperparameters. Techniques: Cross-validation, Grid Search.

Continued on next page

Table B.4: (continued)

Method Name: Description
preprocess_data_with_custom_imputation : Prepares data using custom imputation strategies and optionally appliesPCA. Techniques: Imputation, Scaling, PCA.
train_and_evaluate_models : Trains and evaluates classification or regression models, with Grid Search for hyperparameter tuning. Techniques: Model Training, Cross-validation. Models: Logistic Regression, Decision Tree, Random Forest, SVM, k-NN, MLP. Evaluation: Accuracy, F1-Score, MSE, R^2, MAE.
analyze_datasets_for_target_column : Analyzes datasets for a specific target column, applying feature selection and exclusions. Techniques: Preprocessing, Feature Selection, Imputation, PCA. Models: Logistic Regression, Decision Tree, Random Forest, MLP. Evaluation: Cross-validation scores, Test set evaluation (MSE, F1-Score).
analyze_final_grade_with_advanced_ensembles : Analyzes the prediction of final grades using advanced ensemble models like XGBoost, LightGBM, CatBoost, and Stacking. Techniques: Advanced Ensemble Models. Models: XGBoost, LightGBM, CatBoost, Stacking. Evaluation: MSE, R^2, MAE.
analyze_final_grade_with_imputation_strategies : Analyzes final grade prediction using different imputation strategies with advanced ensemble models. Techniques: Imputation, Ensemble Models, Cross-validation. Models: XGBoost, LightGBM, CatBoost, Stacking. Evaluation: MSE, R^2, MAE.

The methods listed in Table B.4 share several characteristics, such as the use of preprocessing steps like imputation and scaling. Ensemble methods, such as bagging, boosting, and stacking, are frequently applied to improve model performance. Additionally, cross-validation is utilized to ensure robust and generalizable model evaluation.

Code Overview

The following code snippets demonstrate the main methods implemented to train and evaluate both classification and regression models. These methods are used to predict the key target variables such as `Total_AP`, `Total_PP`, `Total_SPA`, `Passed`, and `FinalGrade`. The methods are designed to handle missing data, perform feature selection, and apply advanced ensemble techniques for optimal model performance.

Code Snippet B.20: Training and evaluating models for classification or regression tasks.

```
def train_and_evaluate_models(X_train, X_test, y_train, y_test, task_type='classification'
    ↪ , scoring=None, random_state=2024, cv=10):
   # Define models for classification and regression
   if task_type == 'classification':
```

```
        models = {
            'Logistic Regression': LogisticRegression(solver='saga', max_iter=1000,
                ↪ random_state=random_state),
            'Random Forest': RandomForestClassifier(random_state=random_state),
            'Support Vector Machine': SVC(probability=True, random_state=random_state),
            'Neural Network': MLPClassifier(max_iter=1000, random_state=random_state)
        }
    elif task_type == 'regression':
        models = {
            'Linear Regression': LinearRegression(),
            'Random Forest Regressor': RandomForestRegressor(random_state=random_state),
            'SVR': SVR(),
            'Neural Network': MLPRegressor(max_iter=1000, random_state=random_state)
        }

    # Grid search and evaluation
    for model_name, model in models.items():
        grid_search = GridSearchCV(model, param_grids[model_name], cv=cv, scoring=scoring)
        grid_search.fit(X_train, y_train)
        model = grid_search.best_estimator_
        y_pred = model.predict(X_test)

        if task_type == 'classification':
            test_scores[model_name] = {
                'Accuracy': accuracy_score(y_test, y_pred),
                'F1-Score': f1_score(y_test, y_pred, average='weighted')
            }
        elif task_type == 'regression':
            test_scores[model_name] = {
                'MSE': mean_squared_error(y_test, y_pred),
                'R2': r2_score(y_test, y_pred)
            }
    return results, test_scores
```

The `train_and_evaluate_models` function in Code Snippet B.20 handles both classification and regression tasks. It optimizes hyperparameters through grid search and provides performance metrics like Accuracy, F1-Score for classification, and Mean Squared Error (MSE) and R^2 for regression. This method is integral to the analysis of all target columns, supporting both regression and classification tasks.

Code Snippet B.21: Analyzing datasets for target columns using various strategies.

```
def analyze_datasets_for_target_column(datasets_dict, target_column, exclusion_type,
    ↪ feature_set_name=None, imputation_strategy=None, apply_pca=False, apply_scaling=
    ↪ True, columns_to_scale=None, group_col=None, cv=5):
    """
    Analyze datasets for a specific target column. Optionally apply feature selection and
        ↪ exclusions.
    """
```

```
    print(f"Processing datasets for target column: {target_column} with exclusion type: {exclusion_type}")

    # Filter datasets based on exclusion type, feature set, and target column
    if feature_set_name:
        selected_datasets = {
            name: df for name, df in datasets_dict.items()
            if (f"_{exclusion_type}" in name.lower() and f"_{feature_set_name.lower()}_" in name.lower() and target_column.lower() in name.lower())
        }
    else:
        base_key = f'filtered_combined_df_{target_column}' if target_column == 'FinalGrade' else f'combined_df_{target_column}'
        selected_datasets = {base_key: datasets_dict.get(base_key)}

    if not selected_datasets:
        print(f"No datasets found for the given criteria: {exclusion_type}, {feature_set_name}, {target_column}")
        return {}, None

    results = {}
    best_regression_model_info, best_classification_model_info = None, None
    best_regression_score, best_classification_score = float('inf'), float('-inf')

    for dataset_name, df in selected_datasets.items():
        if df is None:
            continue
        task_type = determine_task_type(target_column)
        X_train, X_test, y_train, y_test = preprocess_data_with_custom_imputation(df, target_column, imputation_strategy, apply_pca, apply_scaling, columns_to_scale, group_col)

        # Perform regression analysis if applicable
        if task_type in ['regression', 'both']:
            regression_results, regression_test_scores = train_and_evaluate_models(X_train, X_test, y_train, y_test, task_type='regression', scoring='neg_mean_squared_error', cv=cv)
            results.update({f"{dataset_name}_regression": regression_results})
            for model_name, test_scores in regression_test_scores.items():
                if test_scores['MSE'] < best_regression_score:
                    best_regression_score = test_scores['MSE']
                    best_regression_model_info = {'model_name': model_name, 'params': regression_results[model_name], 'dataset': dataset_name, 'test_scores': test_scores}

        # Perform classification analysis if applicable
        if task_type in ['classification', 'both']:
            y_train_class, y_test_class = (convert_to_multiclass(y_train), convert_to_multiclass(y_test)) if task_type == 'both' else (y_train, y_test)
```

```
            classification_results, classification_test_scores = train_and_evaluate_models(
                ↪ X_train, X_test, y_train_class, y_test_class, task_type='classification'
                ↪ , scoring='f1_weighted')
            results.update({f"{dataset_name}_classification": classification_results})
            for model_name, test_scores in classification_test_scores.items():
                if test_scores['F1-Score'] > best_classification_score:
                    best_classification_score = test_scores['F1-Score']
                    best_classification_model_info = {'model_name': model_name, 'params':
                        ↪ classification_results[model_name], 'dataset': dataset_name, '
                        ↪ test_scores': test_scores}

    return results, (best_regression_model_info, best_classification_model_info)
```

In Code Snippet B.21, the `analyze_datasets_for_target_column` method processes datasets based on exclusion types and feature sets to predict a target column. It supports both classification and regression tasks and handles missing data using various imputation strategies. The method identifies the best model by comparing performance metrics such as MSE for regression and F1-Score for classification.

Code Snippet B.22: Analyzing final grades with advanced ensemble models.

```
def analyze_final_grade_with_advanced_ensembles(datasets_dict, target_column='FinalGrade',
    ↪  exclusion_type='more_restricted', feature_set_name=None, imputation_strategy=None,
    ↪  apply_pca=False, apply_scaling=True, columns_to_scale=None, group_col=None, cv=5):
    ↪
    """
    Analyze final grades using advanced ensemble models like XGBoost, LightGBM, and
        ↪ CatBoost.
    """
    ensemble_models = {
        'XGBoost': xgb.XGBRegressor(random_state=2024, verbosity=0),
        'LightGBM': lgb.LGBMRegressor(random_state=2024, verbosity=-1),
        'CatBoost': cb.CatBoostRegressor(random_state=2024, silent=True),
        'Stacking': StackingRegressor(estimators=[('xgb', xgb.XGBRegressor()), ('lgbm', lgb.
            ↪ LGBMRegressor()), ('mlp', MLPRegressor())], final_estimator=
            ↪ LinearRegression())
    }

    # Similar steps to train and evaluate models with ensemble techniques
```

The `analyze_final_grade_with_advanced_ensembles` method, shown in Code Snippet B.22, employs ensemble models like XGBoost, LightGBM, CatBoost, and stacking models to predict the final grade. This method evaluates models using cross-validation and selects the best based on performance metrics like MSE and R^2.

Example Analyses for `Total_AP` Prediction

The analysis of `Total_AP`, detailed in Code Snippet B.23, involved the use of multiple imputation strategies (mean, median, zero, kNN), with and without feature selection. For the datasets without feature selection, PCA (Principal Component Analysis) was applied for dimensionality reduction, while for the datasets that underwent feature selection, methods such as Mutual Information (MI), Random Forest (RF), LASSO, and Decision Trees (DTs) were used. This approach enabled a thorough evaluation of prediction methods for student performance.

Code Snippet B.23: Analyzing `Total_AP` using different imputation strategies.

```
# Analyze 'Total_AP' using the original dataset without feature selection
total_ap_results_mean, total_ap_best_model_mean = analyze_datasets_for_target_column(
    datasets_dict=loaded_datasets,
    target_column='Total_AP',
    imputation_strategy='mean',
    exclusion_type='more_restricted' # Using the more restricted dataset
)

# Analyze 'Total_AP' using feature selection (MI method)
mi_total_ap_results, mi_total_ap_best_model = analyze_datasets_for_target_column(
    datasets_dict=mi_total_ap_datasets,
    target_column='Total_AP',
    feature_set_name='MI',
    exclusion_type='more_restricted'
)
```

Predicting FinalGrade using Advanced Ensembles

For predicting `FinalGrade`, advanced ensemble techniques were applied. The method `analyze_final_grade_with_advanced_ensembles`compares models such as XGBoost, LightGBM, and CatBoost, which are known to perform well on regression tasks.

Code Snippet B.24: Analyzing `FinalGrade` using advanced ensemble models.

```
# Apply ensemble techniques for predicting FinalGrade
finalgrade_ensemble_results, finalgrade_best_model =
    ↪ analyze_final_grade_with_advanced_ensembles(
    datasets_dict=loaded_datasets,
    target_column='FinalGrade',
    exclusion_type='more_restricted',
    feature_set_name=None, # No feature selection applied
    imputation_strategy='zero' # Using zero imputation for missing data
)
```

The results of this method, as demonstrated in Listing B.24, showed that ensemble techniques improved the predictive performance when compared to traditional regression

methods. Additionally, testing different imputation strategies revealed that using zero imputation provided the best results.

Classification Results

To evaluate the effectiveness of various models in predicting student performance, a series of classification tasks were performed on different target variables: `Total_AP`, `Total_PP`, `Total_SPA`, and `Passed`. The first three variables were treated as categorical by discretizing continuous values into classes, as described earlier. The models were trained using different imputation strategies, and their performance was measured by accuracy, F1-score, and execution time. The hardware used for each task using Google Colaboratory resources is also reported, providing context for the computational efficiency of each approach [36]. In this context, two types of GPUs were used: the `L4`, a high-performance GPU optimized for inference workloads, and the `T4`, a versatile GPU suitable for both training and inference tasks. For more detailed information about Google Colaboratory and its features, please refer to their official page [36].

Table B.5 summarizes the classification performance for `Total_PP`. Here, the k-Nearest Neighbors (kNN) model without any imputation strategy resulted in an accuracy and an F1-score of around 0.7. The use of GPUs significantly reduced the execution time compared to CPU-based processing.

Table B.5: Best classification results for `Total_PP` prediction (after discretization).

Imputation - Feature Selection Methods	Best Model	Accuracy	F1-Score	Execution Time (seconds)
None - None	k-Nearest Neighbors (kNN)	0.700	0.701	599 (GPU T4)
None - None	kNN	0.700	0.701	445 (CPU)
None - MI	Neural Network (NN)	0.638	0.636	408 (GPU L4)
None - RF	NN	0.627	0.606	832 (GPU L4)
None - DT	RF	0.627	0.627	388 (GPU L4)
None - LASSO	NN	0.627	0.606	418 (GPU L4)
None - FR	NN	0.619	0.609	394 (GPU L4)
Mean - None	Random Forest (RF)	0.721	0.719	741 (GPU T4)
Median - None	RF	0.724	0.721	682 (GPU T4)
kNN - None	RF	0.721	0.718	811 (GPU T4)

Continued on next page

Table B.5: (continued)

Imputation - Feature Selection Methods	Best Model	Accuracy	F1-Score	Execution Time (seconds)
Zero - None	Logistic Regression (LR)	0.733	0.730	726 (GPU T4)
Zero - None	LR	0.733	0.730	904 (CPU)
Zero - MI	**NN**	**0.756**	**0.754**	849 (CPU)
Zero - DT	NN	0.753	0.751	866 (CPU)

In Table B.6, the best classification results for `Total_SPA` are displayed. The RF model, when no imputation was applied, yielded an accuracy of 0.9797 and an F1-score of 0.9788, with a modest execution time of 318.86 seconds on an L4 GPU.

Table B.6: Best classification results for `Total_SPA` prediction (after discretization).

Imputation - Feature Selection Methods	Best Model	Accuracy	F1-Score	Execution Time (seconds)
None - None	**Random Forest** (RF)	**0.980**	**0.979**	414 (CPU)
None - None	**RF**	**0.980**	**0.979**	658 (GPU T4)
None - MI	RF	0.972	0.971	408 (GPU L4)
None - RF	RF	0.903	0.902	832 (GPU L4)
None - DT	RF	0.973	0.973	388 (GPU L4)
None - LASSO	RF	0.903	0.902	418 (GPU L4)
None - FR	RF	0.908	0.907	394 (GPU L4)
Mean - None	**Support Vector Machine** (SVM)	**0.980**	**0.979**	533 (GPU T4)
Median - None	**SVM**	**0.980**	**0.979**	536 (GPU T4)
kNN - None	SVM	**0.980**	**0.979**	575 (GPU T4)
kNN - None	SVM	**0.980**	**0.979**	731 (CPU)
Zero - None	SVM	**0.980**	**0.979**	602 (GPU T4)
Zero - None	SVM	**0.980**	**0.979**	773 (CPU)

Continued on next page

Table B.6: (continued)

Imputation - Feature Selection Methods	Best Model	Accuracy	F1-Score	Execution Time (seconds)
Zero - DT	**RF**	**0.980**	**0.979**	870 (CPU)

Finally, Table B.7 provides the classification results for predicting the `Passed` attribute. The RF model without any imputation demonstrated an accuracy of 0.9943 and an F1-score of 0.9942, with the fastest execution time of 72.92 seconds on an L4 GPU.

Table B.7: Best classification results for `Passed` binary column.

Imputation - Feature Selection Methods	Best Model	Accuracy	F1-Score	Execution Time (seconds)
None - None	Random Forest (RF)	0.9943	0.9942	81 (GPU T4)
None - None	RF	0.9943	0.9942	73 (GPU L4)
Mean - None	RF	0.9962	0.9961	207 (GPU T4)
Median - None	Logistic Regression (LR)	0.9956	0.9955	206 (GPU T4)
Zero - None	LR	0.9956	0.9955	150 (GPU T4)
kNN - None	LR	0.9956	0.9955	183 (GPU T4)
No Imputation - MI	LR	0.9962	0.9962	123 (GPU L4)
No Imputation - RF	LR	0.9962	0.9962	126 (GPU L4)
No Imputation - DT	LR	0.9962	0.9962	125 (GPU L4)
Mean - MI	**LR**	**0.9975**	**0.9975**	193 (CPU)

Regression Results

The regression tasks focused on predicting continuous outcomes for `Total_AP`, `Total_PP`, `Total_SPA`, and `FinalGrade`. Various imputation strategies were applied, and models were evaluated based on metrics such as mean squared error (MSE), R-squared value (R^2),

and mean absolute error (MAE). Execution time and hardware usage were also recorded to assess computational efficiency.

Table B.8 summarizes the regression performance for predicting `Total_PP`. The Support Vector Machine (SVM) model with median imputation delivered the best overall performance, achieving an MSE of 7.81, an R^2 value of 0.747, and an execution time of 682.22 seconds using the T4 GPU. Additionally, the SVM model with zero imputation outperformed others in accuracy with an R^2 value of 0.761 and an MSE of 7.38, demonstrating the model's effectiveness across different imputation strategies.

Table B.8: Best regression results for `Total_PP` prediction.

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
None - None	k-Nearest Neighbors (kNN)	8.9	0.712	599 (GPU T4)
None - None	kNN	8.9	0.712	445 (CPU)
None - MI	Random Forest (RF)	10.7	0.652	408 (GPU L4)
None - RF	Neural Network (NN)	11.4	0.632	832 (GPU L4)
None - DT	NN	10.5	0.659	388 (GPU L4)
None - LASSO	NN	11.4	0.632	418 (GPU L4)
None - FR	NN	11.4	0.632	394 (GPU L4)
Mean - None	Support Vector Machine (SVM)	8.1	0.739	741 (GPU T4)
Median - None	SVM	7.8	0.747	682 (GPU T4)
kNN - None	RF	8.0	0.742	811 (GPU T4)
Zero - None	SVM	7.4	0.761	726 (GPU T4)
Zero - None	SVM	7.4	0.761	904 (CPU)
Zero - MI	**SVM**	**6.6**	**0.788**	849 (CPU)
Zero - DT	**SVM**	**6.6**	**0.787**	866 (CPU)

Table B.9 presents the best regression results for predicting `Total_SPA`. The Support Vector Machine (SVM) model with zero imputation delivered the second most accurate predictions, achieving an MSE of 18.40, an R^2 value of 0.8708, and an MAE of 1.81. Additionally, the model with decision tree feature selection (DT) and zero imputation further improved the R^2 to 0.8724, with a slightly lower MSE of 18.17, underscoring its effectiveness for this dataset.

Table B.9: Best regression results for `Total_SPA` prediction.

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
None - None	k-Nearest Neighbors (kNN)	22.6	0.841	414 (CPU)
None - None	kNN	22.6	0.841	599 (GPU T4)
None - MI	Random Forest (RF)	26.2	0.816	408 (GPU L4)
None - RF	Neural Network (NN)	51.7	0.637	832 (GPU L4)
None - DT	NN	23.5	0.835	388 (GPU L4)
None - LASSO	NN	49.0	0.656	418 (GPU L4)
None - FR	NN	47.3	0.668	394 (GPU L4)
Mean - None	Support Vector Machine (SVM)	20.2	0.858	533 (GPU T4)
Median - None	SVM	19.9	0.860	536 (GPU T4)
kNN - None	RF	22.1	0.845	575 (GPU T4)
kNN - None	RF	22.1	0.845	731 (CPU)
Zero - None	SVM	18.4	0.871	602 (GPU T4)
Zero - None	SVM	18.4	0.871	773 (CPU)
Zero - DT	**SVM**	**18.2**	**0.872**	870 (CPU)

Table B.10 summarizes the regression performance for predicting `FinalGrade` using simpler models. The Support Vector Machine (SVM) model with zero imputation delivered competitive results, achieving an MSE of 1.47 and an R^2 value of 0.514 when executed on a T4 GPU. A similar performance was observed on the CPU, with a slightly improved MSE of 1.465, indicating consistent accuracy across different hardware setups. Additionally, the SVM model demonstrated strong performance when combined with various feature selection methods, such as Random Forest (RF), LASSO, and FR, further highlighting its robustness. In contrast, Linear Regression, while simpler, showed limitations with an MSE of 1.84 and an R^2 of 0.391, making it less effective compared to the more advanced models.

Table B.10: Best regression results for `FinalGrade` prediction using simpler models.

Imputation - Feature Selection Methods	Best Model	MSE	R^2	Execution Time (seconds)
Zero - None	**Support Vector Machine (SVM)**	**1.47**	**0.514**	316 (GPU T4)
Zero - None	**SVM**	**1.47**	**0.514**	403 (CPU)
No Imputation - None	Linear Regression	1.84	0.391	90 (GPU T4)
Zero - RF	SVM	1.80	0.770	135 (GPU T4)
Zero - LASSO	SVM	1.80	0.770	131 (GPU T4)
Zero - FR	SVM	1.81	0.769	129 (GPU T4)
Mean - None	SVM	1.51	0.500	290 (CPU)

Given that the `FinalGrade` remains consistent across different academic years and is a critical measure of student performance, more advanced ensemble algorithms were explored to assess their predictive power specifically for this target variable. Unlike other variables, such as `Total_AP`, `Total_PP`, and `Total_SPA`, which may vary significantly from year to year due to changes in course content or structure, the `FinalGrade` offers a stable target for evaluation. This consistency makes it particularly suitable for testing the effectiveness of complex models like CatBoost and XGBoost. Although the other target columns were more challenging to predict due to their variability, focusing on `FinalGrade` allows for a clearer comparison of model performance.

I

Evolution of this Dissertation Project

This annex describes the main aspects of the side projects developed as part of this dissertation project. It starts by providing context on the shifts that occurred and the reasons behind them, followed by the description of the main results and the tools used for both the web application and the development of Virtual Basic Applications (VBA) code. Additionally, this annex addresses the challenges encountered regarding data sharing and presents Quick Response (QR) codes that link to the GitHub repositories created for the codebase of these various projects.

Contextualization

The original initiative aspired to create a mobile application to assist around 1000 first-year CTCT students. The use of artificial intelligence (AI) to provide real-time performance measurements and support would help improve user experience and guide educational strategies. After researching native and cross-platform mobile architectures, Flutter was selected for its compatibility with the User Interface (UI) on both iOS and Android devices.

However, to ensure universal accessibility, the project pivoted to a web application. This change involved an exhaustive evaluation of web technologies, which ultimately led to the adoption of the Django framework, supporting both front-end and back-end development through Python. The decision was influenced by factors such as applicability, community support, learning curve, and alignment with project needs. Studies [63–65] supported Django's flexibility, security, and fast development capabilities. Additionally, the availability of robust Python machine learning libraries proved advantageous for the project's goals. Despite the growing complexity of development, dynamic capabilities were prioritized, resulting in a more attractive user interface with VueJS.

The findings presented in this annex relate to the business understanding phase of the adopted data mining approach: the CRoss-Industry Standard Process Model for Data Mining (CRISP-DM). This phase involved extensive research on the CTCT course, the development of the web application, and Excel file analysis for the development of VBA code. Part of this complementary work included the development of VBA macros to

automate the correction of exercises in CTCT Excel activities. Technical limitations and communication difficulties highlighted the importance of well-defined evaluation criteria and objectives before automating the correction of educational exercises. The division of resources and the complexity introduced by integrating VueJS necessitated a better understanding of Django's Representational State Transfer (REST) framework and the interaction between Application Programming Interfaces (APIs) and the VueJS frontend, influencing the implementation of the originally planned features.

Additionally, hosting the application presented challenges due to limited access to an academic server. Online alternatives, such as PythonAnywhere for Django and Vercel for VueJS, were selected. Although the program effectively compiled questionnaire data from a group of CTCT students, handling multiple users simultaneously presented difficulties. Given the restrictions on user interaction capabilities of the selected hosting platforms, the pilot test revealed the need for more robust infrastructural solutions. As a result, the research focus shifted from web development and related issues like web security, to data mining.

Despite the design modifications, the project yielded significant insights into data analysis, project management, and software development. The primary objective remains to apply machine learning to historical student performance data to improve the education of transferable skills, as detailed in this dissertation. Instead of a developed application, however, this project analyzed seven datasets containing student grades for each activity over seven iterations of the course.

Data Management and Sharing Limitations

This section outlines the Data Management Plan (DMP) for this dissertation project that it's presented in Annex **??**, refering the challenges encountered regarding the public sharing of the collected data.

The data analyzed in this project spans from 2016 to 2024, covering student performance records from the CTCT curricular unit. Initially, the DMP was created for the 2016 dataset, which included 1169 students and 86 attributes. Over time, additional datasets were incorporated, each including new columns to accommodate updated research objectives. Detailed information regarding the parameters of these datasets, such as filenames, column ranges, and final grade fields, is presented in the presented tables (see Tables A.1 and A.3 for attribute description and Table 4.1 for the number of instances that corresponds to the number of students).

The datasets were maintained with open-source technologies like Python's Pandas package and safely kept on Zenodo, however, there are considerable limitations on their public publication. These issues generally result from institutional constraints and concerns about data privacy. Despite the anonymization of the data, apprehensions persist about the possibility of re-identification, especially for students with special statuses. Disseminating these datasets without explicit authorization from NOVA School of Science

and Technology can lead to ethical and legal consequences, as underscored by the school and my research advisor. Consequently, although the datasets have been uploaded to Zenodo in a private repository to safeguard their confidentiality and integrity, public access is still limited. The datasets are kept in formats compatible with open-source tools (CSV, Feather, Pickle files), and minimal documentation is provided; nevertheless, the data cannot be made publicly accessible until formal institutional consent is secured.

This constraint highlights the necessity of obtaining formal authorization for data gathering and dissemination in research endeavors. Future initiatives must stress securing express institutional clearance to enhance data distribution and promote research replication and transparency.

This limitation underscores the importance of securing formal permissions for data collection and sharing in research projects. Future efforts should prioritize obtaining explicit institutional approval to facilitate the wider dissemination of data and support research replication and transparency.

Despite these constraints, the data remains important to the findings and analyses presented in this dissertation. Though public sharing is currently restricted, the insights derived from the data have significantly contributed to this research, highlighting the need for improved processes and authorization mechanisms in future projects.

Web Application

The CTCTLab web application, developed as part of this project, was designed with the goal of improving the learning experience of students enrolled in the CTCT course. It combines a robust backend powered by Django 5.0 with a dynamic and user-friendly frontend developed using VueJS. The system provides a comprehensive set of tools, including interactive questionnaires, real-time analytics, and customizable user experiences for both students and teachers. This section presents an overview of the key components of the web application, focusing on the backend and frontend implementations and their respective functionalities.

Backend - Django

Django 5.0 was chosen as the backend framework due to its security, scalability, and flexibility, making it well-suited for the needs of this application. One of the primary goals of the backend system is to manage different types of users—students and teachers—through custom user models. This allows for role-specific functionalities, such as creating and managing questionnaires, which are linked to the academic activities in the course. Students are able to complete these questionnaires online, with their responses being securely stored in the backend database for further analysis and grading.

The user registration process is a critical component of the backend, allowing new users to activate their accounts seamlessly. Figure I.1 displays the account activation

interface, which was built using Django's REST user authentication features. Once users are registered, they can interact with the platform according to their designated role—students can access course materials and complete quizzes, while teachers can monitor student progress and manage course content.

In addition, the Django-powered admin panel provides teachers with the ability to manage the platform's content and functionality efficiently. Figure I.2 illustrates the admin interface, where teachers can create and modify questionnaires, view student results, and oversee the overall operation of the system. The backend's architecture, leveraging Django's Object-Relational Mapping (ORM) system, ensures that all data transactions between the application and the database are secure and scalable.

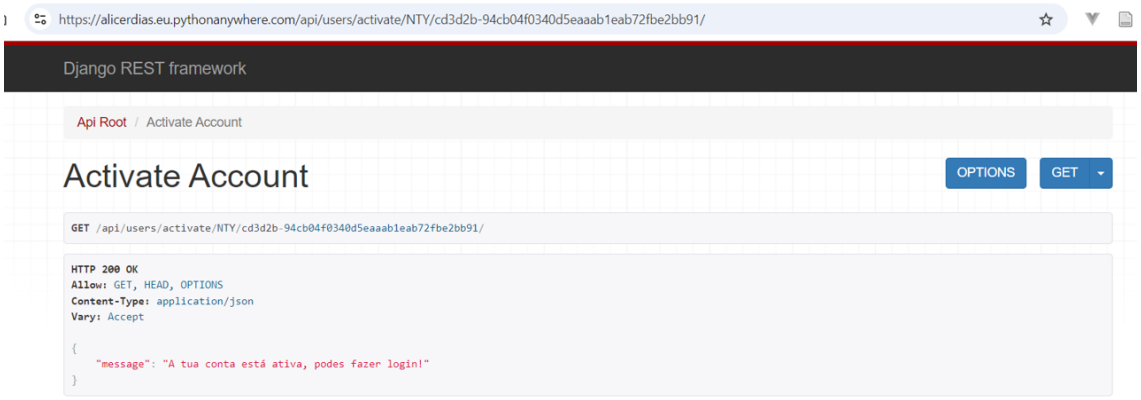


Figure I.1: Account activation interface in Django framework.

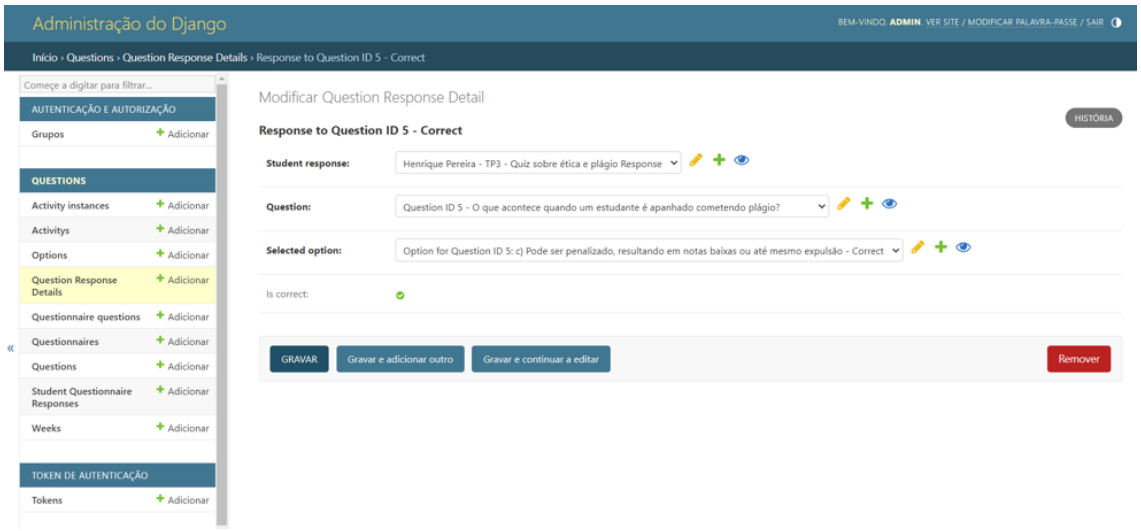


Figure I.2: Admin panel interface for managing users and content.

Frontend - VueJS

The frontend of the CTCTLab application was developed using VueJS, a progressive JavaScript framework well-known for its ability to create dynamic and responsive user

interfaces. The choice of VueJS allows the platform to render content on the client side, providing users with a smooth, interactive experience while maintaining real-time communication with the backend.

The VueJS-based frontend consists of multiple components that enhance the overall user experience. For instance, the main homepage interface, as seen in Figure I.3, provides students with easy access to weekly course content, quizzes, and other relevant resources. The design prioritizes clarity and usability, ensuring that students can navigate through the platform without difficulty. Furthermore, teachers have access to a secure login page, as depicted in Figure I.4, which allows them to manage student progress and access teacher-specific tools. This login system integrates with Django's authentication mechanism to ensure that only authorized users can access the backend functionalities.

Figure I.3: Homepage interface of the application.

One of the key features available to teachers is the ability to monitor student responses to questionnaires in real-time. Figure I.5 illustrates the dashboard where teachers can view an overview of student responses, enabling them to track progress and intervene when necessary. This interaction between the frontend and backend is powered by REST APIs, ensuring that data remains consistent and up-to-date across the platform.

Challenges and Future Work

Throughout the development of the CTCTLab application, several technical challenges were encountered. One of the primary difficulties involved ensuring efficient and secure communication between the VueJS frontend and the Django backend, particularly when handling large volumes of data, such as student questionnaire responses. The real-time

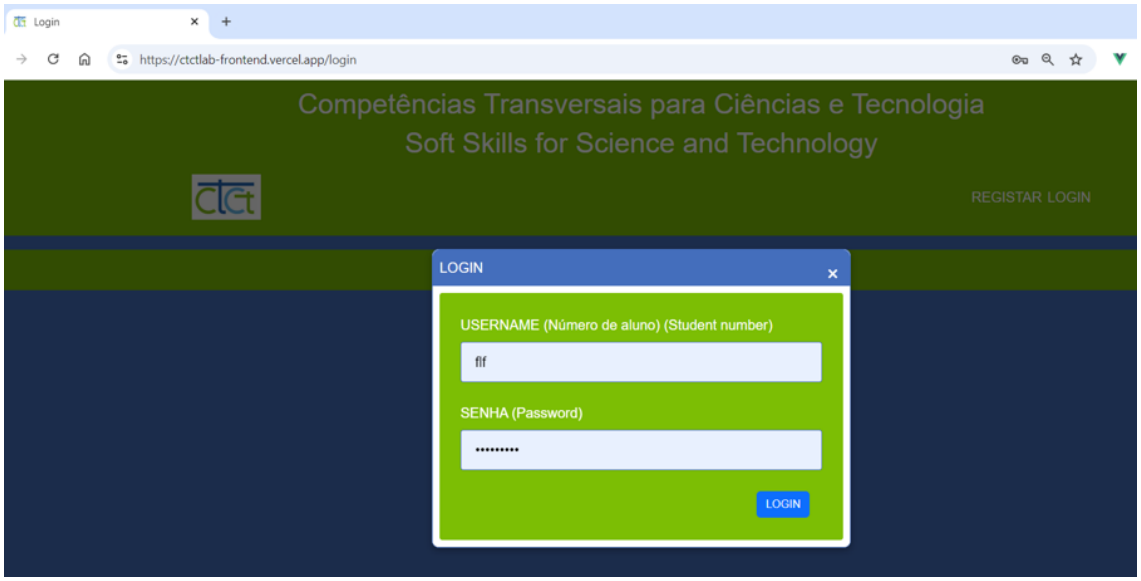


Figure I.4: Teacher login page (username it's not the student number but the begining of the institutional e-mail).

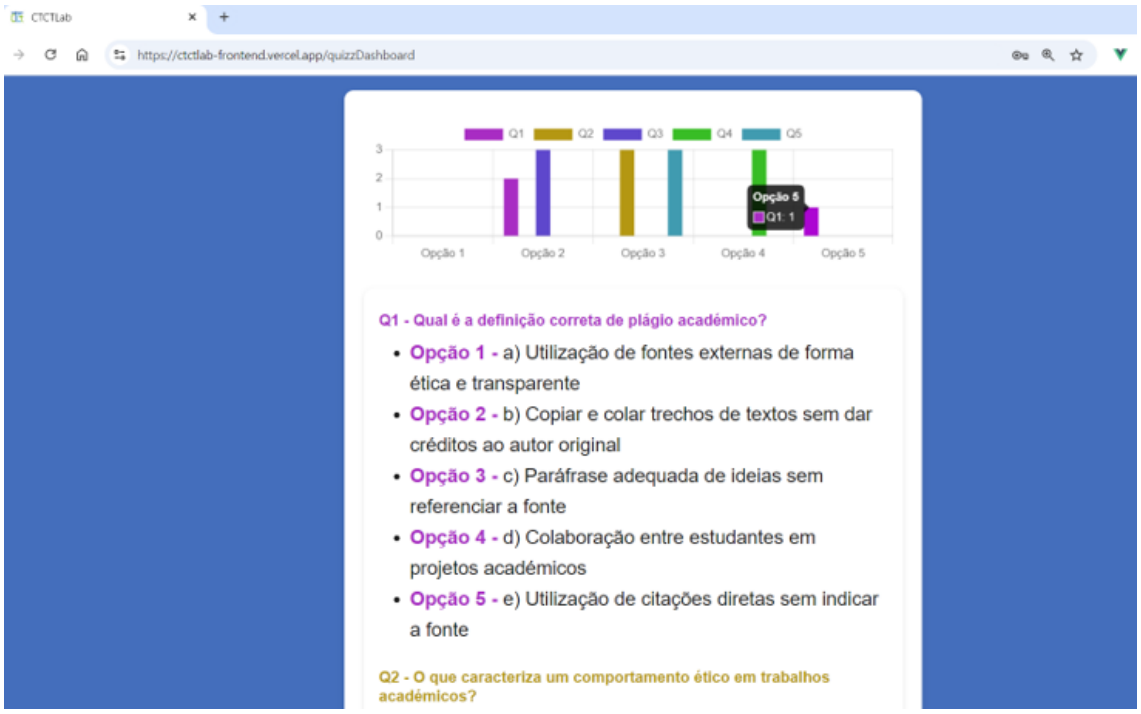


Figure I.5: Detail on the distribution of answers by students obtained when visualizing the teacher's view of the website.

nature of the platform meant that both systems had to remain synchronized at all times, which posed significant challenges in terms of performance optimization (more then 20 students could register but only three students could answer the quizz at the same time when the test was performed).

Additionally, hosting the application presented its own set of challenges. While Django was hosted on PythonAnywhere and VueJS on Vercel, both platforms introduced limitations in terms of scalability and performance under high user load. Figures I.6 and I.7 depict the student registration and correct response tracking features, both of which require optimized backend-frontend interaction to ensure smooth operation.

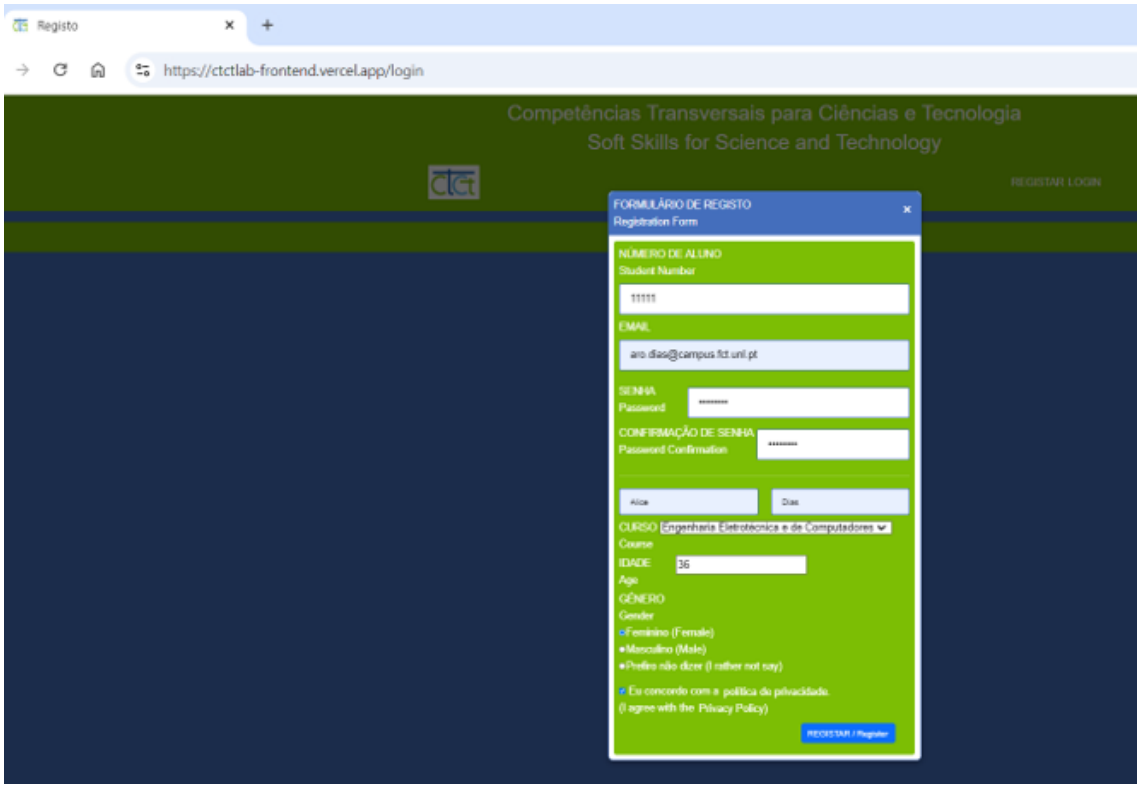


Figure I.6: Student registration modal on the website.

Moving forward, one potential area for improvement is the optimization of communication between the frontend and backend systems. Implementing more efficient data handling methods, such as GraphQL or WebSockets, could improve performance. Additionally, incorporating machine learning models for real-time data analysis could enhance the platform's ability to provide predictive insights based on student performance.

QR Codes for GitHub Repositories

During the development of this project, two GitHub repositories were created to store and manage the code: one for the data mining project and another for the CTCTLab web application. To make it easier to access these repositories, the following QR codes can be scanned to navigate directly to the respective GitHub pages: Figure I.8 for the Data Mining

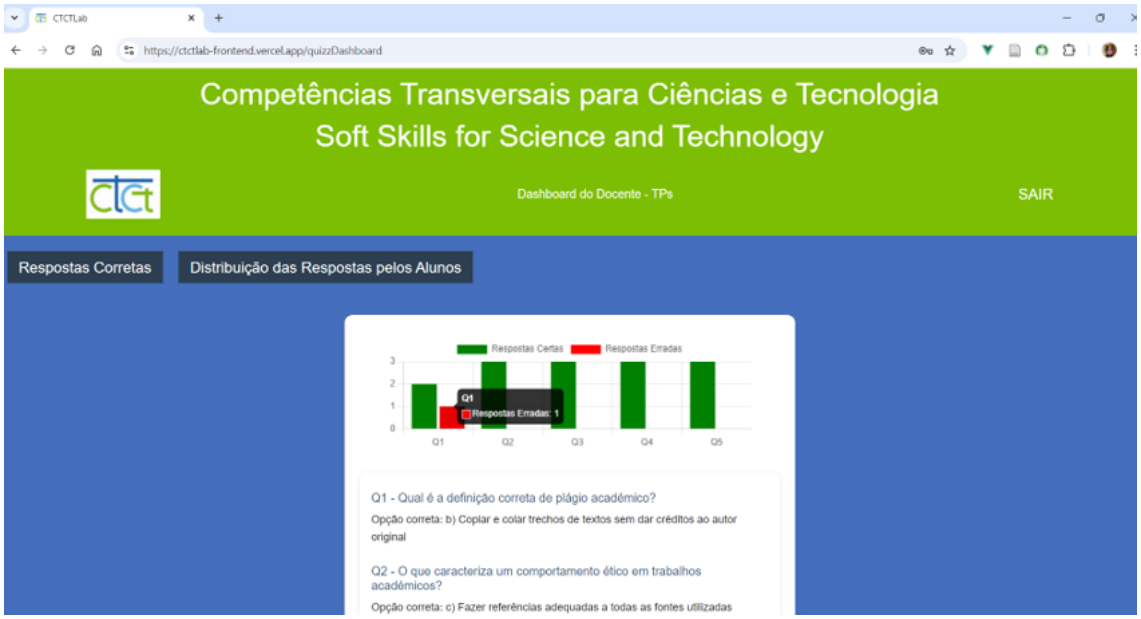


Figure I.7: Correct responses tracking page from teacher's view of CTCTLab.

Project repository (redirecting to `https://github.com/ARoDias/DataMiningProject_CTCT_2024Analysis`), Figure I.9 for the CTCTLab repository (redirecting to `https://github.com/ARoDias/CTCTLab`), and Figure I.10 for the frontend URL of the CTCTLab web application.

Figure I.8: QR code for data mining project repository.

Figure I.9: QR code for CTCTLab web application project repository.

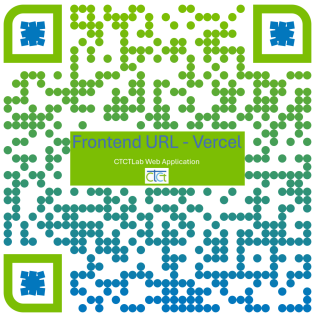


Figure I.10: QR code for frontend URL of the CTCTLab web application.

Automatic Exercise Correction of Excel Files

As part of the complementary work related to this project, a report on the macros developed to automate the correction of Excel-based exercises is presented in Annex **??** to guide CTCT teachers in the execution of the code. These macros were created to address the substantial workload involved in manually grading Excel exercises for large groups of students in the CTCT course. Each year, faculty members are tasked with evaluating individual and group activities from hundreds of students, a process that was inefficient and prone to inconsistencies.

The Excel files used by students in these activities varied in complexity, with some requiring basic data classification and others involving advanced functionalities such as pivot tables. For instance, in one exercise, students were tasked with analyzing a dataset of fruits by categorizing their type, state, and caliber, and then generating tables to summarize this data both in terms of frequency and percentages. The macros were designed to automate the assessment of these tasks by comparing the students' results against predefined correct outputs. This automation ensured not only a reduction in grading time but also increased consistency and fairness in the evaluation process. To achieve this, the VBA code was structured to handle different evaluation criteria depending on the type of task, such as binary checks for simple tasks or more granular assessments for complex data manipulations using pivot tables.

In terms of development, one of the key challenges was ensuring the macros could handle various file formats and inconsistencies in student submissions. For instance, students often made modifications, such as inserting additional columns or changing file structures, which the automation was not initially designed to manage. This caused significant disruptions in the grading process, requiring teachers to intervene manually, which reduced the efficiency the macros were intended to provide. Additionally, compatibility issues across operating systems, especially between Windows and MacOS, created further technical barriers. Despite efforts to make the macros flexible, they often failed to execute correctly under these unforeseen conditions. This underlined the need for a more robust pre-implementation testing phase to ensure the tool could handle real-world variations in student submissions.

A detailed case study on this work was documented in the paper titled:

> **Automating Exercise Correction with VBA Macros: A Case Study in Engineering Education**
> This paper explores the use of Visual Basic for Applications (VBA) macros to automate the correction of Excel-based exercises in an engineering education context. The study aimed to enhance grading efficiency and consistency for large cohorts of students in the Transversal Skills for Science and Technology (CTCT) course at NOVA School of Science and Technology (FCT). The article details the development process of VBA macros, the technical challenges encountered, and the solutions adopted to make the macros operational. In addition, it analyzes feedback from faculty members who used the system, noting benefits in efficiency but also highlighting the need for improvements in accuracy and flexibility. The paper concludes with recommendations for future iterations of automated tools to better support pedagogical goals in higher education.

Although this article was initially submitted to the *Adult Education Quarterly*, it was rejected because it did not align with the journal's scope. The feedback received indicated that the paper's focus on automation in engineering education, specifically through the use of VBA in a highly technical context, was outside the journal's primary focus on adult education. This rejection was not based on the quality of the research but rather on the journal's thematic alignment. As a result, the paper will be revised and submitted to a more appropriate venue focusing on educational technology or engineering education.

The feedback from teachers, collected through a survey, showed that while 64% attempted to use the macros, their experiences were mostly negative. Teachers rated the accuracy of the macros at an average of 2.29 out of 5, and many found that using the automation actually added complexity rather than reducing it. The inconsistency in handling various file formats, combined with the technical difficulties that arose during its use, meant that the VBA macros did not significantly reduce the grading burden. However, this experience highlighted the potential of such tools if further refinements are made. Future iterations should focus on better handling diverse student inputs and improving cross-platform functionality. Although this initial implementation did not succeed in making teachers' lives easier, it opened the door for improvements in automated grading in educational contexts.

EDULEARN24 Conference

The research work titled `Bridging the Gap: Fostering Digital Inclusion and Soft Skills in STEM Education` was presented at the EDULEARN24 conference. This paper investigates the importance of integrating digital inclusion and the development of soft skills in STEM education, using the CTCT course at NOVA University as a case

study. The CTCT course, mandatory for first-year students, aims to cultivate essential non-technical skills such as teamwork, leadership, ethics, and advanced use of spreadsheets. The research identifies a gap between current teaching practices and the cognitive approaches that could better promote the acquisition and retention of these skills.

The paper draws on a literature review in cognitive psychology, specifically focusing on techniques like interleaving and spaced practice. These techniques have been shown to enhance learning retention over time, yet they are underutilized in the current course structure. To address this, the paper proposes a restructuring of the course that spreads out subjects more evenly over time, incorporates more moments for peer review, and creates opportunities for debates. The restructuring is designed not only to improve skill acquisition but also to foster a more inclusive learning environment where students can collaborate and share digital resources, thus bridging the digital divide. By implementing these changes, the goal is to empower students with both the technical and soft skills required for success in STEM fields in a more equitable and inclusive manner.

In addition, the research explores the potential for developing a web or mobile application to facilitate study group formations, allowing students from diverse backgrounds to collaborate more effectively. This digital tool would help ensure that students who are less digitally proficient can still engage with their peers and benefit from shared resources, thereby promoting digital inclusion. The findings presented in this study aim to contribute to the ongoing discourse on effective pedagogical methodologies in STEM education, particularly in addressing the challenges of the digital era. By applying cognitive learning techniques and promoting more collaborative environments, the proposed changes offer a pathway to improve both educational outcomes and digital inclusivity.