NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
SCHOOL OF MECHANICAL ENGINEERING

INTERDISCIPLINARY POSTGRADUATE PROGRAMME
“Translational Engineering in Health and Medicine”

Machine-Learning-based Prioritisation of Long COVID Patients for Specialist Consultation

Postgraduate Diploma Thesis

Argyrios I. Kerezis

Supervisor*: Spyretta Golemati*
Associate Professor, National and Kapodestrian University of Athens

Athens, *June 2024*

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
SCHOOL OF MECHANICAL ENGINEERING

INTERDISCIPLINARY POSTGRADUATE PROGRAMME
"Translational Engineering in Health and Medicine"

Machine-Learning-based Prioritisation of Long COVID Patients for Specialist Consultation

Postgraduate Diploma Thesis

Argyrios I. Kerezis

Supervisor*: Spyretta Golemati*
Associate Professor, National and Kapodestrian University of Athens

The postgraduate diploma thesis has been approved by the examination committee on 20th *June 2024)*

1st member	*2nd member*	*3rd member*
Spyretta Golemati Associate Professor, National and Kapodestrian University of Athens	*Panteleimon Asvestas* *Professor, University of West Attica*	*Konstantina Nikita* *Professor, National Technical University of Athens*

Athens, *June 2024*

……

Argyrios I. Kerezis
Graduate of the Interdisciplinary Postgraduate Programme,
“Translational Engineering in Health and Medicine”,
Master of Science,
School of Electrical and Computer Engineering,
National Technical University of Athens

[This page intentionally left blank]

Abstract

Introduction: Long COVID, a post-acute sequela of SARS-CoV-2 infection, poses a significant burden on healthcare systems. Efficient triaging of patients for specialist consultation is vital for timely and appropriate care. A machine learning (ML) algorithm is proposed in this thesis to prioritise long-COVID patients for initial consultations with cardiologists, pulmonologists, or psychiatrists.

Methods: A ML model was developed using a Support Vector Machine (SVM) classifier with Synthetic Minority Oversampling Technique (SMOTE) for class imbalance handling and feature selection techniques. The dataset was acquired from an ongoing research project that examines holistically Long Covid patients. Data, collected from the first consultation in the Long Covid clinic, from 175 patients included demographics, COVID-19 severity, vaccination status, validated questionnaires (BECK Depression, PTSD, ABC, CAT, EQ-5D, HADS Depression/Anxiety, SF-36), and specific symptoms (intestinal disorders, sleep disorders, precardial pain). The model was trained and validated using 5-fold Stratified Cross-Validation and shuffling of the data.

Results: The SVM model achieved an accuracy of 0.67 and an Area Under the Curve (AUC) of 0.84, outperforming other models (Random Forest, K-Nearest Neighbours, Decision Tree, XGBoost). Notably, the SF-36 General Health (SF-36 GH-N) score emerged as the most important factor for patient risk group classification, while SF-36_BP-N (bodily pain), and the HADS anxiety questionnaire follow.

Conclusion: This study demonstrates the potential of an ML-based approach to prioritise long-COVID patients for specialist consultations. By leveraging patient data and validated questionnaires, the algorithm can guide resource allocation and expedite access to appropriate specialists, improving patient care. The results suggests that a patient's overall health perception plays a crucial role in identifying the most needed specialist for their initial consultation. Future work will investigate the impact of the algorithm on clinical workflow and patient outcomes in a prospective study.

Key Words: Long Covid, Machine Learning, Supervised learning, Unsupervised learning, Multi class classification, AUCROC, Imbalanced dataset

[This page intentionally left blank]

Acknowledgements

> *This thesis is dedicated to my grandparents for teaching me to stay focused, to believe in me and be proud of myself.*

I would like to thank my supervisor, Professor Spyretta Golemati, for providing guidance and feedback throughout this project. I would also like to thank Professor Panteleimon Asvestas and Dr. Kalliopi Dalakleidi for the thoughtful comments and recommendations on this thesis. For providing the data and medical insights I would like to thank Professor Paraskevi Katsaounou and Dr. Archontoula Antonoglou. Finally, I would like to thank my grandma, my family and my friends Kostas, Areti and Elisavet for supporting me during the compilation of this thesis.

[This page intentionally left blank]

Table of Contents

[This page intentionally left blank]

Table of Figures

[This page intentionally left blank]

1. Introduction

This first chapter of the thesis summarises key concepts of Covid and Long Covid, including main features of Covid, the definition of Long Covid, its risk factors and characteristic symptoms, and current methods for diagnosis, and treatment. A comprehensive review is also presented of machine learning applications in Long Covid. The rationale and purpose of this work are outlined at the conclusion of the chapter.

1.1 Introduction

Long-term complications following SARS-CoV-2 infection, often referred to as long Covid, pose a significant challenge to healthcare systems worldwide. It is estimated that 10–20% of people infected by SARS-CoV-2 may go on to develop symptoms that can be diagnosed as long COVID [1]. The diverse and persistent symptoms associated with Long Covid require a multi-faceted approach to patient care, with timely access to the appropriate specialist being crucial for optimal management. However, efficiently triaging patients for cardiology, pulmonology, or psychiatry consultations can be a complex task for clinicians due to the broad spectrum of Long Covid presentations.

This study explores the potential of machine learning (ML) as a tool to prioritise Long Covid patients for initial consultations with specialists. We propose a ML algorithm that utilises patient data, including demographics, Covid-19 severity, vaccination status, validated clinical questionnaires, and reported symptoms, to predict the most relevant specialist for each patient's initial consultation. By leveraging such data, the algorithm aims to streamline patient triaging and expedite access to the most appropriate specialist care, ultimately improving patient outcomes.

This thesis details the development and evaluation of the proposed ML model. We describe the methodology employed, including the chosen ML algorithm (Support Vector Machine with class imbalance handling), feature selection techniques, and the data utilised for training and validation. The performance of the model is then compared to other commonly used ML algorithms. Furthermore, the study investigates the most impactful features identified by the model, providing insights into the clinical factors most relevant for specialist prioritisation in Long Covid patients. Finally, the potential benefits and limitations of this ML-based approach are discussed, along with directions for future research.

1.2 Covid-19 facts

The coronavirus disease 2019 (Covid-19) pandemic has been dubbed the largest worldwide health and socioeconomic calamity in contemporary history [1]. Covid-19, caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), which

can schematically be seen in Figure 1, is an infectious illness that was first found in Wuhan City, China, in late 2019 [2]. Covid-19 can cause a range of symptoms, from mild to severe, including fever, cough, fatigue, and difficulty breathing. The most common symptoms, according to CDC (Centres for Disease Control and Prevention) are outlined in Figure 2[3]. The virus is primarily spread via (i) airborne transmission (respiratory droplets and aerosols), (ii) direct contact (infectious virus deposited on persons), and (iii) indirect contact (infectious virus deposited on fomites) [4]. To prevent the spread of Covid-19, public health measures such as wearing masks, social distancing, and frequent hand washing are recommended. Vaccines have also been developed and are being distributed worldwide to help prevent the spread of the virus.

However, the narrative of Covid-19 goes well beyond its immediate effect. As the epidemic progressed, an additional challenge arose: the shadow of "Long Covid" or "Post-Covid-19 Syndrome."

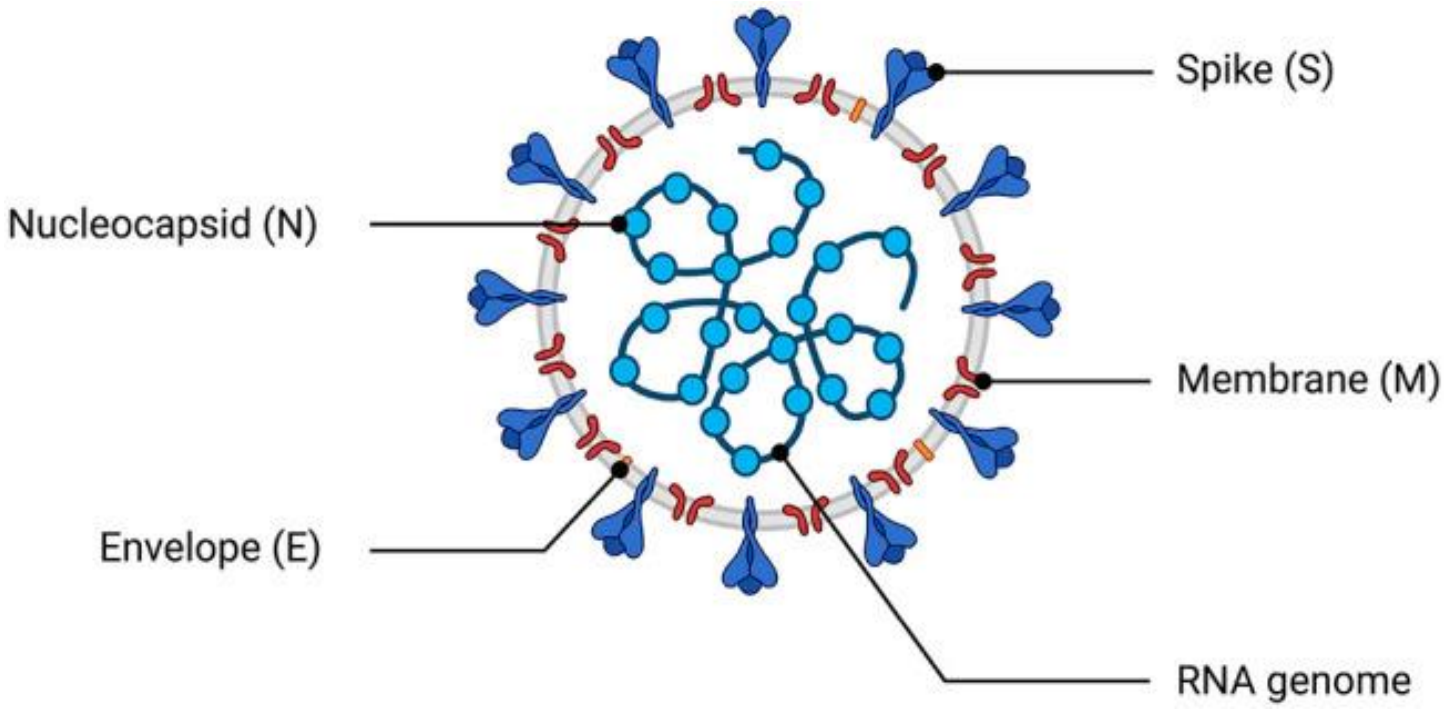


Figure 1: Schematic representation of SARS-CoV-2 [4].

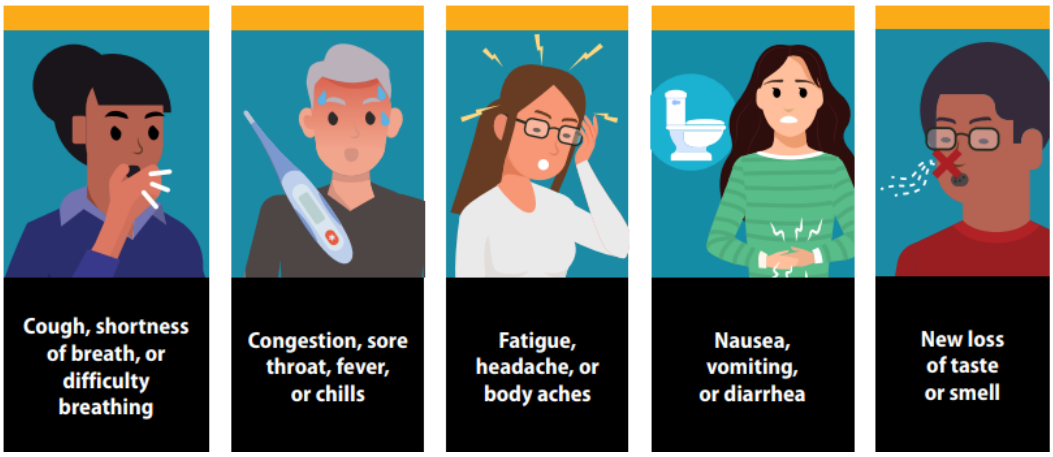


Figure 2: Most common symptoms of Covid-19 according to CDC [3].

1.3 Long-Covid

It is known that certain individuals who have been infected with the virus that causes Covid-19 may experience long-lasting consequences known as Long Covid or Post-Covid Conditions (PCC). Long Covid encompasses a diverse range of persistent symptoms, signs, and health complications that emerge after the acute phase of Covid-19 infection has resolved. This inclusive definition of Long Covid has been collectively formulated by the Department of Health and Human Services (HHS) in conjunction with the Centres for Disease Control and Prevention (CDC) and various other key stakeholders [5]. Additionally, the World Health Organisation (WHO) defines Long Covid as a condition where individuals continue to experience symptoms for a period of at least three months after the acute phase of Covid-19, emphasising the global recognition and impact of this post-viral syndrome [6].

1.3.1 Risk factors

Understanding the risk factors associated with Long Covid is essential in identifying those at higher risk and developing targeted interventions to support and manage this complex condition.

Several factors have been identified that may contribute to the development and severity of Long Covid. These risk factors include the severity of the initial Covid-19 infection, with individuals who experienced more severe acute symptoms being more likely to develop persistent complications. Additionally, age and pre-existing health condition has an important role in the risk of Long Covid, with older individuals and those with diabetes, obesity, or cardiovascular disease being at higher risk of experiencing prolonged symptoms [6].

Genetic factors and immune system responses also play a significant role in the risk of developing Long Covid. Variations in immune response to the SARS-CoV-2 virus and genetic predispositions could influence the persistence and severity of symptoms in individuals with Long Covid. Environmental factors, such as exposure to high levels of viral load during the initial infection or living in areas with high transmission rates, may further contribute to the development of Long Covid [6].

Furthermore, demographic factors have been linked to the likelihood of experiencing Long Covid. Research suggests that women may be at higher risk than men for developing persistent symptoms, and older individuals may be more susceptible to long-lasting effects. Disparities in health outcomes related to race, ethnicity, and socioeconomic status have also been observed, with some minority groups and individuals with lower incomes facing an elevated risk of Long Covid [7]. Demographic disparities in Long Covid highlight the need for equitable care.

1.3.2 Symptoms

Long Covid is characterised by a wide array of persistent symptoms that emerge after recuperating from acute Covid-19 [1]. These symptoms exhibit variability and impact various bodily systems, resulting in a multi-faceted clinical presentation. Common signs of Long Covid encompass fatigue, breathing difficulties, chest discomfort, joint pain, cognitive challenges, and mental health issues like anxiety and depression (see Figure 3) [6]. In this subsection, the pulmonary, cardiovascular, and neuropsychiatric symptoms of Long Covid are discussed, as these are the most commonly encountered in Long Covid clinics.

Pulmonary manifestations may involve ongoing respiratory challenges, such as breathlessness, coughing, and chest pain [6]. Additionally, individuals may encounter pulmonary conditions like fibrosis, impaired lung function, embolism, and pneumothorax, significantly impacting their quality of life and daily activities [6][8].

Cardiovascular symptoms include documented cases of heart muscle dysfunction, irregular heart rhythms, and elevated troponin levels in some patient's data. The underlying physiological mechanisms may include stress-related heart muscle issues, mismatches in microvascular supply and demand, cytokine storms, and heart inflammation with or without pericardial involvement [8]. Monitoring the cardiovascular health of individuals recuperating from Covid-19 is crucial, as heightened D-dimer levels have been linked to severe illness and multi-system disorders.

Neuropsychiatric symptoms affect cognitive abilities, memory retention, emotional well-being, and overall quality of life. These symptoms may encompass cognitive challenges, headaches, alterations in smell and taste perception, sleep disturbances, mood fluctuations, muscle pains, and dysautonomia[8]. The direct infection of the central nervous system (CNS) coupled with systemic inflammation and oxygen

deprivation in Covid-19-positive individuals can contribute to these neuropsychiatric issues.

In summary, the diverse nature of Long Covid symptoms underscores the intricate nature of this condition and the obstacles it presents for both patients and healthcare providers. Comprehensive and interdisciplinary care strategies are imperative to cater to the varied needs of individuals suffering Long Covid. A thorough understanding of the complete spectrum of symptoms linked to Long Covid is essential for devising targeted interventions, enhancing patient outcomes, and refining the overall management of this emerging health challenge.

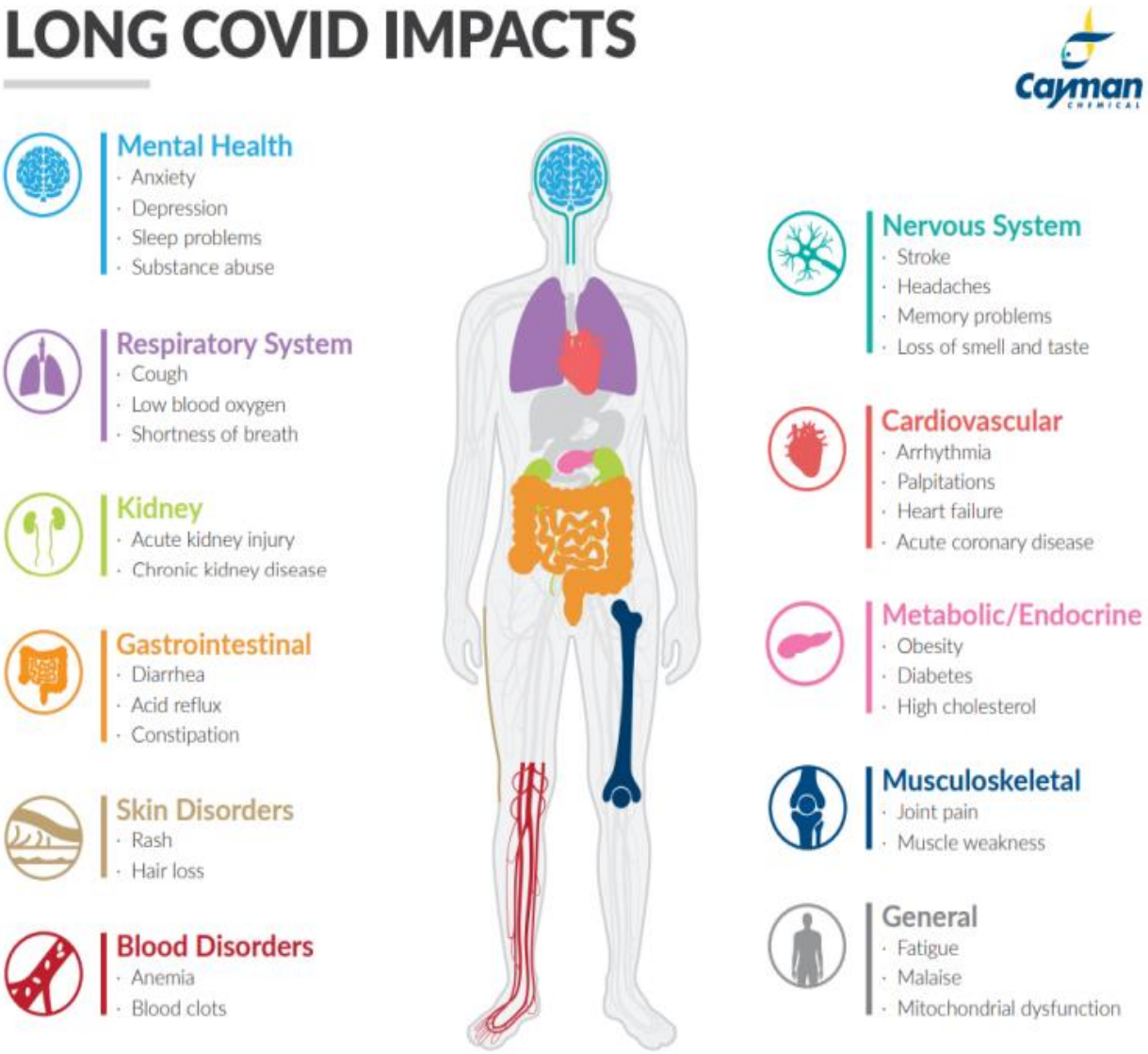


Figure 3: The most common symptoms of Long Covid [9].

1.4 Machine learning applications in Long Covid identification, prediction, and classification: a review of prior work

Long Covid remains poorly understood, with diagnosis often relying on subjective patient reports and lacking standardised criteria. This is where Artificial Intelligence (AI) and Natural Language Processing (NLP) are emerging as powerful tools, offering promising avenues for diagnosis, risk factor identification, and a deeper understanding of this multi-faceted condition.

Scientists use various terms to describe the condition (e.g., Long Covid, Post-Covid conditions, PASC). We also considered various synonyms for the technology used to analyse text data, such as artificial intelligence (AI), machine learning, and natural language processing (NLP). To address this challenge, we searched for relevant publications across several databases (PubMed, Scopus, National Institutes of Health (NIH)). Our search strategy included not only the main term "Long Covid" but also various synonyms and related terms like "Post-Covid conditions" and "PASC." Additionally, we combined these terms with keywords related to artificial intelligence analysis, such as "machine learning," "deep learning," and "natural language processing (NLP). In the following paragraphs, the main tasks addressed using ML methods are briefly described, while Table 1 summarises key information from corresponding works. This review has highlighted various applications of ML, including diagnosis, risk stratification, biomarker identification, and symptoms identification.

1.4.1 Diagnosis

Several studies have explored the potential of AI for Long Covid diagnosis and classification. The first comes from Jha et al. (2023) [10] who used electronic health records (EHR) and High-Resolution Computed Tomography (HRCT) of the lungs of over 1,175 patients, The study aimed to identify the developing risk of pulmonary fibrosis after 90 days of hospital discharge from clinical features retrieved at the time of follow-ups of Covid-19 patients. The dataset included 725 cases of pulmonary fibrosis, the rest were cases of standard lung. They used 75% of the data as training set. They used an optimised XGBoost model and achieved an accuracy of 99.37% on the EHR data and 98.48% on the HRCT data.

Plaff et al. (2022) [11] used the N3C EHR database which includes more than 8 million patients. The study investigated 97,995 individuals with a positive Covid-19 diagnosis. The main purpose of the study was to identify individuals with Long Covid from EHRs. They used information about diagnoses and medications received by patients, along with a requirement of at least 90 days having passed since their initial Covid-19 diagnosis. For this purpose, three XGBoost models were implemented, one for all patients, one for the hospitalised ones and one for the non-hospitalised ones.

For the above models they achieved an Area Under the Curve (AUC) values of 0.92, 0.90 and 0.85, respectively.

The N3C EHR repository was also used by Jiang et al. (2022) [12] to predict Long Covid in hospitalised Covid-19 patients using vital sign data collected within the first seven days of their hospitalisation. The study investigated Long Covid in two patient groups. In the first group, a ML model, XGBoost was used to predict who would develop Long Covid. This model analysed data collected from the patients. In the second group, different models called Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) models were used to analyse sequences of vital sign measurements taken over time. The researchers assessed the performance of all models using AUC with 5-fold cross-validation.

A Random Forest model was used by Sudre et al. (2021) [13] . The study aimed to distinguish between short-term Covid-19 and Long Covid at seven days, from fatigue, headache, dyspnoea and anosmia symptoms. The dataset included data acquired from a mobile health app of 2,149 individuals. The model was implemented using five folds and ten iterations of stratified repeated cross-validation techniques. The model yielded an AUC value of 75.9%. This research suggests that experiencing more than five of these symptoms within the first week of illness may increase the chances of developing Long Covid.

Another study by Subramanian et al. (2022) [14] aimed to distinguish between Covid-19 positive and negative cases and to identify regions in the lungs that may be affected by Covid-19. The authors used two CNN models, one VGG-16 and a ReaNet-50. The models were trained with 925 HRCT images. They achieved an accuracy of 97.13%. Utilising a U-Net model, they segmented and predicted the precise lung area infected with Covid-19 with an accuracy of 99.40%.

Blinka et al. (2022) [15] utilised 26,730 known Long Covid cohort patients' characteristics to identify individuals with Long Covid using health insurance records. They proposed an elastic net regression model with 10-fold cross-validation. The model exhibited specificity of 86% and AUC of 93%.

1.4.2 Risk factor identification

Identifying individuals at higher risk of developing Long Covid is crucial for early intervention and preventative measures. Here too, AI has demonstrated promising potential. Hill et al. (2022) [16] applied multivariate logistic regression, random forest, and XGBoost to the N3C cohort, to examine links between risk factors such as demographics, comorbidities, and treatment, as well as acute characteristics associated with Covid-19, and Long Covid. This study found several important factors that increase the risk of developing Long Covid (PASC). These factors include middle-age, having a severe case of Covid-19, and having certain pre-existing health conditions. The XGBoost and logistic regression models performed similarly, both

achieving an AUC of 0.73. The Random Forest model trailed behind with an AUC of 0.69.

Moreno- Pérez et al. (2021) [17] implemented a multiple logistic regression model on 277 patient demographics and comorbidities acquired from EHRs, to identify risk factors associated with the acute phase of Covid-19 infection that may contribute to the development of Long Covid. The analysis focused on cases where the condition had a 95% chance of occurring within a specific timeframe. While the cause is unclear, the study suggests some risk factors like severe pneumonia and male sex.

1.4.3 Phenotype and biomarker identification

Due to the wide variety of symptoms in Long Covid patients, diagnosing the condition is challenging, highlighting the importance of finding biomarkers to develop specific treatments. A study from Patel et al. (2023) [18] sought to find new blood biomarkers for Long Covid. The study used a random forest classifier trained on expression of 2,925 unique blood proteins. The random forest used 3-fold cross-validation, 10 trees and maximum depth of 3. The study successfully identified 119 key proteins in the blood that can be used to distinguish Long Covid outpatients from healthy controls with exceptional accuracy. This method achieved a perfect score of 100% in classification accuracy, AUC and F1-score.

Zhang et al. (2023) [19] used 34,605 EHRs from two large cohorts, INSIGHT and OneFlorida+, part of the National Patient Cantered Clinical Research Network data to identify subgroups (subphenotypes) of Long Covid in patients who have recovered from a confirmed Covid-19 infection. Using a topic modelling technique, they analysed over 137 symptoms and conditions, grouping them based on how frequently they occurred together in patients. This identified four Long Covid subphenotypes:

- Cardiac and renal issues (33.75% in development cohort, 25.43% in validation)
- Respiratory, sleep, and anxiety problems (32.75% in development cohort and 38.48% in validation)
- Musculoskeletal and nervous system complications (23.37% in development cohort and 23.35% in validation)
- Digestive and respiratory issues (10.14% in development cohort and 12.74% in validation), linked to specific patient demographics

Zhu et al. (2022) [20] aimed to identify Long Covid in outpatients using their 719 patient electronic medical records (clinical notes) from doctor visits. They focused on patients who had experienced persistent symptoms for at least 30 days following a confirmed Covid-19 diagnosis. The timeframe for analysis extended up to one year (365 days) after the initial diagnosis. Additionally, the study explored potential subgroups (phenotypes) within Long Covid based on the identified characteristics. To

ensure the accuracy of their model, the researchers used 5-fold cross-validation. They divided their data into three sets for training, validation, and testing. 60% of the total dataset was used to train the model. The remaining 40% was used for the test and validation sets, 20% each. The study compared three different pre-trained Bidirectional Encoder Representations from Transformers (BERT) models, which are powerful language processing tools, to automatically identify patients with Long Covid from their clinical notes. The model specifically designed for clinical text, ClinicalBERT, achieved the best performance with a sensitivity score of 0.88. This means the model correctly identified 88% of patients with Long Covid from their medical records. Finally, based on the model classifications, the researchers were able to identify potential subgroups (phenotypes) within the Long Covid population.

1.4.4 Symptoms identification

Symptom identification refers to the process of accurately recognizing and classifying the various physical and mental experiences reported by Long Covid patients. Wang et al. (2022) [21] created a Long Covid lexicon of symptoms (PASCLex). The researchers evaluated PASCLex, a method that uses a medical dictionary (ontology) and artificial intelligence (NLP) to automatically extract Long Covid symptoms from clinical notes. They trained PASCLex on a large dataset of 23,505 patients (90% of the total dataset) with 299,140 related clinical notes. A separate validation subset of 2,612 patients (10% of the total dataset) with 29,739 clinical notes was used to confirm the accuracy of the method. The researchers built their method using a comprehensive medical reference system called the Unified Medical Language System (UMLS). This allowed their system to achieve high accuracy in identifying Long Covid symptoms. Specifically, the method achieved a precision of 94% and a recall of 84%.

Another approach was implemented by Katsarou et al. (2022) [6]. They collected free text data from 208 individuals in a Facebook post regarding symptoms of Long Covid. Via a decisions tree model, they classified Long Covid patients based on demographics, disease duration, hospitalisation, and symptoms.

Patient-generated data offer a unique window into the subjective aspects of Long Covid, complementing traditional clinical data and enriching our understanding of the disease.

Furthermore, Branda et al. (2022) [22] utilised 296,154 Twitter data to track the progression of Long Covid over time. By analysing tweets for more than 150 days, researchers aimed to create detailed timelines of symptoms and conditions experienced by individuals. They used an NLP model and reached an accuracy value of 75%.

Table 1: Summary of studies on Machine Learning applications in long-covid.

Study	Data	Al method/model	Task	Output (%)	Key Findings
Diagnosis and classification					
Jha et al. (2022)	1,175 EHR & HRCT	XGBoost	Binary classification of pulmonary fibrosis	Accuracy: 99 precision: 99	An XGBoost system achieved 99% accuracy in detecting pulmonary fibrosis using EHR or HRCT data in Covid-19 patients, suggesting its potential as a fast and efficient tool for early diagnosis.
Plaff et al. (2022)	N3C database	XGBoost	Long Covid classification	Hospitalised AUC: 90 Non-Hosp. AUC: 85 All patients AUC: 92	The models identified features (respiratory problems, fatigue, pre-existing health conditions), and highlighted limitations (data being skewed towards patients who use the healthcare system more).

Jiang et al. (2022)	N3C database	CNN, XGBoost, LSTM	Long Covid classification	LSTM AUC: 59.9 XGBoost AUC: 82.2 CNN AUC: 61.6	Vital signs within the first week of hospitalization were more informative than pre-existing conditions for Long Covid prediction. Time series data in another subcohort benefitted from CNN and LSTM models. Overall, vital signs were significant for Long Covid analysis, but limitations include a small sample size with vital sign data and unclear medical meaning for some features.
Sudre et al. (2021)	2,149 self-reported data	Random Forest	Long Covid classification	AUC: 75.9	14.5% of people with symptomatic Covid -19 were found to have Long Covid lasting 4 weeks, with women and older adults more likely affected. Early disease severity and a simple model using age, sex, and symptom count could predict who would

					experience Long Covid.
Subramanian et al. (2022)	925 HRCT	VGG-16, ResNet-50, U-Net	Long Covid classification	Accuracy: 97.1 - 99.4	The model calculates a CT severity score to assess lung involvement, potentially linked to disease severity and long-term complications.
Blinka et al. (2022)	26,730 patient data	Elastic Net regression	Long Covid classification	AUC: 93 sensitivity: 86 specificity: 86	The model identified several factors associated with Long Covid, including symptoms (shortness of breath and fatigue), pre-existing conditions, and demographics (female sex, older age).
Risk factor identification					

Hill et al. (2022)	N3C database	Random Forest, XGBoost	Identification of risk factors associated with Long Covid	XGBoost AUC: 73 Random Forest AUC: 69	Important risk factors for PASC such as middle age, severe Covid-19 disease, specific comorbidities.
Moreno - Pérez et al. (2021)	277 EHRs	Multiple logistic regression	Identification of risk factors associated with Long Covid	Cumulative Incidence Value 95	Around 50% of Covid-19 patients still experience fatigue, shortness of breath, and psychological distress 10-14 weeks after infection. These symptoms mostly resolved by 16-18 weeks. While the cause is unclear, the study suggests some risk factors like severe pneumonia and male sex.
Phenotype and biomarkers identification					
Patel et al. (2023)	2,925 unique blood proteins expressions	Random Forest, NLP	Blood biomarker detection associated with Long Covid	AUC, accuracy, F1-score: 100	Blood protein analysis in Long Covid patients revealed 119 potential markers. These markers could be key for diagnosis and personalised treatments.

Zhang et al. (2023	34,605 EHRs	Topic modelling clustering	Define Long Covid subtypes	No scores	This study identified four sub-types of Long Covid(PASC) based on EHRs. These sub-types group patients with similar new health problems after Covid infection. The findings suggest PASC is not one condition, but a group affecting different organs. This can inform treatment plans for Long Covid patients.
Zhu et al. (2022)	719 EHRs	BERT	Identification of Long Covid and potential computational phenotypes	Sensitivity: 88.1	This study explored using EHRs to identify Long Covid patients. ClinicalBERT, a pre-trained model, achieved the best results in finding potential cases by analysing written notes. While the analysis provided insights into relevant words like "fatigue" and "hospitalisation", limitations like data size and interpretability

					prevent definitive diagnosis based on these findings.
Symptoms identification					
Wang et al. (2022)	328,879 clinical notes	PASCLex (NLP) model	Symptom identification	Precision: 94 recall: 84	A new method using AI and EHRs can create a more complete list of symptoms for Long Covid patients. This could improve diagnosis and future research for this condition.
Katsarou et al. (2022).	208 individuals via Facebook	Decision Trees	Symptom identification	No scores	This Greek study found similar Long Covid symptoms in hospitalized and non-hospitalized patients, including fatigue, neurological problems, and smell disorders. However, hospitalised patients showed more psychological

					issues. Notably, similar symptoms across patient groups suggest a possible genetic link to Long Covid.
Branda et al. (2022)	296,154 Twitter data	NLP, SVM	Symptom identification	Accuracy: 75	Analysing their experiences revealed symptom patterns and disease progression. Such data can aid clinicians and improve understanding of Long Covid. Social media offers a unique way to capture frequent health updates, potentially applicable to other chronic illnesses. However, limitations include reliance on self-reported data and the unstructured nature of social media text.

1.5 Rationale and purpose of study

Long Covid poses a significant challenge for healthcare systems. Traditional diagnostic and management methods, heavily reliant on subjective patient reports and lacking standardized criteria, often lead to inefficiencies. These inefficiencies manifest as delayed diagnosis, suboptimal treatment planning, and a critical gap in triage strategies for Long Covid patients.

Currently, there is a concerning absence of research focused on developing targeted triage protocols for Long Covid patients. This lack of a systematic approach often results in mismatched referrals, where patients are directed to specialists who may not be best equipped to address their specific needs. This not only wastes valuable time and resources for patients and the healthcare system but also delays access to appropriate care.

ML offers a promising solution to address these limitations. ML algorithms have the potential to analyse vast amounts of patient data, including demographics, medical history, Covid-19 severity, and reported symptoms. By identifying intricate patterns within this data, ML models can be instrumental in:

- Enhancing Diagnostic Accuracy: ML algorithms may be able to identify subtle patterns in patient data that are missed by traditional methods, leading to more accurate diagnoses of Long Covid.
- Optimizing Triage Processes: By learning from the data, ML algorithms can be trained to predict the most appropriate specialist for each Long Covid patient based on their individual characteristics. This can streamline the triage process and ensure patients receive the most relevant care sooner.
- Supporting Treatment Planning: Analysing patient data, ML models can help identify potential risk factors and disease trajectories, informing the development of personalized treatment plans for Long Covid patients.

This study aims to bridge the critical gap in Long Covid patient triage by harnessing the power of machine learning. We propose an innovative ML algorithm that utilizes a comprehensive dataset of patient information. This information includes demographics, Covid-19 severity, vaccination status, validated clinical questionnaires, and reported symptoms. The algorithm will be meticulously trained to predict the most relevant specialist for each patient's initial consultation.

We believe this approach has the potential to revolutionize Long Covid management by streamlining the triage process. Our ML model can expedite patient evaluations, ensuring they are directed to the most appropriate specialist for their specific needs. This can lead to faster access to specialized care, potentially improving patient outcomes and alleviating the burden on healthcare systems by freeing up valuable time and resources for clinicians.

By evaluating the effectiveness of our proposed ML model, this study aims to contribute significantly to the development of innovative tools that can optimize Long Covid management and ultimately improve patient care.

2. Theory of Machine Learning

Machine learning, a subfield of artificial intelligence, empowers computers to learn from data without explicit programming. This learning process allows them to identify patterns, make predictions, and improve their performance over time. By delving into the theoretical underpinnings of ML algorithms, this chapter provides the foundational knowledge necessary to understand, implement, and analyse these powerful tools. In the following sections the theoretical background behind ML, different ML models, optimization techniques, data handling techniques and evaluation metrics will be discussed.

2.1 Introduction

Pioneering figures like Alan Turing laid the groundwork for AI with the Turing test, which challenged machines to exhibit human-like intelligence through conversation [23]. Arthur Samuel is credited with one of the first definitions of machine learning, describing it as "a field of study that gives computers the ability to learn without being explicitly programmed" [24]. Building on this foundation, Tom M. Mitchell, a leading figure in machine learning, offered a more formal definition that captures the essence of this field [25]. These definitions highlight the core aspects of machine learning: the ability to learn from data, improve with experience, and ultimately make predictions or informed decisions [23].

Mitchell's definition emphasises the notion of experience-driven improvement in a computer program's performance on specific tasks. According to Mitchell, a program is considered to learn from experience (E) with respect to a set of tasks (T) and a performance measure (P) if its performance on those tasks, as measured by P, improves with the experience gained through E [25]. This definition highlights three key elements:

- Tasks (T): Machine learning is not a one-size-fits-all solution. There are various tasks a program can learn to perform, such as image recognition, spam filtering, or stock price prediction.
- Experience (E): This refers to the data the program is exposed to during the learning process. The quality and quantity of data significantly impact the program's ability to learn and improve.
- Performance Measure (P): This metric helps us evaluate how well the program performs on the defined tasks. It could be accuracy, error rate, or any other measure relevant to the specific task.

Mitchell's definition highlights that machine learning is not simply about processing data. It's about the program's ability to leverage experience to demonstrably improve its performance on a particular set of tasks.

2.2 Supervised Learning

Supervised learning is a cornerstone of machine learning techniques. It involves training a model using labelled examples, where each data point has a corresponding desired output. This type of learning is widely used in applications where past data can be leveraged to predict future events. Common examples include fraud detection in credit card transactions or classifying species based on specific measurements [26].

Supervised learning tasks fall into two main categories: classification and regression. In classification problems, the output labels are discrete categories. For example, a segment of text could have a category label, such as "Spam" vs. "Legitimate Email" or "Positive" vs. "Negative Movie Review." The model learns to classify new data points into these predefined categories.

On the other hand, regression problems deal with continuous outputs, such as predicting house prices or stock market trends. The model learns from this labelled data to make predictions on new, unseen data. The goal is to minimize the difference between the predicted outputs and the actual values. During the training process, the algorithm separates the raw input data from its corresponding labels. Based on this labelled data, the algorithm builds a predictive model that can then be used to generate labels (classification) or continuous values (regression) for new data points [26]. The following figure 4 explains the concept.
Supervised learning plays a vital role in various real-world applications. It empowers systems to make informed decisions and accurate predictions based on historical data, impacting fields like finance, healthcare, and self-driving cars.

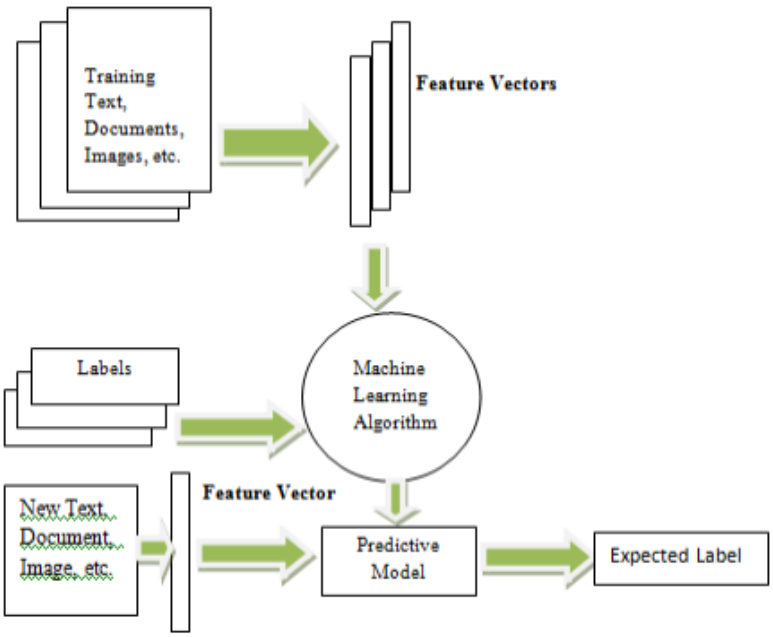


Figure 4: Supervised learning model [26].

2.3 Unsupervised Learning

Unsupervised learning is a fundamental machine learning paradigm concerned with analysing unlabelled data to discover underlying patterns or relationships. Unlike supervised learning, which requires labelled examples for training, unsupervised learning thrives in scenarios where labelled data is scarce or expensive to acquire. This makes it particularly valuable for tasks like customer segmentation in marketing or anomaly detection in financial transactions [26]. This concept is demonstrated in Figure 5.

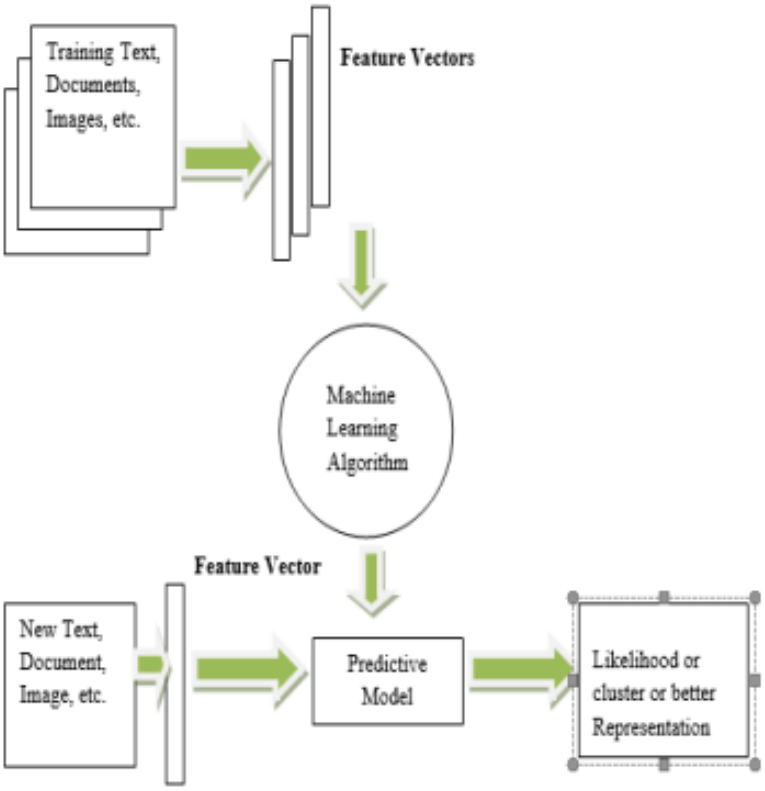


Figure 5: Unsupervised learning model [26].

Unsupervised learning algorithms can be broadly categorized into three main areas: clustering, dimensionality reduction, and anomaly detection.

- Clustering groups data points together based on their similarity. This allows for the identification of distinct categories within the data, even without predefined labels. Common clustering algorithms include K-means clustering and hierarchical clustering.
- Dimensionality reduction aims to simplify complex data by capturing its essential features in a lower-dimensional space. This improves computational

efficiency and visualization capabilities. Techniques like Principal Component Analysis (PCA) and t-SNE are commonly used for dimensionality reduction [26].

- Anomaly detection algorithms identify data points that deviate significantly from the majority of the data. This can be helpful in fraud detection or system health monitoring.

By exploring the inherent structure of the data, unsupervised learning models provide valuable insights that would be difficult to uncover with traditional methods. These insights can then be used for various applications, such as data exploration, pattern recognition, and data preprocessing tasks.

Evaluation in unsupervised learning presents a unique challenge compared to supervised learning. In the absence of pre-defined labels, there is no single "correct" answer for the model's outputs. Therefore, evaluation metrics in unsupervised learning tend to be more nuanced and focus on aspects like the interpretability of the results, the model's ability to capture meaningful patterns, and the overall effectiveness in achieving the desired outcome.

2.4 Theoretical background of ML models

2.4.1 Extreme Gradient Boosting (XGBoost)

XGBoost, an innovative and scalable tree boosting system, has emerged as a powerful tool in the realm of machine learning. Developed by Tianqi Chen and Carlos Guestrin from the University of Washington, XGBoost has garnered widespread adoption among data scientists due to its ability to deliver state-of-the-art across various machine learning challenges its ability to deliver state-of-the-art results across various machine learning challenges.

At the core of XGBoost lies the concept of gradient tree boosting, a technique that performs additive optimization in functional space. This method involves the construction of an ensemble of decision trees, where each subsequent tree is trained to correct the errors made by the previous ones. The final prediction is obtained by summing the predictions of all individual trees. Mathematically, the objective function of XGBoost can be represented as:

$$\text{regularized objective} = \sum_{i} l(\hat{y}_i - y_i) + \sum_{k} \Omega(f_k)$$

Here, *l* represents the loss function measuring the difference between the true label y_i and the predicted label $\hat{y}_i$, while Ω is the regularization term that penalizes complex models to prevent overfitting.

For a given dataset the model uses K additive functions to predict the output. Thus,

$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)\,, f_k \in \mathcal{F},$

where $\mathcal{F}$ is the space of regression trees [27].

The following pseudocode (Figure 6) shows the XGBoost algorithm.
The XGBoost algorithm uses a function called Build-Tree to build the next regression tree in the ensemble. Built-Tree will also need another greedy function which evaluates every possible split at a given node and returns the split with the highest gain.

Algorithm XGBoost$(I, L, K, \eta, \lambda, \gamma)$

Input:

I: a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ with feature dimension m
$L(y, \hat{y})$: a differentiable loss function
K: number of boosting iterations
η: the learning rate
λ: regularization coefficient
γ: minimum loss reduction required for a split

1: Initialize the model with a constant value:

$$F_0(\mathbf{x}) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$$

2: **for** $k = 1$ to K **do**:
3: /* Compute the sum of gradients and Hessians of all the samples */
4: $G \leftarrow \sum_{i \in I} \frac{\partial L(y_i, F_{k-1}(\mathbf{x}_i))}{\partial F_{k-1}(\mathbf{x}_i)}$
5: $H \leftarrow \sum_{i \in I} \frac{\partial^2 L(y_i, F_{k-1}(\mathbf{x}_i))}{\partial F_{k-1}(\mathbf{x}_i)^2}$
6: $T \leftarrow$ Build-Tree(I, G, H)
7: Let the terminal regions of T be R_j for $j = 1, ..., J_k$
8: For $j = 1, ..., J_k$ compute:

$$w_j = -\frac{G_j}{H_j + \lambda}$$

where G_j and H_j are the sum of gradients and Hessians at leaf nod
9: Update the model:

$$F_k(\mathbf{x}) = F_{k-1}(\mathbf{x}) + \eta \sum_{j=1}^{J_k} w_j \mathbb{1}\{\mathbf{x} \in R_j\}$$

10: Output the final ensemble $F_K(\mathbf{x})$

Figure 6: The XGBoost algorithm pseudocode [27].

2.4.2 Decision Trees

Decision trees (DTs), lauded for their interpretability and efficiency. However, their very strength – the ability to capture intricate patterns in data – can lead to overfitting if left unchecked. This section explores two crucial concepts in DT construction: stopping criteria and impurity measures. These elements work in tandem to ensure the model achieves a balance between purity and complexity, ultimately enhancing its generalisation performance [28].

Imagine building a DT for fruit classification. You begin by splitting the data based on colour (red vs. non-red). This effectively separates apples and cherries from oranges and bananas. Further splitting is possible, considering size, texture, etc. However, at what point does additional splitting become detrimental?

Stopping criteria address this very question. They act as safeguards to prevent the DT from becoming overly complex and memorizing the training data too precisely. This memorization, known as overfitting, hinders the model's ability to perform well on unseen data. Here are some common stopping criteria employed in DT learning:

- Maximum Tree Depth (MTD): This parameter sets a limit on the number of splits a DT can undergo, effectively controlling its depth. While deeper trees can capture more complex relationships, they are also more susceptible to overfitting. Setting an appropriate MTD helps to strike a balance between capturing intricate patterns and preventing excessive complexity.
- Minimum Samples per Leaf (MSPL): This criterion ensures a minimum number of data points in each leaf node. If a split results in a leaf with too few samples, it might not generalise well to unseen data. An appropriate MSPL value helps to prevent such scenarios.
- Minimum Impurity Decrease (MID): This criterion focuses on the concept of impurity, which will be discussed in detail in the following section. MID halts splitting if the decrease in impurity after a split falls below a certain threshold. This prevents unnecessary splits that don't significantly improve the model's ability to differentiate between classes.
- Maximum Number of Nodes (MNN): This criterion directly controls the complexity of the DT by limiting the total number of nodes it can have. Setting an appropriate MNN value can help to prevent the tree from becoming overly complex.

The selection of optimal stopping criteria often necessitates experimentation and evaluation using a validation set (data not used for training). By carefully choosing these criteria, we can guide the DT learning process towards creating a model that is both accurate and generalisable.

Impurity measures are metrics that quantify the level of "mixedness" within a node regarding the target variable (what the DT is trying to predict). The objective is to ask questions (through splits) that create the "purest" possible child nodes, where the data points within a node are more likely to belong to the same class. Lower impurity signifies a more homogenous node, while higher impurity indicates a greater mix of

classes. In the following mathematical equations, we denote impurity of a node N as i(N). Also, $P(w_j)$ is the fraction of patterns at a node N that are in category w_j. If all the patterns are of the same category, the impurity is 0; otherwise, it has a positive value with the greatest value occurring when the different classes are equally likely.

Some commonly used impurity measures include:

- **Gini Impurity (Classification):** This measure calculates the probability of misclassifying a randomly chosen data point if it were labelled according to the class distribution within the node. A high Gini impurity indicates a diverse mix of classes, while a value of zero implies perfect separation (all data points belong to the same class). During DT construction, the algorithm aims to select features and split points that minimise Gini impurity, leading to purer child nodes.

$$i(N) = -\sum_{j} P(w_j) \log_2 P(w_j)$$

- **Entropy Impurity (Classification):** This measure reflects the average amount of information needed to classify a data point in the node. Higher entropy signifies greater uncertainty or randomness in class labels within the node. Similar to Gini impurity, the DT learning process seeks to minimise entropy by choosing splits that create child nodes with lower entropy values.

$$i(N) = \sum_{i \neq j} P(w_i) P(w_j) = 1 - \sum_{j} P^2(w_j)$$

- **Mean Squared Error (Regression):** This measure is employed in regression tasks and calculates the average squared difference between the predicted values and the actual target values within a node. A lower mean squared error indicates a closer fit between the predictions and the actual data points. The DT learning process, in the context of regression, aims to select features and split points that minimise the mean squared error, leading to child nodes with improved prediction accuracy.

  By incorporating these impurity measures into the DT learning process, we guide the model towards creating the purest possible child nodes, ultimately resulting in a more accurate model

  Stopping criteria and impurity measures play a crucial role in achieving optimal DT performance. Stopping criteria prevent excessive complexity, while impurity measures steer the tree towards informative splits that effectively separate the data. Finding the right balance between these two forces is essential for building a DT that performs well on unseen data. By understanding these concepts, researchers and practitioners can leverage DTs effectively for various classification and regression tasks. The simplest Decision Tree algorithm is Iterative Dichotomiser 3 (ID3) and the pseudocode can be found in Figure 7.

Algorithm ID3

Require: feature_vectors, labels, attributes

```
R is a tree node
if all labels are the same then                                   ▷ Base case 1
    R − label =labels(1)
else if length(attributes)== 0 then                               ▷ Base case 2
    R − label =mode(labels)
else
    A =attribute in the feature vector that maximizes information gain
    Remove A from attributes
    R − attribute = A
    for each value v of A do
        Add branch B to R that corresponds with A = v.
        split_feature_vectors = subset of feature_vectors where A = v
        if length(split_feature_vectors)== 0 then                 ▷ Base case 3
            Child − label =mode(labels)                           ▷ create a child node
        else
            split_labels are the labels corresponding with split_feature_vectors
            Child = ID3(split_feature_vectors, split_labels, attributes)
        end if
        add Child as the node at the end of B.
    end for
end if
Return R
```

Figure 7: Pseudocode for the ID3 Decision Tree [29].

2.4.3 Random Forest

At its core, a Random Forest is an ensemble method. Ensemble methods combine predictions from a collection of weaker learners to create a more robust and accurate final prediction. In the case of, Random Forests, the individual learners are decision trees. By leveraging the collective intelligence of numerous decision trees, Random Forests can achieve superior performance compared to a single decision tree [30].

The construction of a Random Forest involves several key steps:

1. **Decision Trees:** The forest is comprised of a collection of individual decision trees. Each tree is built using a subset of the training data, typically created using a technique called bootstrap aggregating (bagging). Bagging involves sampling the training data with replacement, resulting in some data points being included in the subset multiple times while others are omitted entirely. This injects randomness into the model creation process.

2. **Random Feature Selection:** To further enhance diversity and prevent overfitting, a random subset of features is considered for splitting at each node within a tree. This reduces the correlation between trees and helps the forest to capture a broader range of patterns within the data.
3. **Tree Growing:** Each tree in the forest is grown independently until a stopping criterion is met. Common stopping criteria include limitations on the maximum depth of the tree or a minimum number of data points required in a leaf node.
4. **Prediction:** For classification tasks, the final prediction of the , Random Forest is determined by a majority vote of the individual trees. Each tree casts a vote for the class it predicts, and the class with the most votes becomes the final prediction. For regression tasks, the final prediction is typically the average of the predictions from all trees in the forest.

The provided pseudocode (Figure 8) offers a concise representation of the Random Forest algorithm. The RandomForest function iterates through a loop, building B individual trees using the RandomizedTreeLearn function. This function constructs a single decision tree by selecting a random subset of features (f) at each node for splitting the data. The BootstrapSample function generates a bootstrap sample by randomly sampling the training data with replacement. By incorporating these elements of randomness, Random Forests are able to achieve superior performance and generalization compared to a single decision tree.

Algorithm Random Forest

Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, features F, and number of trees in forest B.

```
function RandomForest(S, F)
    H ← ∅
    for i ∈ 1, ..., B do
        S^(i) ← A bootstrap sample from S
        h_i ← RandomizedTreeLearn(S^(i), F)
        H ← H ∪ {h_i}
    end for
    return H
end function
function RandomizedTreeLearn(S, F)
    At each node:
        f ← very small subset of F
        Split on best feature in f
    return The learned tree
end function
```

Figure 8: Random Forest pseudocode [31].

2.4.4 Support Vector Machines

In the realm of machine learning, Support Vector Machines (SVMs) have emerged as a prominent algorithm for classification tasks [32][33]. Their effectiveness lies in identifying an optimal hyperplane within a high-dimensional feature space. This hyperplane strategically separates data points belonging to distinct classes, aiming to achieve the maximum margin – the distance between the hyperplane and the closest data points from either class. By maximising the margin, SVMs strive to enhance the model's robustness and generalisation performance on unseen data. Typically, this mapping to the higher dimension space would cost computationally. But fortunately, SVMs do not need to calculate this higher dimension and only the formula of the dot product is needed. Hence, instead of a complexity of $O(n^2)$ we have a complexity of O(n).

Mathematical Notation:

- $\boldsymbol{x_i} \in \boldsymbol{R^d}$: This represents an n-dimensional feature vector, encoding the characteristics of the i-th data point.
- $\boldsymbol{y_i}$: This denotes the class label associated with the i-th data point. In binary classification, +1 signifies class 1, and -1 signifies class 2.
- $\boldsymbol{w} \in \boldsymbol{R^d}$: This represents the weight vector, a crucial element with a direction normal (perpendicular) to the hyperplane. It dictates the direction of separation within the feature space.
- $\boldsymbol{b} \in \boldsymbol{R}$: This signifies the bias term, influencing the position of the hyperplane relative to the origin in the feature space.
- $(\boldsymbol{w^T x_i} + \boldsymbol{b})$: This represents the decision function. Its value indicates on which side of the hyperplane a particular data point lies.
- $\boldsymbol{K(x_i, x_j)}$: This notation (applicable in specific scenarios) represents a kernel function.

The decision boundary in SVM is established by a hyperplane, mathematically expressed as:

$$\boldsymbol{x}^T \boldsymbol{w} + b = 0$$

This equation defines a plane within a d-dimensional space. Data points are segregated based on their class labels on opposite sides of this hyperplane. The weight vector (w) dictates the hyperplane's orientation, while the bias term (b) influences its placement relative to the origin.

In the context of SVMs, the margin signifies the separation between the hyperplane and the closest data points from each class, also known as support vectors. The margin can be understood as the distance between the two support vectors. Mathematically, the margin can be expressed as:

$$margin\ (m) = \frac{2}{||\boldsymbol{w}||}$$

Here, ||**w**|| represents the Euclidean norm (magnitude) of the weight vector. A larger weight vector magnitude corresponds to a wider margin, indicating a clearer separation between the hyperplane and the support vectors.

In Figure 9, one can see an image on how SVM finds the hyperplane. In the figure, m is the margin described earlier. Also, the variable ρ represents the distance between the hyperplane and the origin O. This distance can be calculated as:

$$\rho = \frac{|b|}{||w||}$$

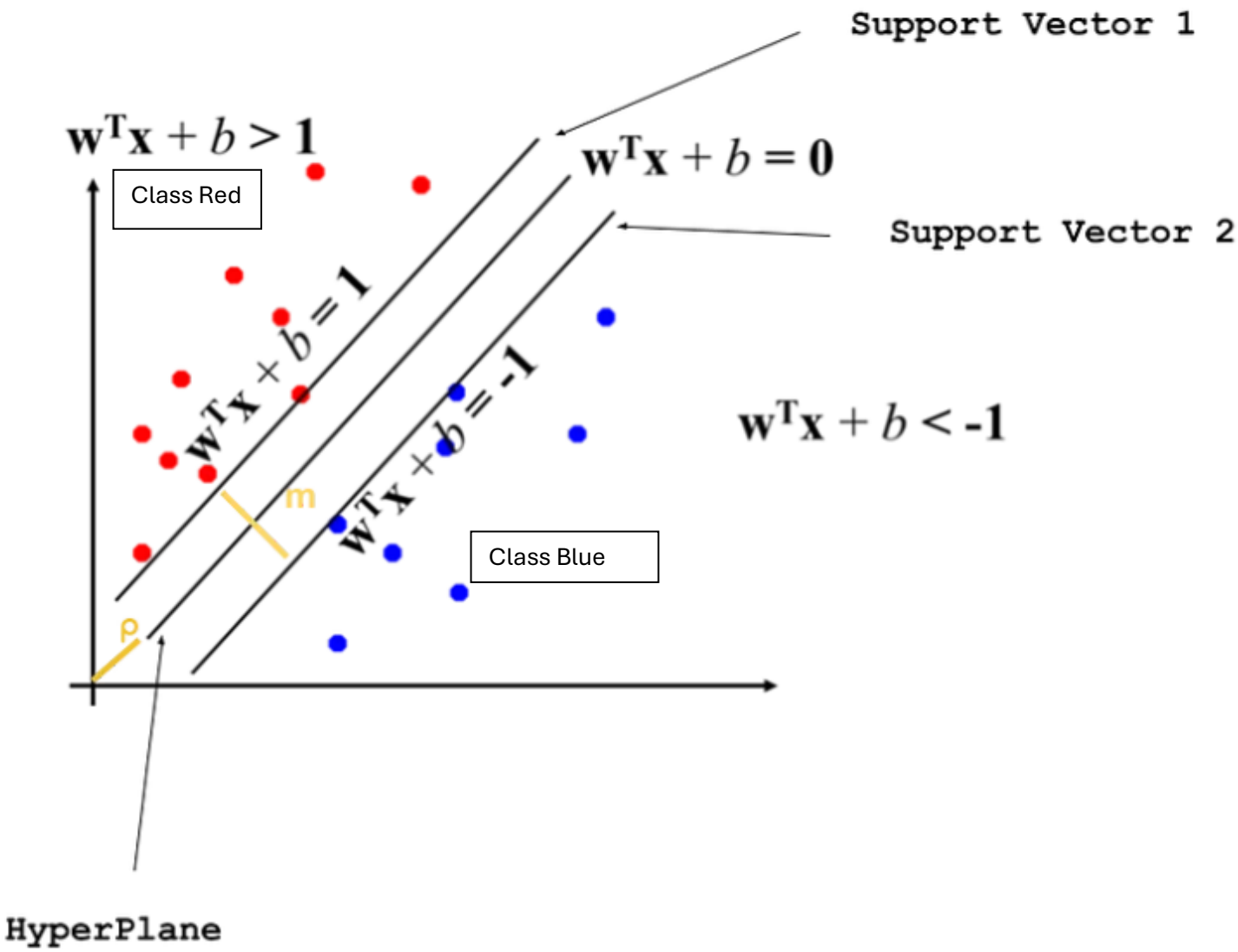


Figure 9: SVM hyperplane [33].

The Optimisation Problem

The core objective of SVM lies in finding the hyperplane that maximises the margin while simultaneously minimising classification errors. This translates into a mathematical optimisation problem:

minimise $\frac{1}{2}\,\boldsymbol{w}^T\boldsymbol{w} = \frac{1}{2}\,||\boldsymbol{w}||^2$ (objective function)

subject to $y_i(\boldsymbol{x}_i^T\boldsymbol{w} + b) \geq 1,\ or\ 1 - y_i(\boldsymbol{x}_i^T\boldsymbol{w} + b) \leq 0,\ (i = 1,\cdots,m)$

This quadratic problem can be solved by the Lagrange multipliers method to minimise the following:

$$L_p(\boldsymbol{w}, b, \alpha) = \frac{1}{2}\,||\boldsymbol{w}||^2 + \sum_{i=1}^{m} \alpha_i \left(1 - y_i\left(\boldsymbol{x}_i^T\boldsymbol{w} + b\right)\right)$$

with respect to **w**, b and the Lagrange coefficients $\alpha_i \geq 0$ (i = 1, $\cdots$, α_m). Letting the partial derivatives of L_p with respect to W and b equal to zero, leads us to the dual problem of the above problem. This approach focuses on maximising a function of the Lagrange multipliers (α_i) while adhering to certain conditions derived from the original problem.

This involves introducing Lagrange multipliers and maximising a function over them. The dual formulation simplifies the problem and avoids explicit computations with high-dimensional data points, especially when using kernel functions (explained later).

The Kernel Trick: Addressing Non-Linear Separability

For data that are not linearly separable in the original feature space, SVM employs the kernel trick. A kernel function $\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j})$ is a function that computes the inner product of two data points $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ in a higher-dimensional feature space, but without explicitly mapping the data points to that space. This offers a computationally efficient way to find non-linear decision boundaries in the original space. The kernel function essentially operates on the original data points and transforms them implicitly into a higher-dimensional space where they become linearly separable. This allows SVM to leverage the power of non-linear models while maintaining computational efficiency.

Here are some commonly used kernel functions:

- **Linear Kernel:** $\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) = x_i^T\ x_j$ (This is the simplest kernel, essentially performing a dot product in the original space. It is useful when the data is already linearly separable in the original features.)
- **Polynomial Kernel:** $\boldsymbol{K}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(x_i^T x_j + c\right)^d$. This kernel maps the data to a higher-dimensional polynomial feature space. By allowing for polynomial interactions between features, it can handle more complex relationships between features in the original space.

- **Radial Basis Function (RBF) Kernel:** $\boldsymbol{K(x_i, x_j)} = exp\left(-\frac{\left||x_i - x_j|\right|^2}{(2\sigma^2)}\right)$. This kernel projects the data points into a high-dimensional space using a radial basis function. The parameter σ controls the smoothness of the decision boundary. A smaller σ leads to a more complex and potentially overfitting decision boundary, while a larger σ creates a smoother but potentially less accurate boundary.
- **Sigmoid Kernel:** $\boldsymbol{K(x_i, x_j)} = tanh\left(\alpha\, x_i^T x_j + \ c\right)$. This kernel is less commonly used due to potential numerical instability issues. It performs element-wise non-linear transformations on the dot product in the original space.

By selecting an appropriate kernel function, SVMs can effectively handle non-linear data distributions, making them a versatile tool for various classification tasks.

In conclusion, SVMs provide a robust and powerful approach to classification problems. Their foundation lies in maximising the margin between classes in a high-dimensional feature space. The mathematical framework, including the optimisation problem and the kernel trick, empowers SVMs to handle both linearly separable and non-linearly separable data. This, combined with their ability to control model complexity through regularisation, makes SVMs a valuable tool in the machine learning practitioner's arsenal.

2.4.6 K Nearest Neighbours

The K-Nearest Neighbours (K-NN) algorithm is a well-established non-parametric classification technique widely used in machine learning [34][35]. Its popularity stems from its inherent simplicity and effectiveness across diverse applications, making it a prevalent choice for pattern recognition tasks. The core principle of K-NN revolves around classifying a new data point by identifying its closest neighbours in the feature space, determined by a distance metric. Subsequently, the class label is assigned based on the majority vote among those nearest neighbours.

K-NN relies on a fundamental mathematical framework to perform classification. This framework involves representing data points, defining a distance metric, identifying nearest neighbours, and employing a majority vote strategy.

- **Data Representation:**
    - A new data point is represented as a vector in the n-dimensional space, denoted by $\boldsymbol{x} \in \boldsymbol{R^n}$. Each element in the vector corresponds to a feature of the data point.
    - The training dataset, denoted by $\boldsymbol{T} = \{\boldsymbol{x_i} \in \boldsymbol{R^n} \mid \boldsymbol{i} = \boldsymbol{1, 2, \ldots, m}\}$, is a collection of m data points, each represented by an n-dimensional feature vector.
    - Every data point in the training set is associated with a class label, denoted by $\boldsymbol{y_i} \in \boldsymbol{C}$. Here, C represents the set of possible class labels.

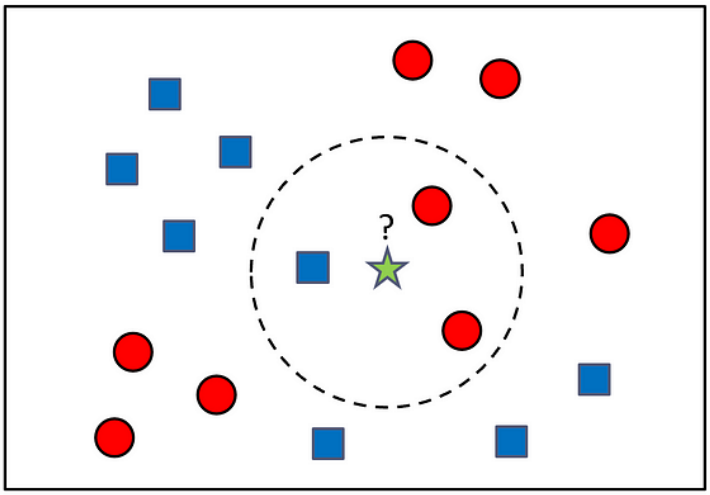


Figure 10: An example of a dataset. Using k-NN the point with the star symbol will be classified as a red circle, since this is the majority class among the 3 neighbours [36].

A crucial element in K-NN is the distance metric, denoted by **d(x, y)**. This function calculates the distance (similarity or dissimilarity) between two data points, x and y, in the n-dimensional feature space. The most common choice is the Euclidean distance, which calculates the straight-line distance between the points. The selection of an appropriate distance metric is important, as it affects how similarity between data points is measured within the context of your specific problem domain.

The distance can be calculated as:

Euclidean distance: This is the most commonly used distance measure, and it is limited to real-valued vectors. Using the formula below, it measures a straight line between the query point and the other point being measured.

$$d(x, y) = \sqrt{\sum_i (y_i - x_i)^2}$$

Manhattan distance: This is also another popular distance metric, which measures the absolute value between two points.

$$d(x, y) = \sum_i |x_i - y_i|$$

Minkowski distance: This distance measure is the generalised form of Euclidean and Manhattan distance metrics. The parameter, p, in the formula below, allows for the

creation of other distance metrics. Euclidean distance is represented by this formula when p is equal to two, and Manhattan distance is denoted with p equal to one.

$$d(x, y) = \left(\sum_i |x_i - y_i|\right)^{\frac{1}{p}}$$

Hamming distance: This technique is used typically used with Boolean or string vectors, identifying the points where the vectors do not match. As a result, it has also been referred to as the overlap metric. This can be represented with the following formula:

$$d(x, y) = \sum_i |x_i - y_i|, \begin{cases} x = y, d = 0 \\ x \neq y, d \neq 1 \end{cases}$$

The objective is to find the K closest data points in the training set to the new data point (x). This process, known as nearest neighbour search, can be achieved using various techniques, depending on factors like dataset size and dimensionality.

After identifying the K nearest neighbours, denoted by **N_K(x)**, the class label prediction for x is determined through a majority vote:

1. For each class label (c ∈ C), the algorithm counts the occurrences of that label within the nearest neighbours (N_K(x)).
2. The class label with the highest count (majority) is assigned as the predicted class for the new data point (x).

Unlike some algorithms that explicitly define a decision boundary, K-NN's decision boundary is implicit. It is shaped by the distribution of data points and the chosen K value:

- A smaller K value results in a more intricate decision boundary that can adapt to complex data patterns. However, it might be more susceptible to noise in the data.
- A larger K value leads to a smoother decision boundary but might be less sensitive to local variations in the data distribution.

When incorporating K-NN it is essential to consider its strengths and limitations.

- **Curse of Dimensionality:** As the number of features (n) increases, distances between data points tend to become more similar, making it challenging to identify meaningful nearest neighbours. Feature selection or dimensionality reduction techniques can be explored to mitigate this issue, especially if it's relevant to your research.
- **Choice of K:** The optimal value of K significantly impacts K-NN's performance. Techniques like cross-validation can be employed to determine an appropriate K for your specific dataset and classification task.

Figure 11 depicts the pseudocode implementing a k-NN algorithm.

Algorithm The k-nearest neighbors classification algorithm

Input:

D: a set of training samples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$
k: the number of nearest neighbors
$d(\mathbf{x}, \mathbf{y})$: a distance metric
$\mathbf{x}$: a test sample

1: **for each** training sample $(\mathbf{x}_i, y_i) \in D$ **do**
2: Compute $d(\mathbf{x}, \mathbf{x}_i)$, the distance between $\mathbf{x}$ and $\mathbf{x}_i$
3: Let $N \subseteq D$ be the set of training samples with the k smallest distances $d(\mathbf{x}, \mathbf{x}_i)$
4: **return** the majority label of the samples in N

Figure 11: k-NN pseudocode algorithm [35].

2.4.7 Linear Regression

Linear regression [37] and logistic regression are two fundamental statistical methods widely used in machine learning for modelling relationships between variables. They offer valuable tools for understanding and predicting various phenomena.

Linear regression establishes a linear relationship between a dependent variable (y) and one or more independent variables (x). The core assumption is that the expected value of the dependent variable (y) can be expressed as a linear function of the independent variables (x). Mathematically, this relationship can be represented as:

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + \varepsilon$$

where:

- y(**x**) denotes the predicted value of the dependent variable for a given input vector (**x**).
- $\boldsymbol{w}^T$ represents the transpose of the weight vector (**w**), containing the coefficients for each independent variable.
- **x** represents the input vector containing the values of the independent variables.
- ε represents the residual error, which captures the difference between the actual value of y and the predicted value (y(**x**)).

The residual error (ε) is typically assumed to follow a normal distribution with a mean of zero and a constant variance. The model parameters are estimated through a process called least squares regression, which minimises the sum of squared errors between the actual and predicted values of y.

Linear regression is a cornerstone of statistical modelling, offering several advantages that make it a popular choice for various research endeavours.

- **Interpretability:** A key strength of linear regression lies in its interpretability. The weight coefficients (w) directly correspond to the impact of each independent variable on the dependent variable. This transparency allows researchers to gain a clear understanding of how changes in the independent variables influence the predicted outcome. By analysing the coefficients, researchers can identify the most influential variables and the direction of their effect (positive or negative). This interpretability is particularly valuable in studies where understanding the underlying relationships between variables is crucial.

- **Computational Efficiency:** Linear regression models are computationally efficient to train. This efficiency makes them well-suited for analysing large datasets, a common occurrence in many research fields. The training process is relatively fast, allowing researchers to quickly iterate and explore different model configurations. This computational efficiency is particularly advantageous when dealing with limited computational resources or when rapid analysis is required.

  However, it is crucial to understand its limitations to ensure its appropriate application.

- **Linearity Assumption:** A fundamental assumption in linear regression is the existence of a linear relationship between the dependent and independent variables. If the underlying relationship is non-linear, the model might not accurately capture the true association. This limitation can lead to misleading results and inaccurate predictions. Researchers need to carefully assess the data to ensure a linear relationship exists before applying linear regression.

- **Sensitivity to Outliers:** Linear regression models can be sensitive to outliers in the data. Outliers are data points that deviate significantly from the majority of the data. The presence of outliers can significantly influence the estimated parameters and affect the model's performance. Techniques for outlier detection and handling might be necessary to ensure the model's robustness and reliability.

By understanding both the strengths and limitations of linear regression, researchers can effectively utilise this technique to gain valuable insights from their data. It is a powerful tool, particularly when the assumptions are met, and interpretability is a priority. However, it is essential to consider the potential for non-linear relationships and the presence of outliers to ensure the model produces reliable results.

2.4.8 Logistic Regression

Logistic regression is a statistical method employed for classification tasks [37]. It builds upon the principles of linear regression but adapts them to model the probability of an outcome belonging to a specific category. In logistic regression, the dependent variable (y) is typically binary (e.g., 0 or 1), representing two distinct classes.

The core idea lies in using a linear function to model the log-odds of an observation belonging to the positive class (y = 1). This log-odds is then transformed using the sigmoid function, which maps any real number to a value between 0 and 1. The resulting value represents the predicted probability of an observation belonging to the positive class.

Mathematically, the relationship in logistic regression can be expressed as:

$$p(y = 1 \mid \boldsymbol{x}, w) = \sigma(w^T \boldsymbol{x})$$

where:

- p(y = 1 | **x**, w) represents the conditional probability of an observation belonging to the positive class (y = 1) given the input vector (**x**) and the model parameters (w).
- σ( ) denotes the sigmoid function, which transforms the linear combination of weights and features ($w^T \boldsymbol{x}$) into a probability value between 0 and 1.

Based on the predicted probability from the sigmoid function, a decision rule can be established to classify new observations. A common threshold of 0.5 is often used, where observations with a predicted probability greater than 0.5 are classified as belonging to the positive class, and others are classified as belonging to the negative class.

The interpretation of logistic regression coefficients differs from linear regression. While the coefficients still reflect the influence of each independent variable, their interpretation focuses on the direction and relative strength of the association with the log-odds of belonging to the positive class.

Logistic regression stands out as a cornerstone technique within machine learning, particularly for classification tasks. While it offers distinct advantages, it is crucial to understand its limitations for effective application.

Unlike linear regression, logistic regression is specifically designed to excel at classification problems. It thrives in predicting binary outcomes, making it a robust tool for tasks like identifying spam emails or classifying customers into risk categories (high-risk vs. low risk). This classification prowess stems from its ability to model the probability of an observation belonging to one of two classes.

However, logistic regression is not without limitations. A key constraint lies in its restriction to binary classification tasks. If a problem involves more than two classes (e.g., classifying an image as cat, dog, or bird), adaptations like multinomial logistic regression are necessary. Additionally, logistic regression assumes a certain level of separability between the classes in the feature space. Ideally, data points belonging

to each class should be clearly distinguishable. If the classes significantly overlap or the data is not linearly separable, logistic regression might not perform optimally in such scenarios.

Despite these limitations, logistic regression offers valuable insights even with its interpretable coefficients. While not as straightforward as interpreting coefficients in linear regression, they still reveal the relative importance of each feature in influencing the classification. Researchers can leverage this information to understand which features play a more significant role in distinguishing between the two classes. This interpretability becomes particularly valuable in studies where understanding the underlying relationships between variables is crucial.

In conclusion, logistic regression is a powerful tool for binary classification tasks, particularly when interpretability is desired. However, researchers need to consider the number of classes and the separability of data points to ensure the model's suitability for the specific problem at hand. By understanding both its strengths and limitations, researchers can effectively utilise logistic regression to gain valuable insights from their classification tasks.

2.4.9 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) represent a specialised architecture within the realm of deep learning, particularly adept at image recognition tasks. Unlike traditional neural networks characterised by extensive inter-layer connections, CNNs leverage a more localized approach. This focus on local information within an image significantly enhances their efficiency in processing image data [38].

Two fundamental concepts underpin the efficiency of CNNs:

- Limited Connectivity: A defining characteristic of CNNs is the restricted connectivity between neurons. Each neuron establishes connections only with a small region of neurons in the subsequent layer. This approach dramatically reduces the overall number of connections and, consequently, the total number of parameters within the network compared to fully connected architectures.
- Weight Sharing: CNNs employ filters, known as kernels, that traverse the image, identifying specific features such as edges or shapes. Crucially, these kernels are shared by multiple neurons, leading to a further reduction in the number of required parameters. The output from these filters generates a map, termed an activation map, that highlights the regions within the image that elicited a response from the filter. By incorporating a diverse set of kernels, CNNs progressively learn a variety of features, ultimately constructing intricate representations of the image.

The synergy between local connections and weight sharing offers substantial advantages. This approach significantly diminishes the number of parameters within the network, fostering both efficiency and a reduced susceptibility to overfitting. Consequently, CNNs achieve remarkable performance in image recognition tasks,

solidifying their position as powerful tools for applications such as object detection and classification.

Convolutional layers are the essential building blocks of CNNs, particularly adept at image recognition tasks. Similar to hidden layers in regular neural networks, they transform the input image into a higher-level representation. However, unlike fully connected layers that utilise connections between every neuron, convolutional layers leverage local connections for calculations, making them more efficient for processing image data.

Neurons in a convolutional layer connect only to a small region of the input image. This significantly reduces the number of connections and parameters compared to fully connected layers.

The core operation of a convolutional layer is the convolution, performed using filters called kernels. These kernels slide across the image, detecting specific features like edges or shapes. Each kernel acts as a feature extractor and importantly, shares its weights with all neurons in the layer, further reducing parameters.

The output of the convolution process is stored in activation maps. Each activation map highlights the image regions where a specific kernel detected its corresponding feature. By using a variety of kernels, CNNs can progressively learn and represent complex features within the image.

Several hyperparameters influence the size of the resulting activation maps:

- Kernel Size (N): The size of the kernel window, also known as the receptive field, determines the region of the input involved in the convolution operation.
- Stride (S): This parameter defines how many pixels the kernel moves after each convolution. A larger stride reduces the activation map size.
- Zero-Padding (P): Padding the image with zeros around the borders can be used to maintain the input dimensions during convolution.

Sharing weights across neurons significantly reduces the total number of parameters in the network, improving efficiency and reducing the risk of overfitting.

Local convolutions capture local image characteristics, which are often highly correlated. This allows kernels with shared weights to effectively identify patterns across various local regions in the image, while using different kernels helps extract diverse feature types.

Often inserted after convolutional layers, pooling layers aim to reduce the activation map dimensions while preserving key information. Pooling also introduces spatial invariance, making the model less sensitive to small shifts in features within the image. Max-pooling, a common technique, extracts the maximum value from a defined window within the activation map.

Activation functions are non-linear activation functions (like ReLU) are typically applied to the convolution outputs within the activation maps before feeding them to the next layer.

While not always used in modern CNN architectures, some traditional approaches add a fully connected layer between the final convolutional layers and the output layer. This layer can help model non-linear relationships between the extracted features. However, due to the increased number of parameters it introduces, fully connected layers can potentially lead to overfitting. As a result, alternative techniques like max-over-time pooling are being explored to replace the role of fully connected layers.

2.4.10 Long Short-Term Memory

Long Short-Term Memory (LSTM) networks, introduced by Hochreiter and Schmidhuber in 1997, address a significant limitation of standard Recurrent Neural Networks (RNNs): the inability to handle long-term dependencies in sequences. Unlike RNNs, LSTMs can learn and remember information over extended periods, making them particularly suitable for tasks involving sequential data like speech or video [39].

The core unit of an LSTM network is the memory block. This block consists of a cell that stores information and three gates:

Input Gate (i^t) that controls the flow of information into the cell. It takes the previous output $h^{(t-1)}$ and current input $x^{(t)}$ as input and uses a sigmoid function (σ) to generate values between 0 and 1. A value of 0 completely blocks information, while 1 allows all information to pass through.

$$i^t = \sigma(W^{(ix)} x^t + U^{ih} h^{t-1} + b^i)$$

Cell Input Layer (l^t) is similar to the input gate, it receives $h^{(t-1)}$ and $x^{(t)}$ but uses a hyperbolic tangent (tanh) activation function to generate values between -1 and 1, representing the candidate information for the cell state.

$$l^t = tanh(W^{lx} x^t + U^{lh} h^{t-1} + b^l)$$

Forget Gate (f^t), crucially determines what information from the past ($c^{(t-1)}$) should be remembered or forgotten. It takes $h^{(t-1)}$ and $x^{(t)}$ as input and uses a sigmoid function to generate values between 0 and 1. This value is then multiplied (Hadamard product) with the previous cell state ($c^{(t-1)}$) to update the new cell state (c^t). A value of 0 completely forgets the old information, while 1 retains it entirely.

$$f^t = \sigma(W^{fx} x^t + U^{fh} h^{t-1} + b^f)$$

Cell State (c^t), acts as the memory of the LSTM cell, storing information over longer periods. It's updated by combining the previous cell state ($c^{(t-1)}$) controlled by the forget gate (f^t) and the product of the current input gate (i^t) and cell input (l^t).

$$c^t = f^t \circ c^{t-1} + i^t \circ l^t$$

The output gate (o^t) controls the flow of information out of the cell. It takes $h^{(t-1)}$ and $x^{(t)}$ as input and uses a sigmoid function to generate a value (o^t) between 0 and 1. This value is then multiplied with the current cell state activated by a tanh function ($\tanh(c^t)$) to obtain the final output vector ($h^{(t)}$).

$$o^t = \sigma(W^{ox}x^t + U^{oh}h^{t-1} + b^o)$$

$$h^t = o^t \circ tanh(c^t)$$

By regulating the flow of information through these gates and the cell state, LSTMs can effectively learn long-term dependencies within sequential data, overcoming the limitations of traditional RNNs.

2.5 Theoretic approach of the optimisation techniques

In this section, different optimisation techniques that will be are discussed, which are used to optimise the ML models.

2.5.1 Select Features (SelectKBest)

Feature selection is a crucial step in ML, particularly when dealing with large datasets [40]. It involves identifying and selecting the most relevant features from the entire feature set. This not only improves the efficiency of ML algorithms but also helps to avoid overfitting, a common problem where a model performs well on training data but poorly on unseen data.

SelectKBest is a popular method for feature selection [40]. It is a filter-based approach that relies on statistical measures to score and rank features based on their relationship with the target variable (what you are trying to predict). SelectKBest offers several advantages. It is easy to use, computationally efficient, and works independently of any specific ML algorithm. It provides different scoring functions like chi-squared, mutual information, and F-value, allowing you to choose the one that best suits your data type (numerical or categorical) and the task at hand (regression or classification). By leveraging SelectKBest, it is possible to streamline the data analysis workflow and achieve better performance from ML models.

2.5.2 K-fold Cross-Validation

The quest for accurate and reliable predictions is paramount in ML. However, a significant hurdle stands in the way - overfitting. This phenomenon occurs when a ML model becomes overly fixated on the intricacies of the training data, losing its ability to generalise effectively to unseen data. Imagine a student who memorises every detail on a practice test but struggles with a slightly different version on the actual exam. That is essentially overfitting in action - the model excels at the specific data it trained on but fails when confronted with real-world scenarios.

The consequences of overfitting can be severe. A model that appears to perform exceptionally well on training data might produce inaccurate and misleading results when deployed in real-world applications. For instance, a spam filter trained on a specific dataset of spam emails might incorrectly classify legitimate emails as spam if it overfits to the training data's idiosyncrasies.

This is where the concept of data scarcity comes into play. Ideally, we would have an abundance of data to train and evaluate our models. With ample data, we could train multiple models with varying complexities or utilise a validation set - a separate dataset specifically used to assess a model's generalisability. The model performing best on the validation set, which hasn't been "seen" by the model during training, is then chosen for deployment.

However, in many practical scenarios, data is a precious commodity. We might not have the luxury of a dedicated validation set. This creates a dilemma: using a validation set with limited data can lead to overfitting to that very set, defeating its purpose. To circumvent this challenge, data scientists turn to a powerful technique called cross-validation.

Cross-validation is a strategic approach to effectively utilise limited data for both training and performance assessment. It essentially involves a clever partitioning of the available data into folds (groups). The model is then trained on a combination of these folds (usually S-1 folds), S being the number of groups, and evaluated on the remaining fold. This process is repeated for all possible choices of the held-out fold, ensuring a comprehensive evaluation across the entire dataset. This process is shown in Figure 12. Finally, the performance scores from each run are averaged to obtain a more robust and generalisable estimate of the model's accuracy [41].

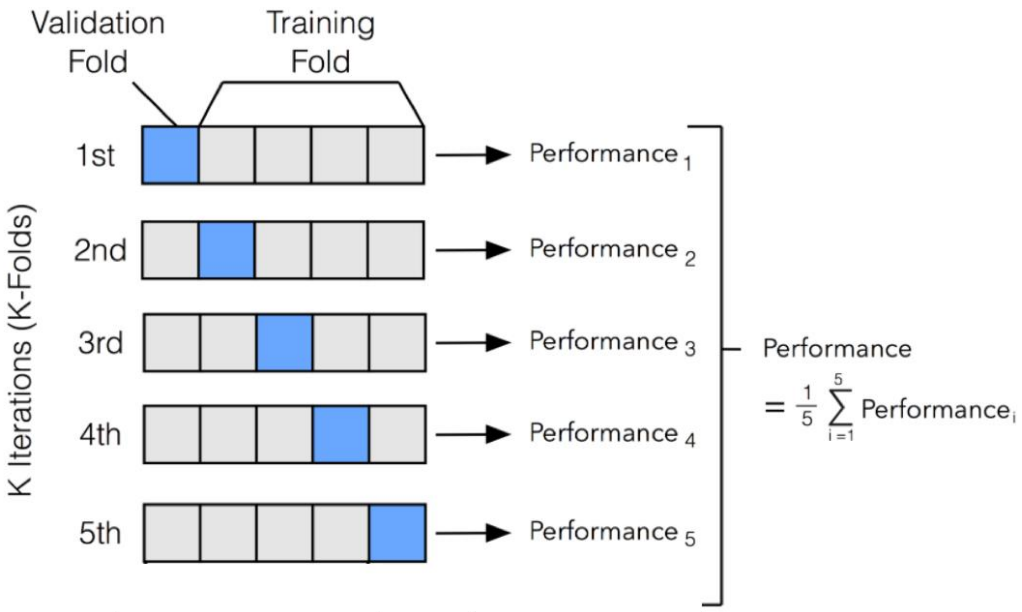


Figure 12: Cross validation process [41].

By leveraging cross-validation, we can effectively squeeze the most out of our limited data resources. It allows us to train models, assess their generalisability on unseen data within the same dataset, and ultimately select the best performer for real-world applications. In essence, cross-validation acts as a safeguard against overfitting, ensuring that our models can not only learn from the data but also adapt and perform well on new, unseen data encounters.

2.5.3 Hyperparameter tuning

Grid search is a popular technique for finding the optimal combination of hyperparameters for a machine learning model. It works by creating a grid - a set of discrete values - for each hyperparameter. The model is then trained and evaluated on every possible combination of these values within the grid. To assess performance, cross-validation is employed, which provides a more robust estimate of the model's generalisability.

The key advantage of grid search lies in its comprehensiveness. It explores every possible combination within the defined grid, ensuring that the optimal hyperparameter configuration is not missed. This exhaustive approach makes it a reliable method for identifying the absolute best combination.

However, grid search comes with a significant drawback - its computational cost. Evaluating every single combination can be extremely time-consuming, especially for models with many hyperparameters. The use of k-fold cross-validation further exacerbates this issue, as each combination requires k training steps.

Despite its slowness, grid search remains a valuable option when the goal is to find the absolute best hyperparameter configuration, especially if computational resources are not a major constraint.

2.6 Data preparation techniques

2.6.1 Simple Imputer

The presence of missing data, often represented by values like "NaN" or "None," can significantly impact the performance of ML models. These missing entries can introduce bias or hinder the model's ability to learn accurate relationships between features. To address this challenge, data imputation techniques are employed.

SimpleImputer works by replacing missing values with a chosen value based on a specific strategy. It provides various options, like using the mean or median for numerical data, or the most frequent value for categorical data. One can even specify a constant value to fill in the blanks. This flexibility allows the user to tailor the imputation approach to their data type and the nature of the missing values. However, one should be mindful of outliers when using the mean strategy, as they can skew the results.

By effectively addressing missing data with SimpleImputer, one can ensure that their ML models have a solid foundation of clean and complete data. This, in turn, leads to more accurate predictions and improved model performance.

2.6.2 Synthetic Minority Oversampling Technique

Class imbalance, where one class vastly outnumbers others in a dataset, can Lead to biased predictions in ML models. Imagine training a fraud detection model – if only a

tiny fraction of transactions is fraudulent, the model might simply classify everything as safe. The Synthetic Minority Oversampling Technique (SMOTE) tackles this issue.

SMOTE helps the model by creating synthetic samples for the rare, minority class. This lets the model learn the minority class's characteristics better, leading to more accurate detection. Additionally, SMOTE balances the class distribution, reducing bias towards the majority class. With a richer dataset and less bias, models trained using SMOTE become more robust and generalise better to unseen data.

While SMOTE is a powerful tool, it is important to remember that it increases the dataset size by creating artificial data points. Tuning parameters like the number of nearest neighbours and the oversampling rate is crucial. Combining SMOTE with under sampling the majority class can also be a good strategy for a more balanced approach.

2.6.3 One-Hot Encoding

Machine learning algorithms thrive on numerical data, utilising features represented by numbers to make predictions and uncover patterns within datasets [39]. However, real-world data often present a challenge in the form of categorical variables. These variables represent qualities or classifications using labels or text, such as "red," "blue," or "large," "medium," "small" [40]. While categorical variables hold valuable information, machine learning models struggle to directly process and learn from them due to their non-numerical nature. One-hot encoding emerges as a critical technique in this scenario, transforming categorical data into a numerical format that empowers machine learning models to understand and leverage the power of these features.

The significance of one-hot encoding lies in addressing two key challenges associated with incorporating categorical data into machine learning models:

- **Numerical Compatibility:** Many machine learning algorithms, from linear regression to complex neural networks, rely on features represented by numerical values for computations [39]. One-hot encoding bridges this gap by converting categorical data into numerical vectors. Consider a dataset with a "colour" variable containing "red," "green," and "blue." Directly assigning numerical values (e.g., red = 1, green = 2, blue = 3) to categories can be misleading. The model might interpret "blue" as inherently "better" due to its higher numerical value. One-hot encoding avoids this pitfall by creating a distinct numerical representation for each category.
- **Preserving Categorical Information:** Assigning numerical values disregards the inherent categorical nature of the data. Categories often represent distinct classifications without an inherent order. For instance, "red" is not inherently better or worse than "blue." One-hot encoding tackles this by creating a binary vector for each data point. The length of this vector matches the total number of categories. Each position in the vector represents a specific category. If the data point belongs to the corresponding category, the value at that position is set to 1, and 0 otherwise. This

essentially creates a new feature for each category, with the binary vector indicating category membership.

The one-hot encoding process itself is a two-step waltz:

1. **Integer Encoding:** Each unique category within a categorical variable is assigned a unique integer value. This initial step provides a numerical representation for each category while maintaining its distinct identity.
2. **Binary Vector Creation:** For each data point, a binary vector is created with a length equal to the total number of categories. Each position in the vector corresponds to a specific category. The value at a position is set to 1 if the data point belongs to the corresponding category, and 0 otherwise.

Consider the "colour" variable example again. After one-hot encoding, a data point with the value "red" would be transformed into a vector like (1, 0, 0), signifying membership in the "red" category and exclusion from "green" and "blue."

One-hot encoding offers several advantages for incorporating categorical data into machine learning models:

- **Improved Model Performance:** By providing a format compatible with machine learning algorithms, one-hot encoding can lead to more accurate model predictions. The model can now effectively learn the relationships between categories and other numerical features within the dataset [40].
- **Enhanced Data Interpretability:** The binary representation of categories allows for easier interpretation of model behaviour and feature importance. By analysing the weights assigned to features in a model, we can gain insights into which categories have a stronger influence on the model's predictions.
- **Effective Handling of Unordered Data:** One-hot encoding is particularly useful for categorical data where the order of categories has no inherent meaning. Clothing size (small, medium, large) is a perfect example. Assigning a higher numerical value to "large" doesn't imply superiority over "small." One-hot encoding preserves this information by creating separate features for each category.

However, it is crucial to acknowledge the considerations and potential alternatives to one-hot encoding:

- **Increased Data Dimensionality:** One-hot encoding can significantly increase the number of features in the dataset, especially when dealing with a large number of categories. This can impact model training time and computational resources [42].
- **Potential for Sparse Data:** When dealing with a vast number of categories, the resulting binary vectors can become sparse, with many zeros. This can lead to inefficiencies in some algorithms.

Here are some alternative encoding techniques to consider depending on the specific characteristics of the dataset:

- **Label Encoding:** Assigns a unique integer to each category. However, this method assumes an inherent order among categories, which may not always be true.
- **Frequency Encoding:** Replaces categories with their frequency counts within the dataset. This can be helpful for capturing the relative importance of categories but may not be suitable for all algorithms [42].
- **Embedding Techniques:** Advanced methods like word embeddings can learn low-dimensional vector representations of categories, capturing semantic relationships between them. This can be particularly useful for large numbers of categories or situations where categories have inherent relationships (e.g., colours on a spectrum). However, embedding techniques often require more complex models and computational resources [43].

The choice of encoding technique depends on the specific characteristics of the dataset, the number of categories, and the ML model used. Here are some factors to consider when making this decision:

- **Number of Categories:** If there is a small number of categories (less than 10), one-hot encoding is generally a good choice. However, with a large number of categories, it can lead to high dimensionality and sparse data. In such cases, consider alternative techniques like encoding with fewer dimensions or using embedding techniques.
- **Inherent Order in Categories:** If the order of categories has a meaningful interpretation (e.g., shirt sizes - small, medium, large), label encoding might be suitable. However, for unordered categories (e.g., colours), one-hot encoding is preferred.
- **Model Complexity:** Some ML models, like decision trees, can handle categorical data natively. However, most models benefit from some form of encoding.

In conclusion, one-hot encoding is an essential tool in the ML toolbox, enabling us to unlock the power of categorical data. By transforming categorical features into a format that preserves category information and aligns with ML algorithms, one-hot encoding paves the way for more accurate and interpretable models.

2.7 Evaluation Metrics

The success of any ML project hinges on building a model that excels at making predictions on unseen data. To achieve this, reliable methods are needed to both train and assess the model's performance using the available training data. Evaluating a machine learning algorithm is a fundamental step in any such project, and this section explores various metrics that estimate a model's effectiveness.

2.7.1 Confusion Matrix

A confusion matrix is a powerful tool for assessing how well a classification model performs. It is essentially a table with two dimensions: rows represent the actual classifications (the true labels), and columns represent the model's predictions. This layout allows us to see how often the model's predictions matched the true classifications.

For consistency, this thesis uses a convention where the model's predictions appear in the columns, and the true classifications are displayed in the rows. Importantly, the order of the categories (classes) is the same in both rows and columns. This means the correctly classified instances fall on the main diagonal, running from the top left corner to the bottom right corner. These diagonal values represent the number of times the model's predictions agreed with the true classifications.

An example of confusion matrix can be found in Figure 13. In this figure, true positive (TP) captures the number of positive data points the model correctly classified as positive. These are the ones the model got right – truly positive instances identified as positive. False Positive (FP), or type 1 error, refers to negative data points that the model mistakenly classified as positive. These are the model's errors – negative instances flagged as positive. True Negative (TN) represents negative data points the model correctly classified as negative. The model identified these negative instances accurately. False Negative (FN), or type 2 error, describes positive data points the model incorrectly classified as negative. The model missed these positive instances by classifying them as negative.

		Predicted Values	
	Class	Positive	Negative
Actual Values	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Figure 13: Confusion matrix for multi-class dataset [44].

2.7.2 Precision

Precision measures the accuracy of the model's positive predictions. It essentially asks: "Out of all the instances the model classified as positive, how many were truly positive?".

To calculate precision, we divide the number of True Positives (correctly classified positive instances) by the total number of elements the model predicted as positive (the sum of all positive values in a specific column of the confusion matrix), as shown below.

$$precision = \frac{TP}{TP + FP}$$

In simpler terms, precision helps us understand how good the model is at identifying actual positive cases from the pool of instances it flagged as positive.

2.7.3 Recall

Recall focuses on the model's ability to identify all the actual positive cases in the data. It asks: "Out of all the instances that are truly positive in the data, how many did the model correctly classify as positive?".

To calculate recall, we divide the number of True Positives (correctly classified positive instances) by the total number of actual positive cases in the data (the sum of all positive values in a specific row of the confusion matrix), as shown below.

$$recall = \frac{TP}{TP + FN}$$

In simpler terms, recall helps us understand how well the model captures all the positive instances, ensuring it doesn't miss any important cases. Another way to think about it is measuring the model's ability to be "complete" in finding positive cases.

2.7.4 Accuracy

Accuracy is a popular metric used to evaluate how well a classification model performs, especially in multi-class problems. It is calculated directly from the confusion matrix.

Accuracy is the sum of correctly classified instances (True Positives and True Negatives) on the main diagonal, divided by the total number of instances in the dataset (including both correctly and incorrectly classified ones). In simpler terms, if a data point is randomly picked and predicted its class, accuracy depicts the probability of the model being right.

To calculate the accuracy one can, utilise the formula:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

One advantage of accuracy is its simplicity. It gives you an overall idea of how well the model performs across all classes in the dataset. Each data point contributes equally to the accuracy score.

However, accuracy has limitations. In datasets with imbalanced classes (where some classes have many more data points than others), accuracy can be misleading. The model could be performing poorly on less frequent classes, but this might be masked by its good performance on the dominant class. Accuracy does not provide information on how well the model performs on individual classes.

2.7.5 F1-Score

The F1-Score [45] emerges as a powerful tool for evaluating a classification model's performance, leveraging insights from both precision and recall. It takes centre stage after we have analysed the confusion matrix, a grid summarising classification results.

The F1-Score is calculated using the harmonic mean of precision and recall. This harmonic mean essentially rewards models that achieve similar levels of precision (correctly identifying positive cases) and recall (capturing all true positive cases). The formula itself is a weighted average, with F1 reaching its best value (1) when both precision and recall are perfect, and its worst value (0) when either metric is very low.

This metric can be calculated as:

$$F1 - score = \frac{2}{precision^{-1} + recall^{-1}} = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall}\right)$$

The F1-Score can be applied to both binary (two-class) and multi-class problems.

- **Binary F1-Score:** In binary classification, it considers only the positive class (ignoring True Negatives). The confusion matrix guides the calculation of precision and recall for the positive class, which are then plugged into the F1-Score formula.
- **Multi-Class F1-Score:** Things get a bit more complex with multiple classes. Here, we have two main approaches: Macro F1-Score and Micro F1-Score.
    - **Macro F1-Score:** This approach treats all classes equally. It calculates average precision and recall across all classes and then combines them using the harmonic mean to get the Macro F1-Score. This method is beneficial when you want to ensure the model performs well on all classes, regardless of their size.
    - **Micro F1-Score:** This approach considers all data points together, disregarding class boundaries. It essentially calculates the overall accuracy (fraction of correctly classified instances) as the Micro F1-Score. This method is useful when you prioritize overall model performance and might not be as concerned about individual class imbalances.

The F1-Score heavily penalises models with very low precision or recall. This characteristic makes it particularly useful for identifying weaknesses in a model's prediction capabilities. A high F1-Score generally indicates a well-performing model.

2.7.6 Micro-Averaging

In the realm of multi-class classification, where models juggle multiple categories, evaluating performance can get cluttered with a barrage of metrics for each class. Micro-averaging emerges as a technique to streamline this by condensing the information into a single, overall metric.

The core concept is simple: instead of analysing individual classes, we consider all data points as a whole. Micro-averaging achieves this by:

1. **Combining Class Data:** It breaks down the barriers between classes and aggregates the total number of true positives (correctly classified positives), false positives (incorrectly classified positives), and false negatives (missed positives) across all classes.
2. **Calculating Overall Metrics:** With these combined counts, micro-averaging calculates precision (true positives divided by the sum of true and false positives) and recall (true positives divided by the sum of true positives and false negatives) for the entire dataset.

Imagine taking a multi-section exam. Micro-averaging would be like adding up all your correct and incorrect answers across all sections to calculate your overall score. This approach prioritises the model's overall accuracy in classifying data points, regardless of their specific class labels.

However, micro-averaging can be insensitive to class imbalances. If a dataset has one dominant class with significantly more data points than others, the model's performance on smaller classes might be masked by its performance on the dominant class.

Therefore, micro-averaging is a good choice when overall model accuracy is the primary concern and class imbalances are not a major issue.

2.7.7 Macro-Averaging

Micro-averaging simplifies things by focusing on the overall picture, but what if we want to ensure the model performs well on all classes, even the smaller ones? Here's where macro-averaging steps in.

Macro-averaging champions fairness by giving equal weight to each class, regardless of its size:

1. **Measure by Class:** It calculates precision and recall for each individual class, treating them like separate mini tests within the larger multi-class exam.

2. **Average Across Classes:** Finally, it takes the average of these individual class precisions and recalls to arrive at the final macro-averaged scores.

Macro-averaging is like averaging grades on a test with multiple sections, but where each section has the same weight, even if it has fewer questions. This approach ensures that the model's performance on smaller classes isn't overshadowed by the performance on larger ones.

Macro-averaging is a valuable tool when all classes are important, and ensuring good performance across the board is crucial. However, it might not be the best choice if overall model accuracy is your primary concern.

2.7.8 AUC - ROC Curve

Before delving into AUC-ROC, it is essential to establish a clear understanding of two fundamental metrics [45][46][47]:

- **True Positive Rate (TPR) or sensitivity or recall:** This metric quantifies the proportion of true positive instances (correctly classified positive cases) relative to all actual positive cases in the data. Mathematically, it is expressed as

$$sensitivity = \frac{TP}{TP + FN},$$

  where TP denotes true positives and FN denotes false.

- **True Negative Rate or specificity:** Specificity focuses on the model's ability to correctly classify negative instances. It is calculated as

$$specificity = \frac{TN}{TN + FP},$$

  where TN represents true negatives and FP represents false positives.

Both Sensitivity and Specificity range between 0 and 1. A value closer to 1 signifies better performance.

Another metric that we should mention is **False Positive Rate (FPR)** that is equal to

$$FPR = 1 - specificity$$

The Receiver Operating Characteristic (ROC) curve serves as a graphical representation of the model's performance across different classification thresholds. It plots the Sensitivity (y-axis) against 1-Specificity (False Positive Rate) on the x-axis. As the classification threshold is varied, the ROC curve depicts the trade-off between sensitivity and specificity. An example of a ROC curve can be seen in Figure 14.

In an ideal scenario, a perfect model would achieve a sensitivity of 1 (classifying all true positives correctly) and a specificity of 1 (classifying all true negatives correctly). This translates to an ROC curve that aligns with the upper left corner of the graph.

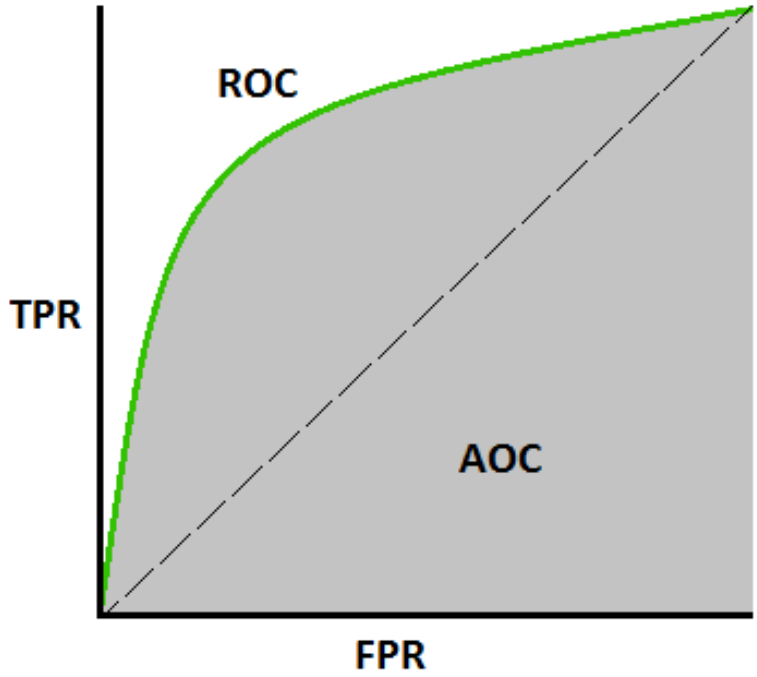


Figure 14: ROC-AUC curve [47]. *The long-dashed line in the middle corresponds to a random classifier. TPR denotes the True Positive Rate, while FPR the False Positive Rate. Roc is the Receiver Operating Characteristic curve and AUC the Area Under the ROC Curve*

While the ROC curve offers valuable insights, it can be cumbersome to summarise performance using a single value. AUC-ROC (Area Under the ROC Curve) addresses this challenge. AUC-ROC essentially quantifies the entire two-dimensional area encompassed by the ROC curve.

Interpreted probabilistically, AUC-ROC represents the likelihood that the model will rank a randomly chosen positive instance higher than a randomly chosen negative instance. Consequently, a higher AUC value indicates a greater ability for the model to distinguish between the positive and negative classes. The AUC-ROC metric ranges from 0 to 1, with 1 signifying perfect performance (where the ROC curve aligns perfectly with the upper left corner) and 0 indicating entirely random classification.

3. Materials and Methods

This chapter details the procedures for acquiring the dataset of Long Covid patients from the Long Covid Clinic of Evangelismos General Hospital and the rigorous screening process implemented to ensure accurate diagnoses. The process of data warehousing is also outlined, which includes the storage, organisation, and management of the collected data for efficient analysis. Finally, the analytical methods employed in the study are discussed along with the specific ML and statistical techniques used to analyse the data from the initial screening process in order to prioritise Long Covid patients.

3.1 Dataset

3.1.1 Data acquisition

The data used in this study was obtained from a private dataset compiled at the Long Covid Clinic of Evangelismos General Hospital. The dataset is derived from a novel study by Dr. Katsaounou. The study is the first to holistically analyse Long-Covid. Patients attending the clinic underwent a comprehensive screening process to determine their eligibility for the study. This screening involved a physician evaluation and a series of questionnaires. Only patients diagnosed with Long Covid by the medical team were included in the dataset. The dataset includes 174 Long Covid patient data.

Following the initial screening, patients diagnosed with Long Covid participated in a series of specialist examinations. These included consultations with a pneumonologist, a cardiologist, and a psychologist. During each examination, relevant medical data was collected and added to the dataset. Additionally, the medical team documented the most urgent treatment recommendation for each patient, categorised as pneumonology, cardiology, psychology, or none.

This data acquisition approach offers several advantages. Firstly, by focusing on patients diagnosed by a team of specialists at a dedicated Long Covid clinic, the dataset ensures a high degree of accuracy in Long Covid identification. Secondly, the inclusion of diverse medical data from various specialist examinations provides a comprehensive picture of the patients' Long Covid experience. Finally, documenting the most urgent treatment recommendation provides valuable insights into the multifaceted nature of Long Covid and the potential need for multi-specialty interventions.

3.1.2 Feature description

Given the objective of prioritising patients based on their initial screening, the dataset is limited to features from the initial screening process and the physician recommendation for the most urgent treatment specialty. This ensures that the prioritisation is based solely on information available at the time of the first visit, enabling timely and efficient patient triage. Our dataset is consisted of 88 features encompassing demographics, Covid-19 history, Long Covid symptoms, mental health status, lifestyle factors, quality of life measures and the label for each patient. The features and the target value are described below, to provide a comprehensive understanding of the dataset.

- **Demographic Information:**
    - **Gender:** Categorical variable indicating the patient's gender (0=Male, 1=Female).
    - **Age:** Numerical variable representing the patient's age in years.

- **Covid-19 History:**
    - **Disease Severity:** Categorical variable indicating the severity of the patient's Covid-19 illness (0=Mild, 1=Moderate, 2=Severe).
    - **Number of Covid-19 Episodes:** Numerical variable indicating the number of times the patient has contracted Covid-19.
    - **Hospitalisation:** Binary variable indicating whether the patient was hospitalised due to Covid-19 (0=No, 1=Yes).
    - **ICU Admission:** Binary variable indicating whether the patient required intensive care unit (ICU) admission due to Covid-19 complications (0=No, 1=Yes).
    - **Intubation:** Binary variable indicating whether the patient required mechanical ventilation (intubation) due to Covid-19 (0=No, 1=Yes).

- **Long Covid Symptoms:**
    - **High Oxygen Mixtures:** Binary variable indicating whether the patient required supplemental oxygen therapy (0=No, 1=Yes).
    - **Dyspnoea:** Binary variable indicating whether the patient experiences shortness of breath (0=No, 1=Yes).
    - **Nasal Congestion:** Binary variable indicating whether the patient experiences nasal congestion (0=No, 1=Yes).
    - **Cough:** Binary variable indicating whether the patient experiences a persistent cough (0=No, 1=Yes).
    - **Chest Pain:** Binary variable indicating whether the patient experiences chest pain (0=No, 1=Yes).

- **Precordial Pain:** Binary variable indicating whether the patient experiences precordial pain (0=No, 1=Yes).
- **Constrictive Precordial Pain:** Binary variable indicating whether the patient experiences constrictive precordial pain (0=No, 1=Yes).
- **Palpitations:** Binary variable indicating whether the patient experiences palpitations (0=No, 1=Yes).
- **Tachycardia:** Binary variable indicating whether the patient experiences a rapid heart rate (tachycardia) (0=No, 1=Yes).

Additional Symptoms:

- **Fatigue:** Binary variable indicating whether the patient experiences fatigue (0=No, 1=Yes).
- **Muscle Weakness:** Binary variable indicating whether the patient experiences muscle weakness (0=No, 1=Yes).
- **Paraesthesia:** Binary variable indicating whether the patient experiences numbness or tingling (paraesthesia) (0=No, 1=Yes).
- **Myoclonus:** Binary variable indicating whether the patient experiences involuntary muscle twitches (myoclonus) (0=No, 1=Yes).
- **Muscle Spasm:** Binary variable indicating whether the patient experiences muscle spasms (0=No, 1=Yes).
- **Tremor:** Binary variable indicating whether the patient experiences tremors (0=No, 1=Yes).

- **Mental Health Symptoms:**
  - **Arthralgias:** Binary variable indicating whether the patient experiences joint pain (0=No, 1=Yes).
  - **Headache:** Binary variable indicating whether the patient experiences headaches (0=No, 1=Yes).
  - **Dizziness:** Binary variable indicating whether the patient experiences dizziness (0=No, 1=Yes).
  - **Vertigo:** Binary variable indicating whether the patient experiences vertigo (0=No, 1=Yes).
  - **Sleep Disturbance:** Binary variable indicating whether the patient experiences sleep disturbances (0=No, 1=Yes).
  - **Concentration Difficulty:** Binary variable indicating whether the patient experiences difficulty concentrating (0=No, 1=Yes).
  - **Brain Fog:** Binary variable indicating whether the patient experiences brain fog (0=No, 1=Yes).

- **Memory Impairment:** Binary variable indicating whether the patient experiences memory impairment (0=No, 1=Yes).

- **Other Health Factors:**
  - **Anosmia:** Binary variable indicating whether the patient experiences loss of smell (0=No, 1=Yes).
  - **Ageusia:** Binary variable indicating whether the patient experiences loss of taste (0=No, 1=Yes).
  - **Weight Loss:** Binary variable indicating whether the patient has experienced weight loss (0=No, 1=Yes).

- **Physiological Measurements:**
  - **Fever:** Binary variable indicating whether the patient experiences a fever (0=No, 1=Yes).
  - **Sweating:** Binary variable indicating whether the patient experiences excessive sweating (0=No, 1=Yes).
  - **Hair Loss:** Binary variable indicating whether the patient experiences hair loss (0=No, 1=Yes).
  - **Rash:** Binary variable indicating whether the patient experiences a rash (0=No, 1=Yes).

- **Gastrointestinal Symptoms:**
  - **Dyspepsia:** Binary variable indicating whether the patient experiences indigestion (0=No, 1=Yes).
  - **Bowel Issues:** Binary variable indicating whether the patient experiences bowel problems (0=No, 1=Yes).

- **Ophthalmic Symptoms:**
  - **Ophthalmic Problems:** Binary variable indicating whether the patient experiences eye problems (0=No, 1=Yes).

- **Psychological Distress:**
  - **Anxiety:** Binary variable indicating whether the patient experiences anxiety (0=No, 1=Yes).
  - **Depression:** Binary variable indicating whether the patient experiences depression (0=No, 1=Yes).
  - **Fear:** Binary variable indicating whether the patient experiences fear (0=No, 1=Yes).

- **Nervousness:** Binary variable indicating whether the patient experiences nervousness (0=No, 1=Yes).

- **Vaccination Status:**
  - **Vaccination Before:** Binary variable indicating whether the patient received a Covid-19 vaccination prior to their initial Covid-19 illness (0=No, 1=Yes).
  - **Vaccination After:** Binary variable indicating whether the patient received a Covid-19 vaccination after their initial Covid-19 illness (0=No, 1=Yes).

- **Lifestyle Factors:**
  - **Smoking Status:** Categorical variable indicating the patient's smoking status (0=Never, 1=Current, 2=Ex-smoker).
  - **Height:** Numerical variable representing the patient's height in meters
  - **Weight:** Numerical variable representing the patient's weight in kilograms.
  - **BMI (Body Mass Index):** Categorical variable indicating the patient's Body Mass Index (BMI) category (0=Underweight, 1=Normal Weight, 2=Overweight, 3=Obese Class I, 4=Obese Class II, 5=Obese Class III, 6=Morbid Obesity).
- **Quality of Life Questionnaires:**
  - **SF-36:** This section includes various subscales from the Short Form-36 (SF-36) Health Survey, a widely used tool to assess health-related quality of life [48]. Each subscale score ranges from 0 to 100, with higher scores indicating better health.

- **Mental Health Questionnaires:**
  - **HADS:** This section includes scores from the Hospital Anxiety and Depression Scale (HADS), which measures anxiety and depression symptoms. Scores range from 0 to 21, with higher scores indicating more severe symptoms [49].

- **Physical Activity Questionnaires:**
  - **IPAQ:** This section includes various scores from the International Physical Activity Questionnaire (IPAQ), which assesses physical activity levels. Scores are categorised based on activity intensity and total metabolic equivalent (MET) score [50].

- **Other Questionnaires:**

- **EQ-5D:** This section includes scores from the EuroQol 5 Dimensions (EQ-5D), a generic measure of health-related quality of life. Scores are derived from five dimensions: mobility, self-care, usual activities, pain/discomfort, and anxiety/depression [51].
- **CAT and mMRC:** These sections include scores from the COPD Assessment Test (CAT) and the modified Medical Research Council (mMRC) dyspnoea scale, which assess dyspnoea (breathlessness) severity [52][53].
- **Fatigue Severity Scales:** These sections include scores from various fatigue severity scales, which measure the intensity and impact of fatigue. [54]
- **MoCa:** The Montreal Cognitive Assessment (MoCA) questionnaire is a brief in-office tool used by healthcare professionals to assess cognitive function. It can quickly detect mild cognitive impairment, a potential sign of conditions like Alzheimer's disease. The MoCA evaluates various areas like memory, attention, and language through tasks and questions. Scoring a maximum of 30, a score below 24 suggests a need for further evaluation [55].
- **ABC:** The Activities-Specific Balance Confidence (ABC) questionnaire is not related to cognition. Instead, it focuses on a person's confidence in maintaining balance during daily activities. Here, individuals rate their confidence level (from 0% to 100%) for specific actions like walking on uneven surfaces. This helps healthcare professionals assess fall risks and recommend interventions like balance training [56].
- **Chalder Fatigue:** The Chalder Fatigue Questionnaire, also known as the CFQ, is a tool used to assess fatigue severity. Developed in 1993, it's a self-administered questionnaire designed to be easy to understand. The CFQ asks eleven questions to gauge both physical and mental fatigue experienced over the past month. Scores range from 0 to 33, with higher scores indicating greater fatigue. This questionnaire is commonly used in clinical settings to evaluate fatigue in conditions like chronic fatigue syndrome and multiple sclerosis, but it can also be helpful for anyone experiencing persistent tiredness [57].
- **FACIT:** This section includes scores from the Functional Assessment of Chronic Illness Therapy (FACIT), which assesses the impact of chronic illness on daily life [58].
- **PTSD Scales:** These sections include scores from PTSD symptom assessment scales, which measure the presence and severity of post-traumatic stress disorder (PTSD) symptoms [59].
- **Beck Depression Inventory (BDI) and PCFS:** These sections include scores from the Beck Depression Inventory (BDI), which measures depression severity, and the Patient Concerns Form Short (PCFS),

which assesses patient concerns and priorities [60][61].

- **Doctor's Recommendation:**
    - **OUTPUT:** This variable captures the doctor's recommendation for the most urgent treatment specialty based on the initial screening (e.g., pneumology, cardiology, psychology, none). The meaning of each target value is described below.
        i. 0 → none
        ii. 1 → pneumologist
        iii. 2 → cardiologist
        iv. 3 → psychiatrist

3.1.3 Datasets

To investigate the impact of data complexity on model performance, we derived three distinct datasets from the original data. The first dataset included all possible target values (0, 1, 2, 3), representing the most complex scenario for analysis. This dataset served as the foundation for training the initial machine learning model.

The second dataset was created by excluding patients classified as needing cardiology consultations (class 2). This decision was motivated by the model's observed lower accuracy in predicting this specific specialist need. By removing this class, we aimed to assess the effect of a simplified classification task on model performance.

Finally, the third dataset involved combining classes 1, 2, and 3 into a single class. This manipulation resulted in a binary classification problem, further reducing the complexity compared to the original dataset. This approach allowed us to observe how model accuracy changes as the classification task transitions from a multi-class to a binary problem.

3.2 Exploratory Data Analysis

Exploratory data analysis (EDA) serves as the foundation for understanding the characteristics and potential relationships within our dataset. This initial investigation is crucial for identifying patterns, outliers, and potential issues that may require further exploration or data cleaning techniques [62]. In this section, the details of the dataset are described, including the distribution of variables, identifying potential missing values, and uncovering any initial insights that may guide subsequent analysis.

Initial exploration begins by examining the distribution of the target variable, depicted in Figure 15. This initial inspection reveals an inhomogeneous dataset, meaning the distribution of patients across different target classes is not uniform. Class 2 appears to hold the most patients, suggesting a potential imbalance.

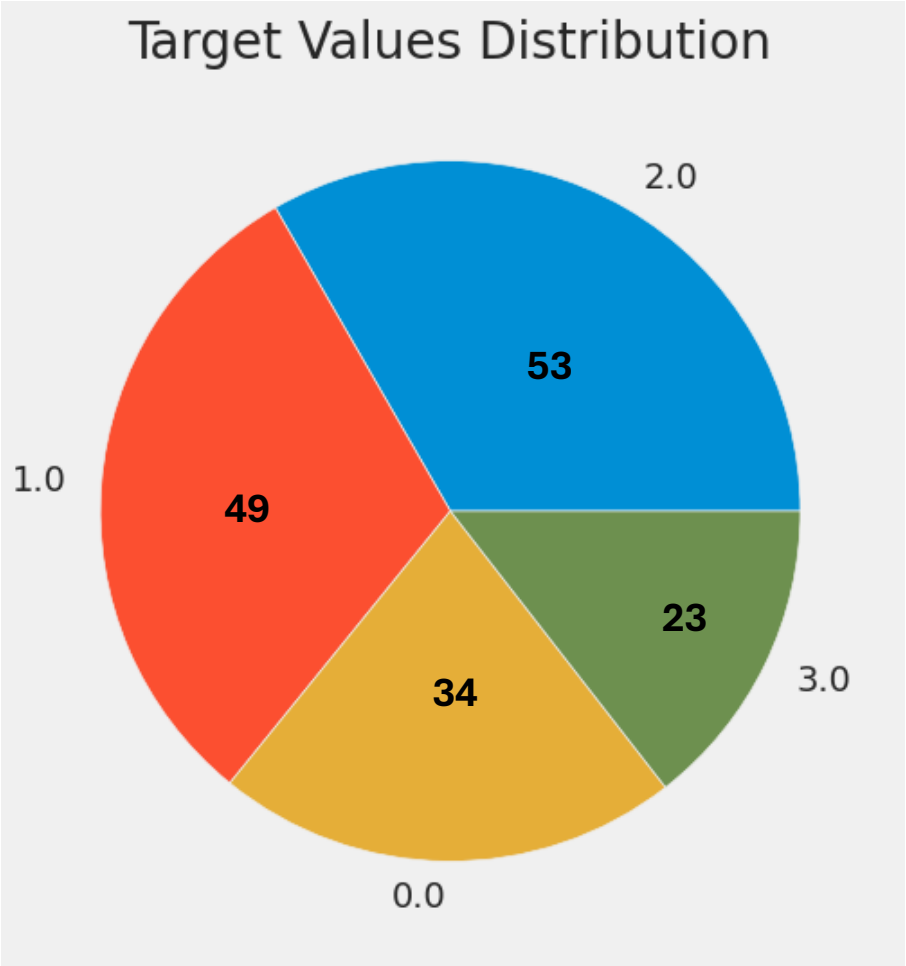


Figure 15: Distribution of the target values. The values refer to the dataset after deleting the patients missing the target values.

Figure 16 presents the distribution of gender across target values. The target variable categories are displayed on the x-axis, while the y-axis represents the patient count for each combination of target value and gender (0: Male, 1: Female). Interestingly, the gender distribution is not uniform across target categories. Target values 1.0, 2.0, and 3.0 show a higher prevalence of females, while Target 0.0 exhibits a predominantly male population. This observation suggests a potential association between gender and the target variable.

Thirty-three features, out of the eighty-eight of the dataset, were continuous features, These features were investigated using density plots (Figure 17). In data visualization, density plots offer a statistically robust technique for depicting the distribution of continuous variables. They employ kernel density estimation, a non-parametric approach, to create a smooth curve representing the probability density

function (PDF) of the data. This visual exploration revealed outliers within the distributions of height and the IPAQ questionnaire features. Careful examination of the data confirmed that these outliers coincided with missing values. To address this issue, we opted to treat the outliers in height and all IPAQ features as missing values and imputed them with zeros. This approach prioritises data completeness while acknowledging the potential limitations of replacing missing values with zeros.

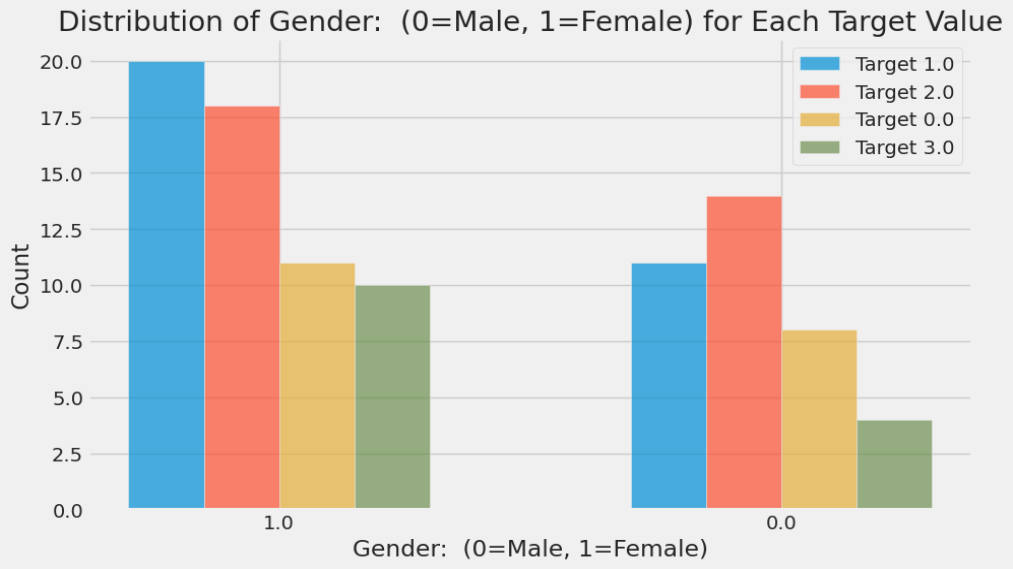


Figure 16: Gender ((0.0=Male, 1.0=Female) distribution among target values.

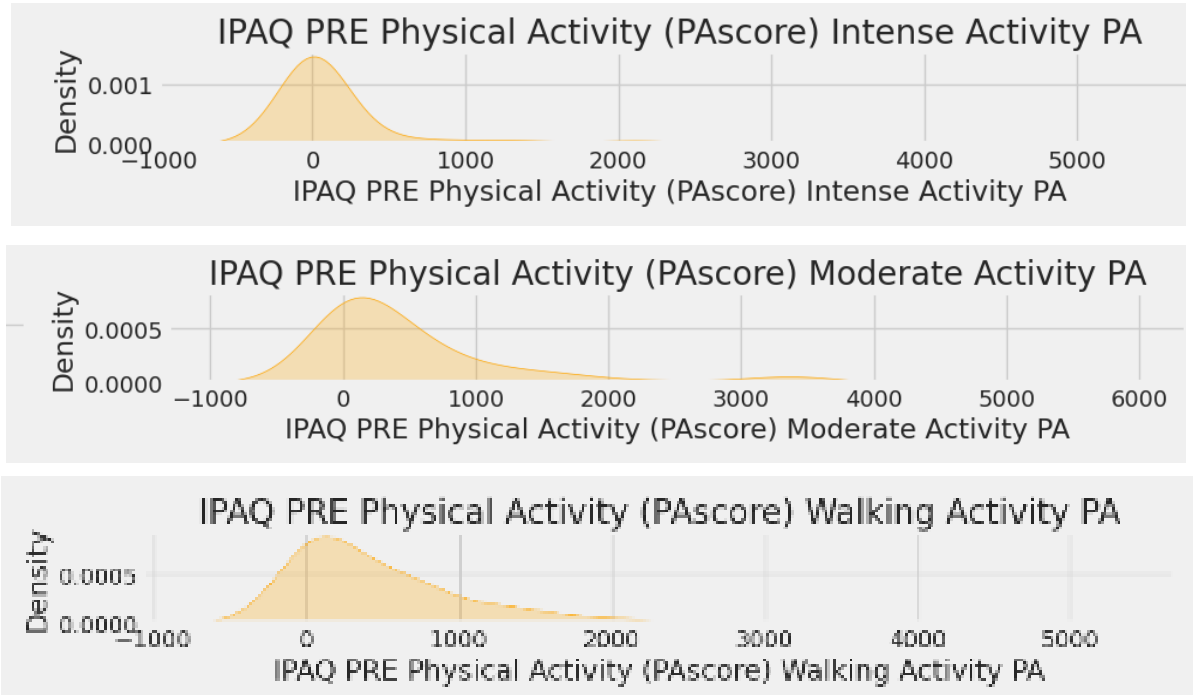


Figure 17: Selected density distribution of continuous features that had outliers.

Following the identification and treatment of outliers in height and IPAQ features (described in Section 2.3.1), revised density plots were generated (Figure 18). It is important to note that for these plots, IPAQ features besides the "Total PAscore (METs)" variable have been excluded. The rationale behind this exclusion will be addressed in the next section (2.3.2). This figure allows us to visualise the distribution of continuous features after addressing the outliers, providing a clearer understanding of the data for subsequent analysis.

The majority of features followed a normal distribution. This can be validated by the Quantile-Quantile (QQ) plots [63]. A QQ plot is a graphical tool used to compare the distribution of a dataset to a theoretical distribution, such as a normal distribution. It achieves this by plotting the quantiles (percentiles) of the data against the quantiles of the theoretical distribution. If the data closely follows the theoretical distribution, the points on the QQ plot will fall approximately along a straight diagonal line. Deviations from this line indicate that the data may not be normally distributed. Points falling above the line suggest the data have more extreme values (higher or lower) than the theoretical distribution, while points below the line indicate the data is more concentrated around the centre. Figure 19 presents the QQ plots for the continuous features of the dataset.

We further investigated the distribution of continuous features to assess normality. Figure 19 provides a visual representation of these distributions. While some features appear to deviate from a normal distribution, the majority exhibit a relatively normal shape. Additionally, the figure suggests the presence of potential outliers.

To gain a deeper understanding of these outliers, box plots were generated for all continuous features (Figures 20 and 21). The box plot for "Total PAscore (METs)" is presented separately (Figure 21) due to the higher range of values in this feature compared to the others. By examining the box plots, we can identify the interquartile range (IQR) for each feature and visually assess the distribution of data points, including the location of potential outliers relative to the rest of the data.

As described earlier (Section 2.3.1), we identified and addressed outliers in height and IPAQ features by treating them as missing values and imputing them with zeros. Following this step, further exploration of the data distribution using boxplots (Figures 20 and 21) revealed a limited presence of outliers. Given the minimal number of outliers remaining, we opted not to pursue more aggressive outlier removal techniques. This decision balances the value of retaining data points with the potential influence of outliers on the analysis. The current state of the data, with minimal outliers and a reasonable distribution for most features, provides a solid foundation for subsequent analysis.

Finally, the bar plot of the top 10 correlated features, acquired using the correlation coefficient, with the target value (Output) was created (Figure 22). This plot indicates that SF-36 PRE GH-N, the general health aspect of the SF-36 questionnaire, values are crucial to predict the target values.

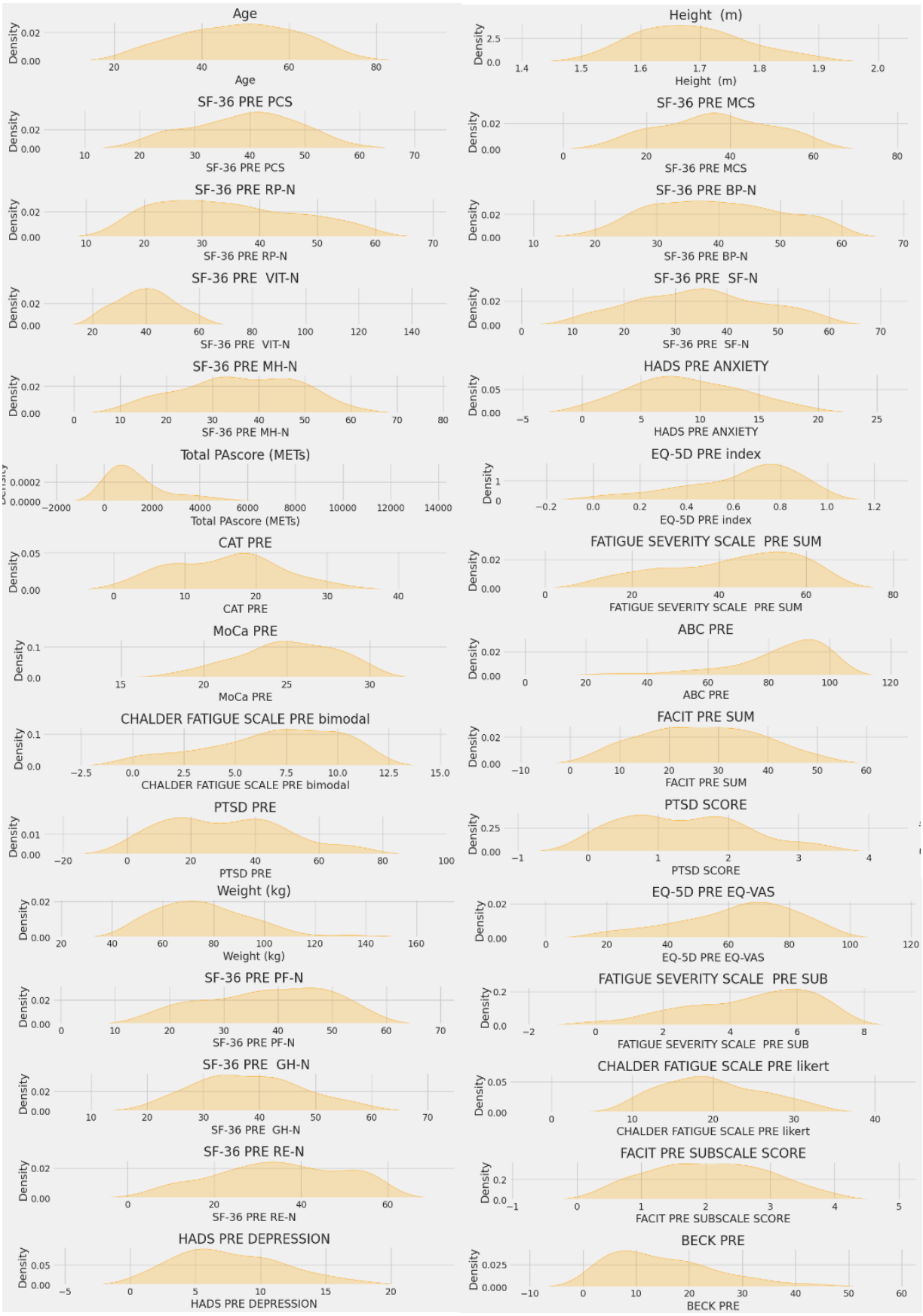


Figure 18:New density distribution of the features.

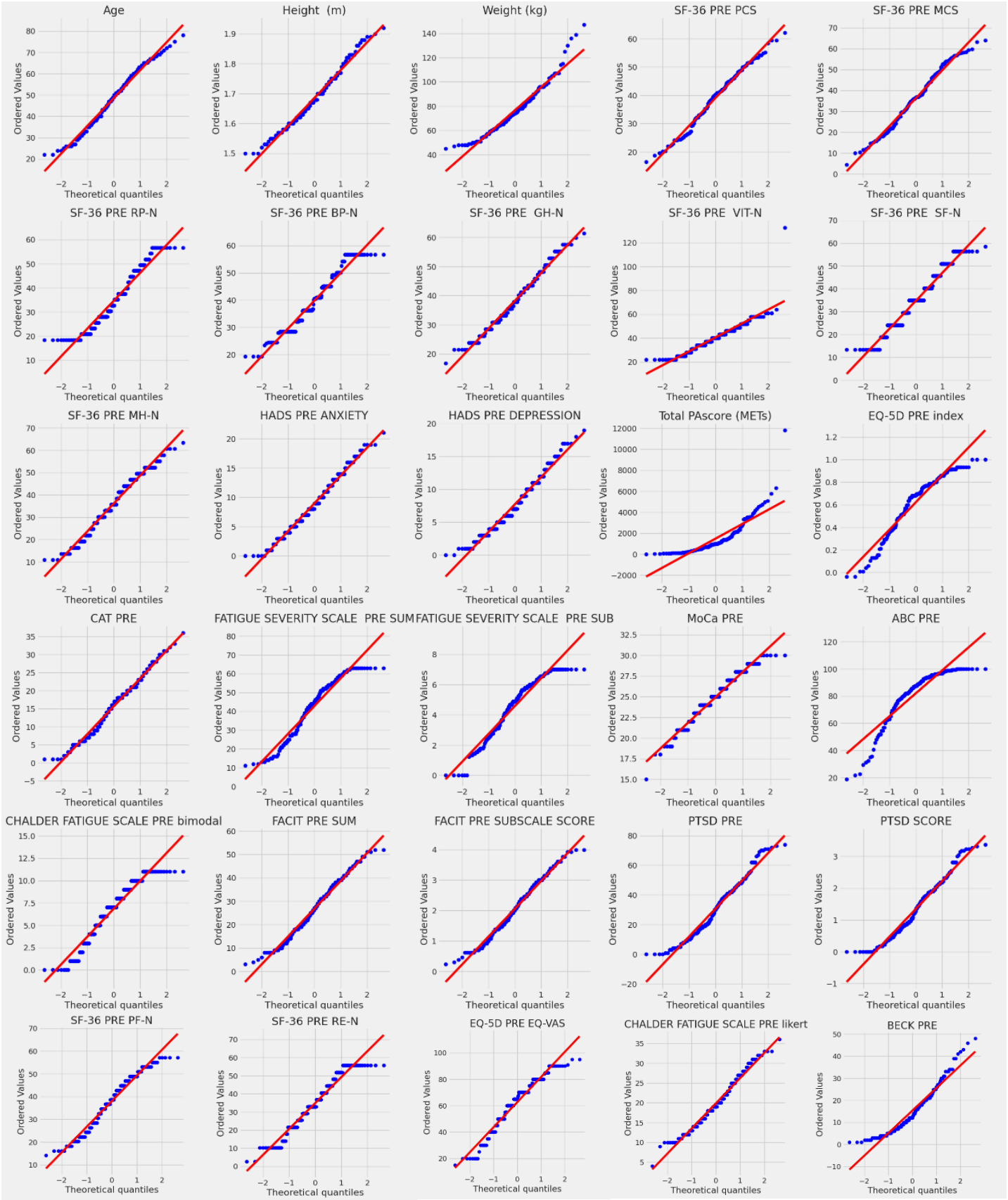


Figure 19: Quantile-Quantile (QQ) plots of the continuous features of the dataset.

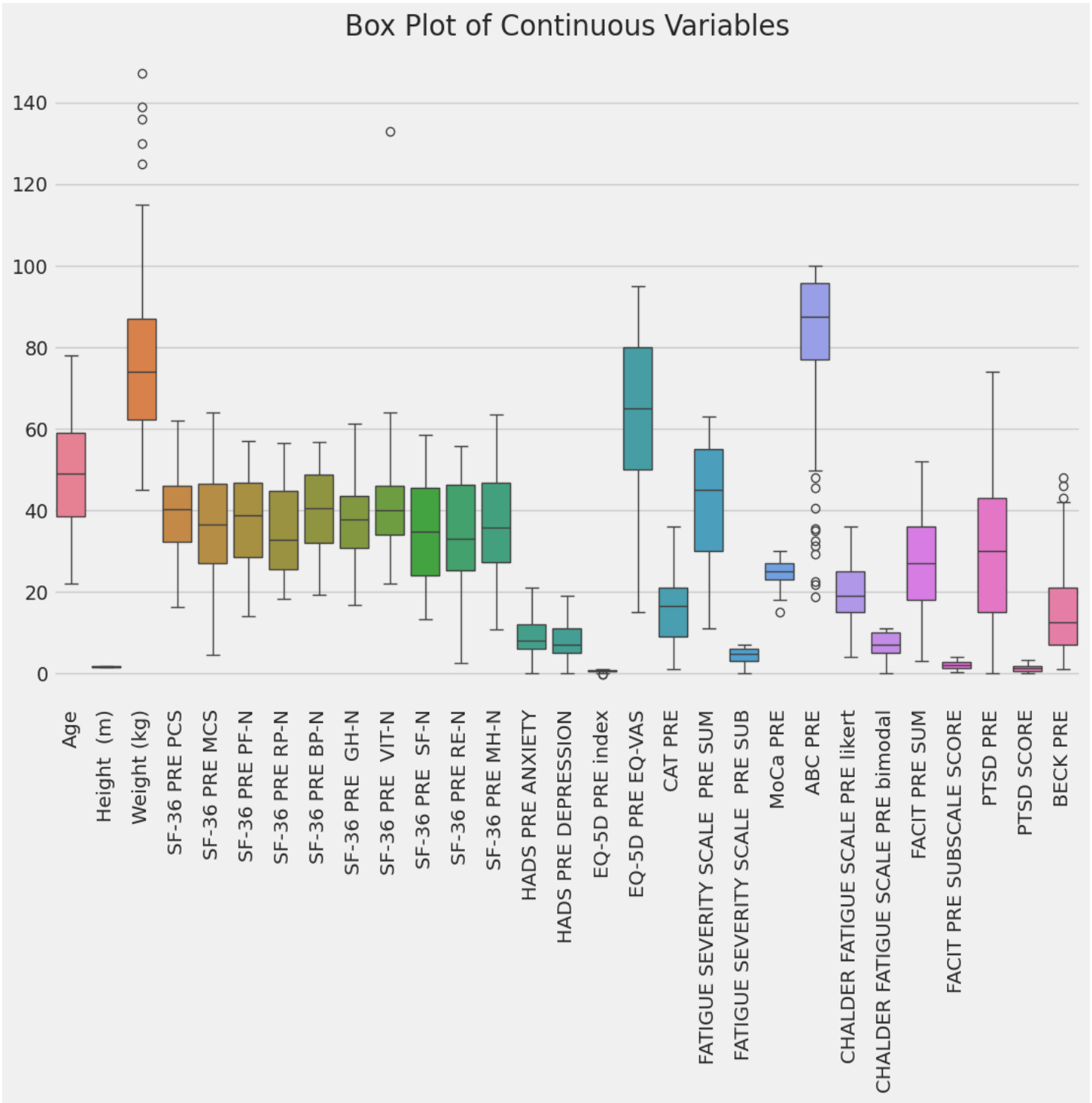


Figure 20: Box plots of the continuous features.

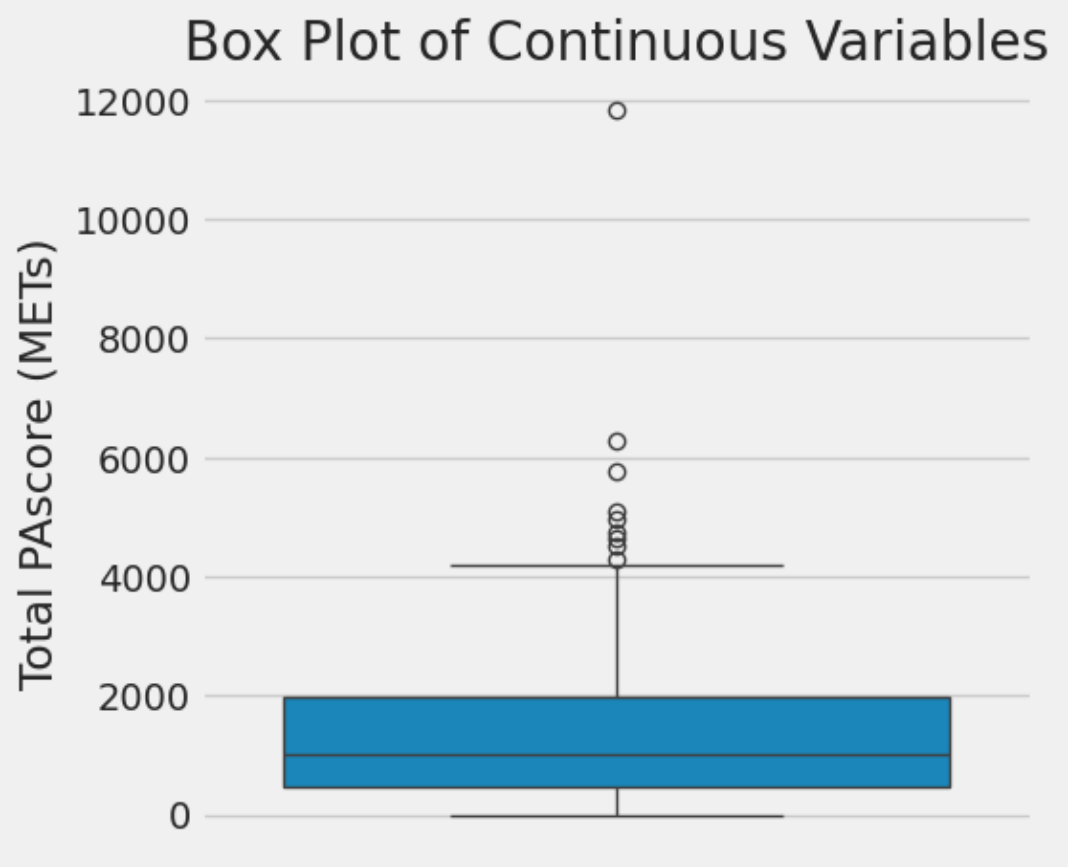


Figure 21: Box plot of the feature "Total PAscore (METs)".

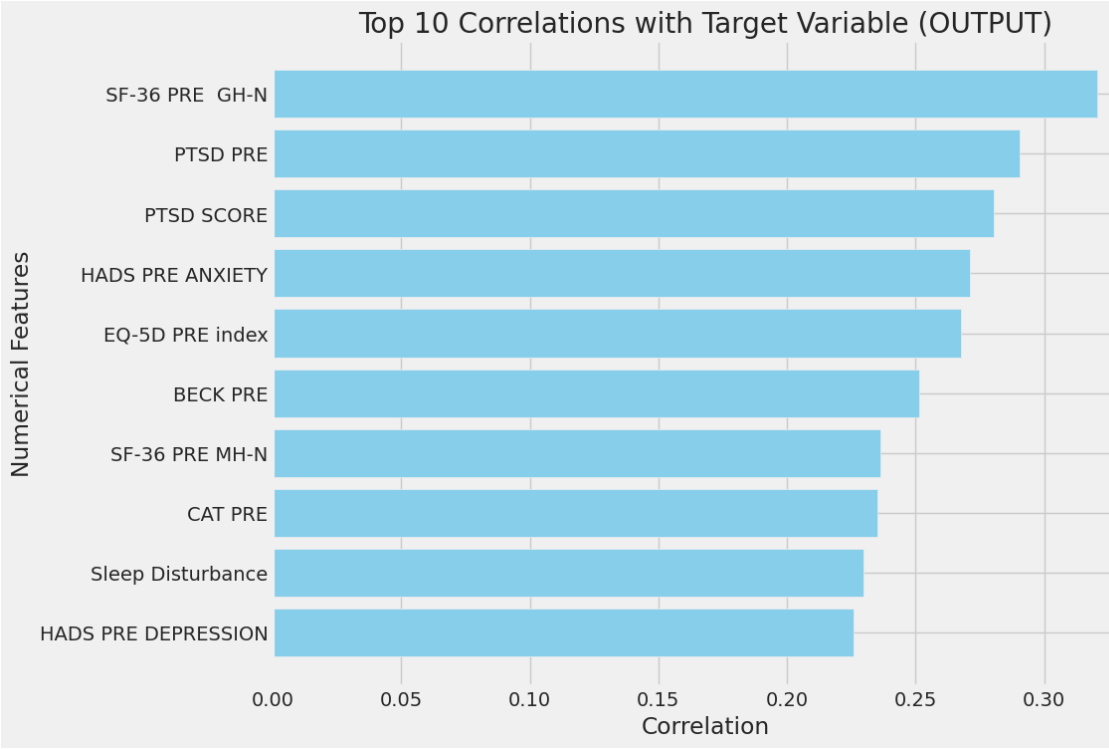


Figure 22: Top ten correlations with target variable.

3.3 Data Warehousing

3.3.1 Anonymisation

To ensure patient confidentiality and compliance with regulations like the General Data Protection Regulation (GDPR), all patient names within the dataset were anonymised. A unique five-digit random code was generated for each patient using the Excel *random* function. This approach effectively removes any personally identifiable information from the data, protecting patient privacy and allowing for ethical research practices. Anonymisation is crucial because it minimises the risk of data breaches and ensures that patients cannot be identified from the analysed data. This is especially important in the healthcare field, where patient data is highly sensitive.

3.3.2 Missing Values

The presence of missing data is a common challenge encountered in data analysis. In this study, the dataset contained approximately 7% missing values. We employed a targeted approach to address this issue, ensuring the integrity of the data and minimising potential biases.

For patients with missing values in the crucial "OUTPUT" variable, which is the target variable, exclusion was deemed the most appropriate course of action. These missing values could not be reliably imputed, and their inclusion could have significantly impacted the analysis. This exclusion resulted in the removal of 15 patients from the final dataset. A visualisation of the missing data can be found in Figure 23.

The analysis of missing data revealed significant absence of values for specific features within the IPAQ questionnaire. Notably, the "IPAQ PRE Physical Activity (PAscore) Intense Activity PA" feature lacked data for 82% of participants (144 values). Due to this substantial amount of missing information, this feature was excluded from further analysis. Similarly, "IPAQ PRE Physical Activity (PAscore) Moderate Activity PA" and "IPAQ PRE Physical Activity (PAscore) Walking Activity PA" were excluded due to missing data of 43% and 30%, respectively. This exclusion strategy prioritises using features with a higher data retention rate. Consequently, the IPAQ questionnaire is now represented by the "Total PAscore (METs)" variable, which exhibits a more manageable missing data rate of 19%. This approach ensures the inclusion of relevant information from the IPAQ questionnaire while minimising the impact of missing data on the overall analysis.

For missing values, we opted for a mean imputation technique based on patient groups defined by the "OUTPUT" variable. This approach acknowledges the potential relationship between missing values and the output categories. We calculated the mean value for each feature within each group defined by the "OUTPUT" variable. This group-wise approach ensures that the imputed values are more relevant to the specific patient characteristics associated with each output category. To improve

interpretability, especially for continuous variables like height, the calculated mean values were then rounded to a specific number of decimal places. Finally, the missing values in each patient record were filled in using the corresponding rounded mean values based on their "OUTPUT" category.

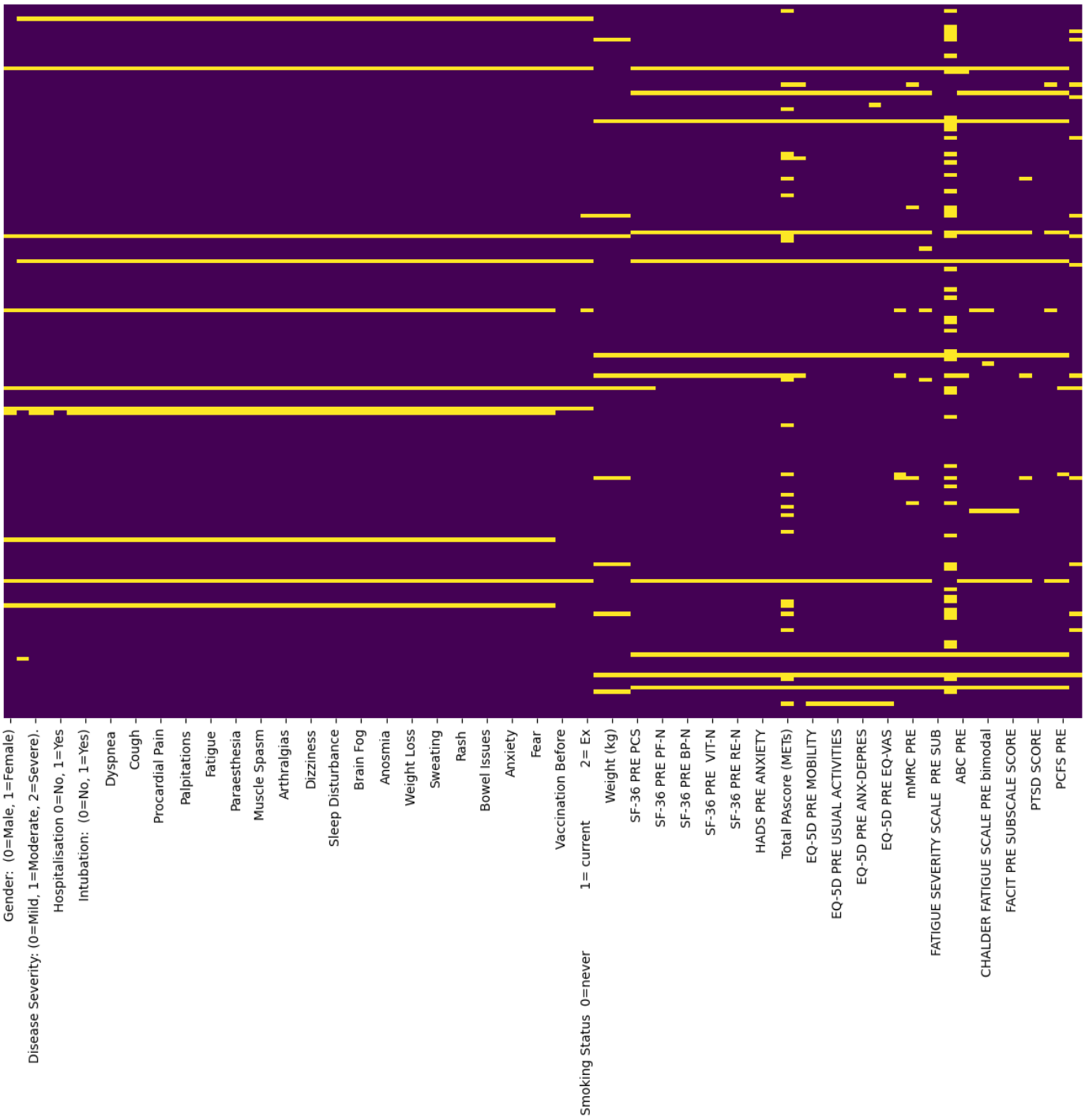


Figure 23: Heatmap of the missing data. Yellow points indicate the missing data, while the velvet area represents the data that are available.

This approach to missing data management balances the need to retain as much data as possible while minimising the introduction of bias. By strategically excluding patients with missing output values and employing a group-wise mean imputation for other features, we ensured a robust and reliable dataset for further analysis.

3.3.3 Handling of categorical features

This analysis explores two approaches to handling categorical data within the dataset. The first approach utilises the raw categorical data, where features and the target variable hold values like 0, 1, 2, and 3. While this representation preserves the original data, it can be challenging for some ML algorithms to directly interpret these categories.

The second approach addresses this by converting the categorical variables into dummy variables. This technique creates a new binary feature for each category within a categorical variable (often encoded as 0 or 1). This allows the ML model to learn the relationship between the different categories and the target variable more effectively. The choice between these approaches depends on the specific model being used and the nature of the data.

By comparing the performance of models trained on both representations of the data, we aimed to investigate the potential for bias reduction and improvement in model accuracy achieved through the use of dummy variables. The results of this comparison will be presented in a subsequent section (Section 4) to determine the most appropriate approach for further analysis.

3.3.4 Feature selection

To optimise model performance and accuracy, a feature selection process was implemented. This process aimed to identify the most relevant features from the dataset while minimising redundancy. We employed a function named *select_and_fit_features* that takes the input features (X), target variable (y), and a feature selector object as arguments. The feature selector object utilises a specific method (e.g., *fit_transform*) to analyse the features in relation to the target variable and select a subset that holds the most predictive power.

In this analysis, we utilised a feature selector from scikit-learn called *SelectKBest*. This selector employs the *f_classif* scoring function to evaluate features based on their ANOVA F-value, a statistical measure that assesses the separability of classes based on a feature. The k parameter of the selector determines the desired number of features to retain, which will be optimised to achieve maximum model accuracy. The *select_and_fit_features* function is called with the imputed features, target variable, and the chosen feature selector to perform the selection and transformation. This approach ensures that the models are trained on a focused set of features most likely to contribute to accurate predictions.

3.3.5 SMOTE

Since our dataset exhibited an imbalance in class distribution, where some specialist consultation categories had fewer data points compared to others, we employed the Synthetic Minority Oversampling Technique (SMOTE) to address this issue [64]. SMOTE helps create synthetic samples for underrepresented specialist consultation groups, balancing the data and ensuring the ML model learns effectively from all categories. This step is crucial for building a robust model that can accurately prioritise patients for different specialist consultations.

3.4 Model Training

3.4.1 Stratified K-Fold Cross-Validation

Our dataset presented a challenge due to its imbalanced class distribution. This means that some specialist consultation categories had significantly fewer data points compared to others. To ensure the model's generalizability and prevent evaluation sets (folds) from containing only one class during cross-validation, we employed Stratified K-Fold Cross-Validation. This technique addresses class imbalance by creating evaluation sets within the data that maintain the same class proportions as the entire dataset (stratification). This ensures that each class is fairly represented across all these evaluation sets used for model evaluation. In supervised classification tasks, this specifically guarantees balanced representation across all folds, providing a more robust assessment of the model's performance on unseen data.

3.4.2 Exploring Model Training Session Through Data Complexity and Missing Data Handling

For each combination of dataset and missing data approach, we trained and evaluated a variety of ML models. These models included Support Vector Machines (SVM), Random Forests, Decision Trees, XGBoost, and K-Nearest Neighbours (KNN). Also, a Convolutional Neural Networks (CNN) model and a Long Short-Term Memory (LSTM) model, for the datasets whose missing data handled using the non-dummy approach, were explored. This selection aimed to assess the effectiveness of deep learning models for this specific scenario while acknowledging time constraints.

The above can be visualised as in Figure 24, showing how the experimental part of this thesis was implemented.

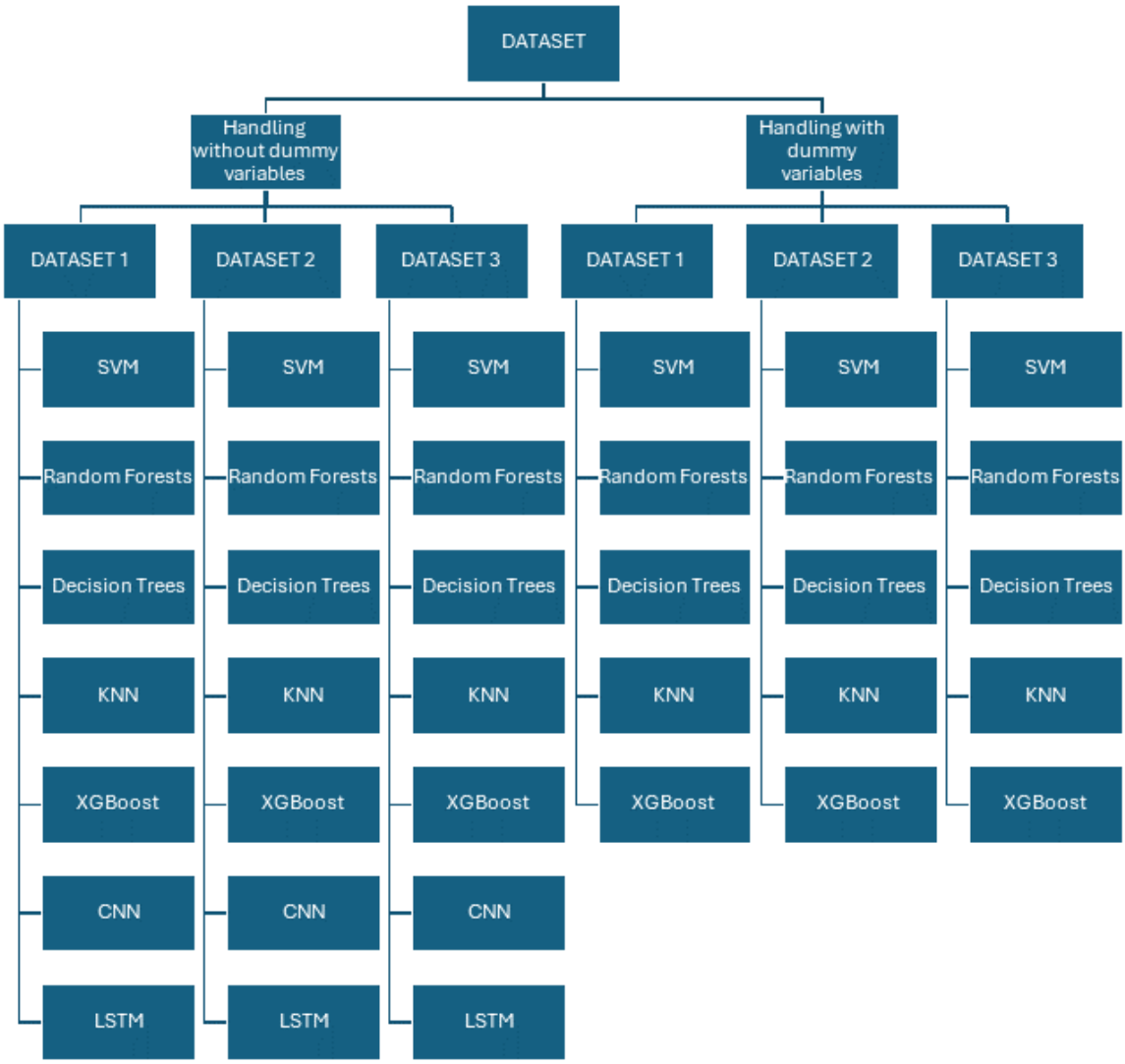


Figure 24: Experimental structure of this thesis.

3.4.2 Training sessions

3.4.2.1 SVM model

Our training process addressed class imbalance with SMOTE and employed 5-fold cross-validation for robust evaluation (via the evaluate_model function). Feature selection technique SelectKBest with k parameter equals to 20, was iteratively applied, followed by SMOTE on the selected features. For the Support Vector Classifier (SVC) model, hyperparameter tuning with RandomizedSearchCV identified the best configuration, see Table 2. This optimised model was then evaluated using the evaluate_model function. This SVM model follows a One-vs-Rest (OvR) approach. This approach involves training one classifier per class. Each classifier is trained to distinguish one class from all the other classes. In other words, for each classifier, one

class is considered the positive class, and all other classes are considered the negative class

Table 2: The hyperpameters for the Support Vector Machine (SVM) model.

non-dummy variables approach			
	kernel	gamma	C
Dataset 1	poly	0.001	1
Dataset 2	rbf	0.001	1000
Dataset 3	rbf	0.001	100
dummy variables approach			
Dataset 1	rbf	0.001	1000
Dataset 2	poly	0.001	1
Dataset 3	rbf	0.001	100

The hyperparameters above are:

- kernel or kernel function: maps the data to a higher-dimensional space where a linear decision boundary can be found. There are various types of kernels, like linear, polynomial, radial basis function (RBF), and sigmoid.
- gamma or kernel coefficient, or: affects the shape and smoothness of the decision boundary.
- C or regularisation parameter: balances between maximizing the margin and minimising the training error.

3.4.2.2 Random Forest model

For the Random Forest model, the training procedure consists of a feature selection process with k variable being 30. The value for k is derived after an exhaustive research among values from 2 to 80 with a step of 1. After that SMOTE is used on the chosen features for each technique. RandomizedSearchCV identified the best hyperparameters (see Table 3) for each RFC on each dataset. These optimised models were then evaluated using the evaluate_model function.

Table 3: Hyperparameters for the Random Forests model.

non-dummy variables approach					
	n_estimators	min_samples_split	min_samples_leaf	max_depth	bootstrap
Dataset 1	100	5	2	9	True
Dataset 2	400	2	1	7	True
Dataset 3	100	5	2	9	True
dummy variables approach					
Dataset 1	400	5	1	9	False
Dataset 2	400	5	1	9	False
Dataset 3	200	5	2	9	True

The hyperparameters mentioned above are:

- Number of Trees/estimators: The quantity of decision trees in the forest.
- Minimum Samples per Split: Minimum samples required to split a node.
- Minimum Samples per Leaf: Minimum samples required in a leaf node.
- Max Depth: Maximum depth of each decision tree.
- Bootstrap Sampling: Whether to use bootstrap sampling while building trees.

3.4.2.3 Decision Tree model

The training regimen for the decision tree classifier follows a rigorous approach to ensure optimal performance and mitigate potential data challenges. Stratified K-Fold cross-validation is employed to segment the data into training and testing sets while maintaining the inherent class distribution within each fold. To maximise feature relevance, SelectKBest was implemented. This method identified the top 20 most informative features for the model. Subsequently, to address potential class imbalance issues, SMOTE was utilised to synthetically generate data points for the underrepresented class. Hyperparameter tuning was then conducted via GridSearchCV, exploring a comprehensive grid of criteria, maximum depth, and other relevant parameters for the decision tree classifier (Table 4). Finally, the model's performance was evaluated using cross-validation predictions. This evaluation encompassed metrics such as the classification report and ROC AUC score, providing a thorough assessment of the model's effectiveness.

Some key decision tree hyperparameters and how they influence the model are described below.

- Criterion (gini or entropy)**:** This parameter defines how the quality of a split at each node in the tree is measured. The two most common options are:

    - gini impurity**:** Favours splits that maximise the difference in the class distribution between child nodes. This leads to a purer separation of classes.
    - entropy**:** Measures the randomness or uncertainty in a node. A lower entropy value indicates a better split.

    Choosing between gini and entropy can depend on the dataset and task. Generally, gini is preferred for classification with imbalanced datasets, while entropy might be suitable for balanced datasets.

- Max_Depth (integer)**:** This hyperparameter controls the maximum depth a tree can grow. A deeper tree can potentially capture more complex relationships in the data but might also lead to overfitting. Setting a smaller max_depth restricts the tree's complexity and helps prevent overfitting, especially on smaller datasets.

- Min_Samples_Split (integer or float)**:** This parameter determines the minimum number of samples required to split an internal node. Splits are only considered if there are at least min_samples_split samples in the node. A higher value can prevent overfitting by avoiding splits with very few data points. However, setting it too high might lead to underfitting by preventing the tree from capturing finer details in the data.

- Min_Samples_Leaf (integer or float): This hyperparameter specifies the minimum number of samples allowed in a leaf node. Splits are not performed if the resulting child nodes would have fewer samples than min_samples_leaf. Higher values can help prevent overfitting by ensuring some minimum representation of each class in the leaf nodes.

- Max_Features (integer, float, or specific options)**:** This parameter controls how many features are considered at each split. There are three main options:

    - integer**:** Only the specified number of features werr considered for splitting.
    - float (fraction)**:** A fraction of the total features was randomly selected for each split.
    - specific options**:**
        - "auto": Square root of the total features is used.
        - "sqrt": Same as "auto".

- "log2": Logarithm base 2 of the total features is used.

Table 4: Hyperparameters for the Decision Tree model.

non-dummy variables approach					
	criterion	max_depth	max_features	min_samples_leaf	min_samples_split
Dataset 1	entropy	50	sqrt	1	2
Dataset 2	entropy	30	log2	2	5
Dataset 3	entropy	50	sqrt	4	10
dummy variables approach					
Dataset 1	gini	40	sqrt	1	2
Dataset 2	gini	10	auto	1	2
Dataset 3	entropy	40	sqrt	1	10

3.4.2.4 K- Nearest Neighbours model

The model utilises SelectKBest as a feature selection technique within the KNN model training process. This approach selects a predefined number of features (k) based on a specific scoring function. In this case, the code uses SelectKBest to select the top 45 features based on the f_classif scoring function. This function evaluates features based on the ANOVA F-value, favouring features that have a high difference in the class distribution between child nodes in a decision tree.

While GridSearchCV is used for hyperparameter tuning (Table 5) of the KNN model, it does not directly impact the number of features used. Therefore, the number of features in the final KNN model is determined by the selection with SelectKBest. This

approach focuses on features that are most relevant to class separation as measured by the F-value.

Table 5: Hyperparameters for KNN model.

non-dummy variables approach			
	metric	n_neighbours	weights
Dataset 1	manhattan	3	distance
Dataset 2	manhattan	5	distance
Dataset 3	euclidean	7	distance
dummy variables approach			
Dataset 1	manhattan	3	distance
Dataset 2	manhattan	7	distance
Dataset 3	manhattan	3	distance

The above hyperparameters are explained bellow.

- metric**:** This parameter defines the distance metric used to calculate the distance between data points. The default metric is 'minkowski', which includes Euclidean distance (p=2) as a common choice. Other options include Manhattan distance (p=1), Chebyshev distance (p=inf), and custom distance metrics defined as callable functions. The appropriate metric depends on the characteristics of your data and the relationships between features.

- number of neighbours**:** This parameter defines the number of nearest neighbours to consider when classifying a new data point. Higher values can lead to smoother decision boundaries and potentially reduce the impact of noise but might also increase the risk of overfitting. Lower values can capture more local variations in the data but might be more susceptible to noise.

- weights**:** This parameter specifies how to weight the contributions of neighbouring points when making predictions. There are three main options:
  - 'uniform'**:** All neighbours within the specified distance contribute equally to the prediction.

- 'distance': Neighbours closer to the query point have a higher weight than those farther away. This can be useful when the closer neighbours are likely to be more informative.
- callable: A custom function can be defined to assign weights based on a specific logic.

Choosing the appropriate weight function can influence the decision boundary of the KNN model and potentially improve its performance.

3.4.2.5 XGBoost model

The XGBoost model training session, uses stratified K-Fold cross-validation for balanced evaluation across folds. Feature selection with SelectKBest was employed to pinpoint the 20 most relevant features. The value for k, derived after an exhaustive search among different values from 2 to 80 with a step of 1. To rectify class imbalance, SMOTE augments the minority class with synthetic samples. The heart of the training lies in hyperparameter tuning (Table 6) with RandomizedSearchCV, which optimises the XGBoost model's performance. This is followed by unseen data evaluation using metrics like classification reports and ROC AUC score (not shown here). Finally, Recursive Feature Elimination (RFE) hones the model by iteratively removing the least significant features, ultimately identifying the most crucial ones for accurate classification.

Table 6: Hyperparameters for XGBoost model.

non-dummy variables approach					
	subsample	n_estimators	max_depth	learning_rate	colsample_bytree
Dataset 1	0.8	400	9	0.2	1.0
Dataset 2	0.7	300	5	0.01	0.6
Dataset 3	0.7	500	7	0.05	0.6
dummy variables approach					
Dataset 1	0.8	100	7	0.3	0.6
Dataset 2	0.6	200	3	0.2	0.6
Dataset 3	1.0	300	7	0.3	0.8

A brief description of the above hyperparameters can be found below.

- Tree-Specific Hyperparameters:
    - **max_depth:** This parameter controls the maximum depth a tree can grow. Deeper trees can capture more complex relationships in the data but are also more prone to overfitting.
    - **subsample:** This value determines the proportion of rows used for building each tree. Lower subsample values can help prevent overfitting by training on a smaller portion of the data in each iteration.
    - **colsample_bytree:** This parameter controls the percentage of features considered when splitting a node in each tree. Lower values can help reduce overfitting by training on a subset of features at each split, potentially leading to more robust models.
- Learning Task-Specific Hyperparameters:
    - **n_estimators:** This parameter defines the number of trees used in the XGBoost ensemble model. More trees can improve model accuracy but also increase training time and complexity.
    - **eta (learning rate):** This value controls the step size taken during model updates and helps prevent overfitting. Smaller learning rates lead to more conservative updates, making the model less prone to overfitting.

For the non-dummy approach only

3.4.2.6 Convolutional Neural Network (CNN) model

The implementation of a 1D Convolutional Neural Network (CNN) to address a classification task with imbalanced classes is described here. After data preparation and splitting into training and test sets, SMOTE is applied to generate synthetic samples for the minority class, balancing the class distribution in the training data. The data is then reshaped to a format suitable for the 1D CNN architecture.

The CNN model itself is a sequential model with multiple convolutional layers for feature extraction, interspersed with pooling layers for dimensionality reduction[38]. A flattening layer prepares the extracted features for dense layers, which handle higher-level abstraction. The final dense layer with softmax activation outputs probabilities for the four predicted classes.

To address class imbalance during training, class weights were calculated based on the class distribution in the training data. These weights were used during model compilation, assigning greater importance to the minority classes during the training process. Finally, the model was trained on the resampled data with class weights, and its performance was evaluated on the unseen test data using metrics like accuracy. The training history, including training and validation accuracy over epochs, is visualised to assess the model's learning behaviour and potential for overfitting.

A more detailed description of the CNN layers follows.

Convolutional Layers (Conv1D):

- These layers are the core of feature extraction in CNNs. They apply filters (also called kernels) that slide across the input data, capturing local patterns and feature activations.
- The code defines three convolutional layers:
    - **Conv1D(64, activation='relu')**: This layer uses 64 filters. The "relu" activation function introduces non-linearity, allowing the network to learn more complex relationships between features.
    - **Conv1D(128, activation='relu')**: Similar to the first layer, this layer uses 128 filters. As the network progresses through layers, it can learn increasingly complex features by combining simpler ones learned in earlier layers.
    - **Conv1D(256, activation='relu')**: The final convolutional layer uses 256 filters with "relu" activation. This layer likely extracts even more intricate features based on the preceding layers' outputs.

Pooling Layers (MaxPooling1D):

- These layers perform downsampling, reducing the dimensionality of the data while preserving the most significant features.
- The code uses two MaxPooling1D layers, likely with a pooling size of 2 (not explicitly mentioned). Max pooling takes the maximum value within a window of size 2 on the input, effectively reducing the size of the data by half in each dimension. This helps control overfitting and reduces computational costs.

Flattening Layer:

- This layer transforms the multi-dimensional output from the convolutional layers (typically with height, width, and channels) into a single long vector. This is necessary to connect the extracted features to the fully connected layers.

Dense Layers:

- These layers perform more complex, non-linear transformations on the flattened data. They learn higher-level abstractions and relationships between the extracted features.
- The code defines two dense layers:
    - **Dense(128, activation='relu')**: This layer has 128 neurons and uses "relu" activation. It processes the flattened features and learns more intricate relationships between them.
    - **Dense(n, activation='softmax')**: The final dense layer has n neurons, corresponding to the n predicted classes in the classification task. It

uses "softmax" activation, which outputs a probability distribution for each class, ensuring the sum of probabilities across all classes is 1.

3.4.2.7 Long short-term memory (LSTM) model

The following paragraphs outline a training process for a Long Short-Term Memory (LSTM) network for a multiclass classification task with imbalanced classes. After data preparation, including handling missing values and encoding the target variable, the data were split into training and test sets while maintaining class proportions using stratified splitting.

To address class imbalance, SMOTE was applied to the training data, generating synthetic samples for the minority class. The data were then reshaped to a format suitable for LSTM processing, where each feature vector was treated as a sequence with one dimension.

The LSTM model itself is a sequential model with a single LSTM layer containing 64 memory units. LSTMs are adept at capturing long-term dependencies in sequential data, which can be crucial for classification tasks involving sequences. The LSTM layer is followed by a dense layer with four output neurons and softmax activation. The four neurons correspond to the four classes, and softmax activation ensures the model outputs probabilities for each class, enabling multiclass classification.

The model was compiled with the Adam optimiser and sparse categorical cross-entropy loss, which is suitable for this multiclass classification scenario. Training occurs for 10 epochs on the resampled data, and validation data were used to monitor the model's performance and prevent overfitting. Finally, the model was evaluated on the unseen test data, and its performance metrics like accuracy are reported. The training history, including training and validation accuracy over epochs, is visualised to assess the model's learning behaviour.

A detailed description of the classes follows below.

1. LSTM Layer (layers.LSTM(64, input_shape=(X_resampled.shape[1], 1))):

- **Function:** This layer is the core component of the LSTM model and is responsible for capturing long-term dependencies in sequential data. It utilizes a special architecture with memory cells, gates, and internal connections that allow it to learn from past information and retain relevant context for future predictions.
- **Parameters:**

- **64:** This defines the number of memory units within the LSTM layer. Each memory unit has the capability to store and process information relevant to the sequence. A higher number of units can potentially capture more complex relationships but also increases model complexity.
- **input_shape=(X_resampled.shape[1], 1):** This specifies the expected format of the input data. Here, X_resampled.shape[1] represents the number of features in each sample, and 1 indicates that the features are treated as a single sequence with one dimension.

2. Dense Layer (layers.Dense(n, activation='softmax')

- **Function:** This layer performs the final classification task. It takes the output from the LSTM layer (the hidden state), which represents the processed sequence information, and maps it to class probabilities.
- **Parameters:**
  - **n:** This signifies the number of output neurons, corresponding to the four classes the model needs to predict.
  - **activation='softmax':** This activation function ensures the output from the layer sums to 1, interpreting the values as probabilities for each class. The class with the highest probability is predicted for a given sequence.

In summary, the LSTM layer extracts and retains crucial information from the sequential data, while the dense layer with softmax activation translates the processed information into class probabilities for multiclass classification. This combination allows the LSTM network to effectively learn patterns and make predictions based on sequential data.

4. Results

This part of the thesis reviews the performance of the models used to our classification problem. The evaluation scores for the different models are presented followed by the feature importance plot and the confusion matrix for the results per class for the SVM model used in dataset 1.

From the results, it is clear theta all models exhibited variations in performance depending on the data handling approach (dummy vs. non-dummy) and the number of target classes present

4.1 SVM model

When all target classes were included (dataset 1), the model achieved higher Accuracy (0.67) and ROC AUC score (0.84) using the dummy approach, than using the non-dummy approach to handle the categorical features. Also, for the scenario with one missing target class (dataset 2), the dummy approach maintained a comparable Accuracy (0.70) and ROC AUC score (0.78) to the non-dummy approach.

Interestingly, the dummy approach also yielded better scores than the non-dummy approach for the binary classification scenario (Table 7). Overall, the SVM model demonstrated promising performance for multi-class and potentially binary classification tasks.

4.2 Random Forests model

When all target classes were included, the “dummy” model achieved an accuracy of 0.64 and an ROC AUC score of 0.83. Interestingly, the non-dummy approach yielded a lower Accuracy of 0.55 but a comparable ROC AUC score of 0.79 for this complete dataset. Also, for the scenario with one missing target class (dataset 2), the dummy approach generally yielded higher performance across both metrics (Accuracy: 0.75 vs. 0.67, ROC AUC: 0.88 vs. 0.84). Overall, the random forest model demonstrated promising performance for multi-class classification tasks, highlighting the importance of choosing an appropriate data handling technique depending on the data characteristics, see Table 7.

4.3 Decision Tree Model

The impact of the data handling approach also differed across the chosen performance metrics (Accuracy and ROC AUC Score) see Table 7.

For Accuracy, the dummy approach consistently achieved higher scores across all scenarios. When all target classes were included, the dummy approach reached a score of 0.51 compared to the non-dummy approach's 0.46. This trend continued for the three target value (dataset 2) scenario (0.58 vs. 0.56) and even extended to the

binary classification dataset (dataset 1), where the dummy approach achieved a notable lead (0.74 vs. 0.64). These results suggest that the dummy approach might be generally preferable for maximizing Accuracy in this dataset, regardless of the number of target classes.

The ROC AUC score results were more nuanced. The dummy approach maintained consistent scores (around 0.68) for both all and three target value scenarios. However, in the binary classification case, the dummy approach showed a slight improvement (0.72) compared to the non-dummy approach (0.63). While the difference is smaller than for Accuracy, it suggests the dummy approach might also offer some benefit for ROC AUC score, particularly in simpler classification tasks with fewer target classes.

4.4 KNN model

There's a clear advantage to using dummy target values for the KNN model in this case, particularly when maximising Accuracy is the primary concern, see Table 7. The dummy approach consistently yielded higher Accuracy scores across all scenarios. With all target values included, the dummy approach achieved an Accuracy of 0.60 compared to the non-dummy approach's 0.54. This trend continued for the three target value (dataset 2) scenario (0.69 vs. 0.65) and became even more pronounced in the binary classification (dataset 1) case (0.80 vs. 0.77).

The impact of data handling on ROC AUC score is less clear-cut compared to Accuracy. The dummy approach maintained consistently high scores (around 0.78-0.82) across all scenarios. The non-dummy approach also achieved good scores (around 0.75-0.81), with a decrease compared to the dummy approach in some cases.

4.5 XGBoost model

Interestingly, the model generally achieved better results with the dummy approach for both Accuracy and ROC AUC score, see Table 7.

For Accuracy, the dummy approach yielded slightly better results across all scenarios. When all target classes were included, the dummy approach achieved an Accuracy of 0.59 compared to the non-dummy approach's 0.53. This trend continued for the three-target value (dataset 2) scenario (0.75 vs. 0.71) and even extended to the binary classification (dataset 1) case (0.81 vs. 0.79).

The impact of the data handling approach was even more pronounced for ROC AUC score. The dummy approach achieved consistently high scores across all scenarios (around 0.81-0.87). In contrast, the non-dummy approach, while still exhibiting good scores (around 0.77-0.86), showed a decrease compared to the dummy approach in all cases.

Table 7: Evaluation scores for the XGBoost model. SVM: Support Vector Machine, RF: Random Forest, DT: Decision Tree, KNN: K- Nearest Neighbours.

Performance Metric	Data Handling Approach	Dataset 1	Dataset 2	Dataset 3
SVM				
Accuracy	Dummy	0.67	0.70	0.75
	Non-dummy	0.63	0.67	0.70
ROC AUC Score	Dummy	0.84	0.78	0.78
	Non-dummy	0.78	0.77	0.77
RF				
Accuracy	Dummy	0.64	0.75	0.81
	Non-dummy	0.55	0.67	0.79
ROC AUC Score	Dummy	0.83	0.88	0.88
	Non-dummy	0.79	0.84	0.86
DT				
Accuracy	Dummy	0.51	0.58	0.74
	Non-dummy	0.46	0.56	0.64
ROC AUC Score	Dummy	0.64	0.68	0.72
	Non-dummy	0.62	0.68	0.63
KNN				
Accuracy	Dummy	0.60	0.69	0.80
	Non-dummy	0.54	0.65	0.77
ROC AUC Score	Dummy	0.78	0.82	0.80
	Non-dummy	0.75	0.79	0.81
XGBoost				
Accuracy	Dummy	0.59	0.75	0.81
	Non-dummy	0.53	0.71	0.79
ROC AUC Score	Dummy	0.81	0.87	0.87
	Non-dummy	0.77	0.84	0.86

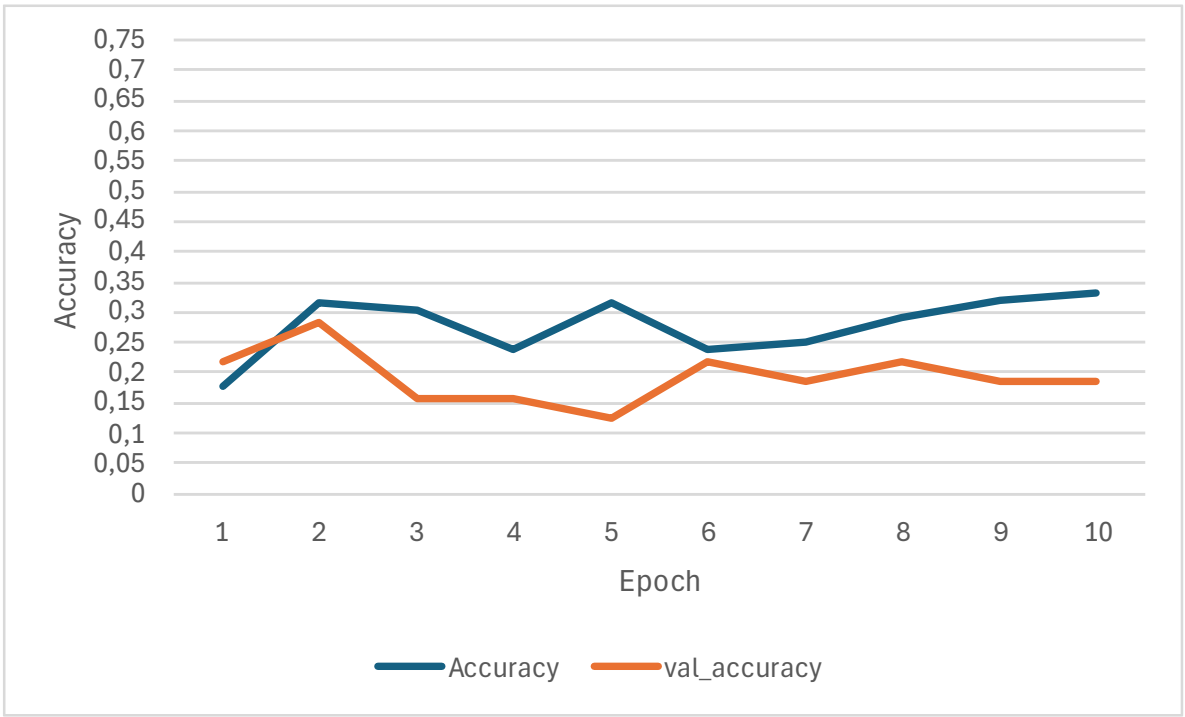
4.6 CNN model

For the non-dummy approach, the CNN model's performance exhibited variations depending on the number of target classes present in the dataset.

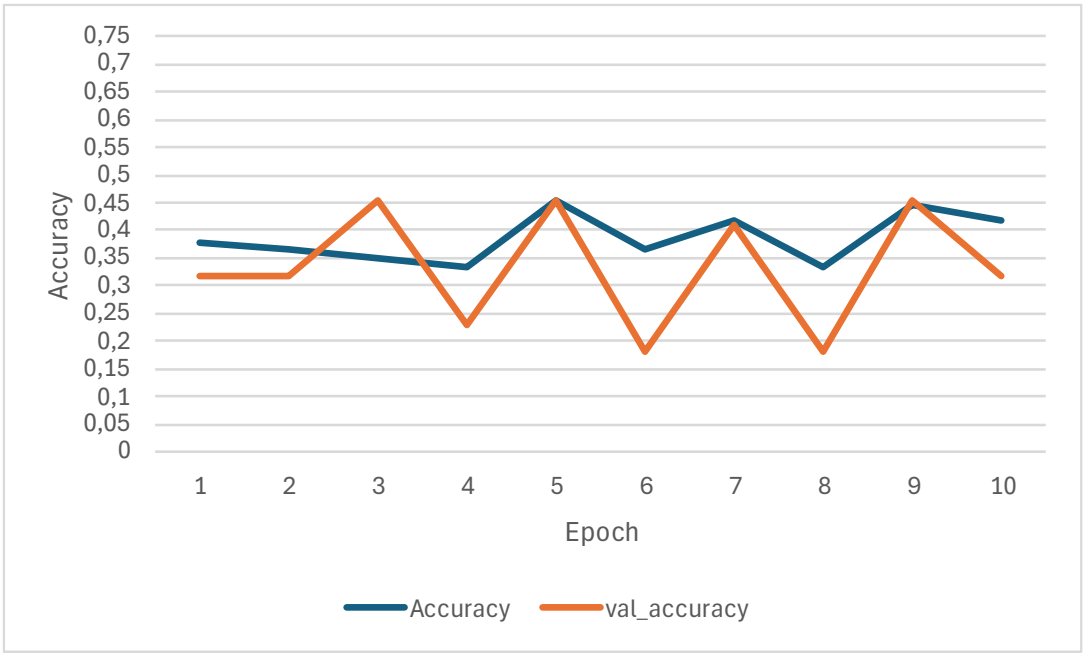
The model achieved the lowest average test accuracy within the cross-validation folds (0.1875) when dealing with all target values (dataset 1). This suggests the model struggled with the complexity of a larger number of classes. As the number of target classes decreased, the Test Accuracy improved significantly. In the three-target values (dataset 2) scenario, the Test Accuracy reached 0.3182, and it climbed further to 0.5000 for the binary classification case (dataset 3). This trend indicates the model performs better with fewer target classes.

While the training loss generally decreased across epochs for all scenarios, suggesting the model learned from the data, there were fluctuations in the loss curves. These fluctuations could point to potential challenges with overfitting or underfitting. Additionally, the validation accuracy didn't always show a clear upward trend. This suggests that further hyperparameter tuning might be necessary to improve the model's ability to generalize to unseen data.

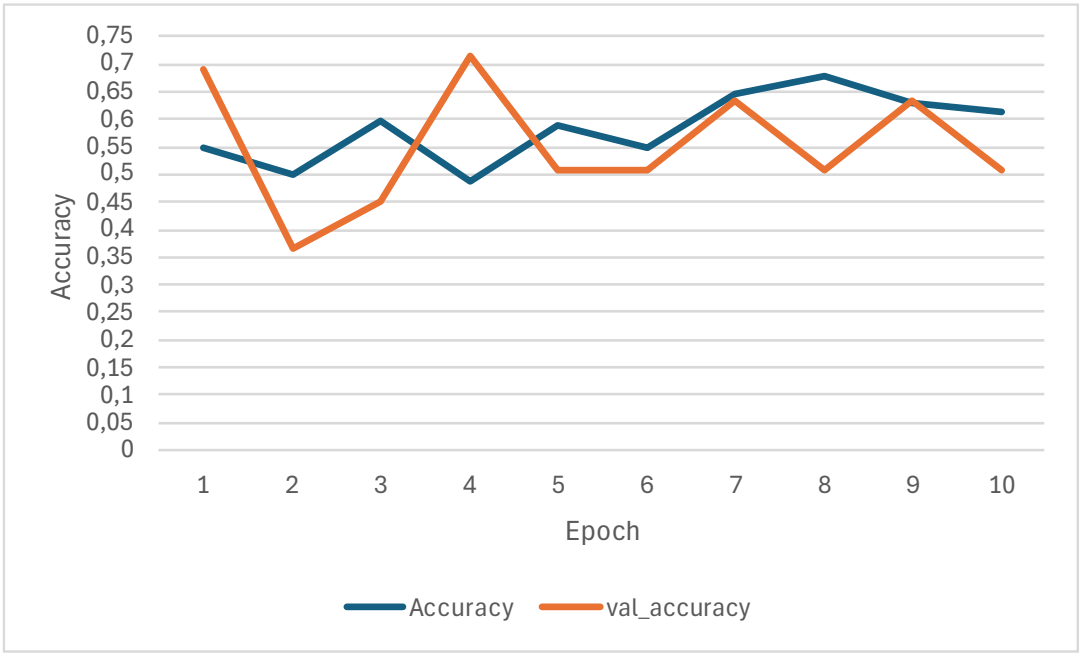
Overall, the CNN model seems to struggle with a larger number of target classes. Its performance improves as the number of target classes decreases, reaching its peak in the binary classification scenario. This suggests the model's architecture or training strategy might require adjustments to handle multi-class classification tasks more effectively.



(a)



(b)



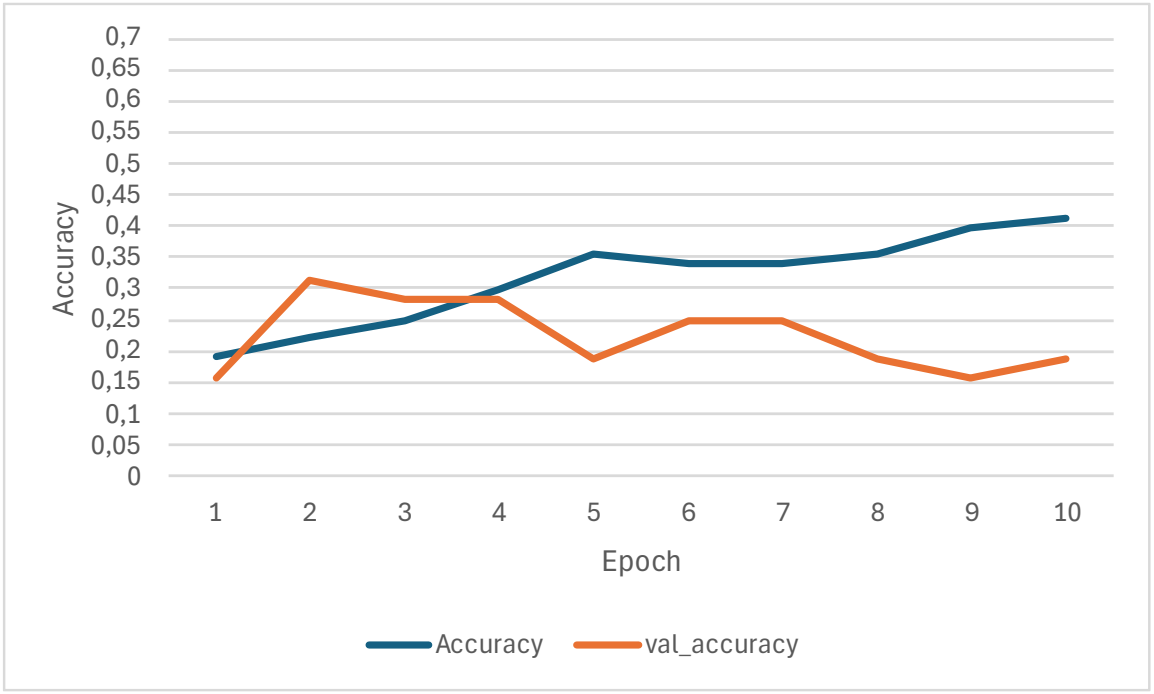
(c)

Figure 25: Training and testing sessions of the CNN model for (a) the full dataset (dataset 1), (b)for the three-target values dataset (dataset 2) and (c) for the binary dataset (dataset 3). Accuracy refers to the accuracy achieved during the training session, while val_accuracy refers to the accuracy achieved using the validation set.

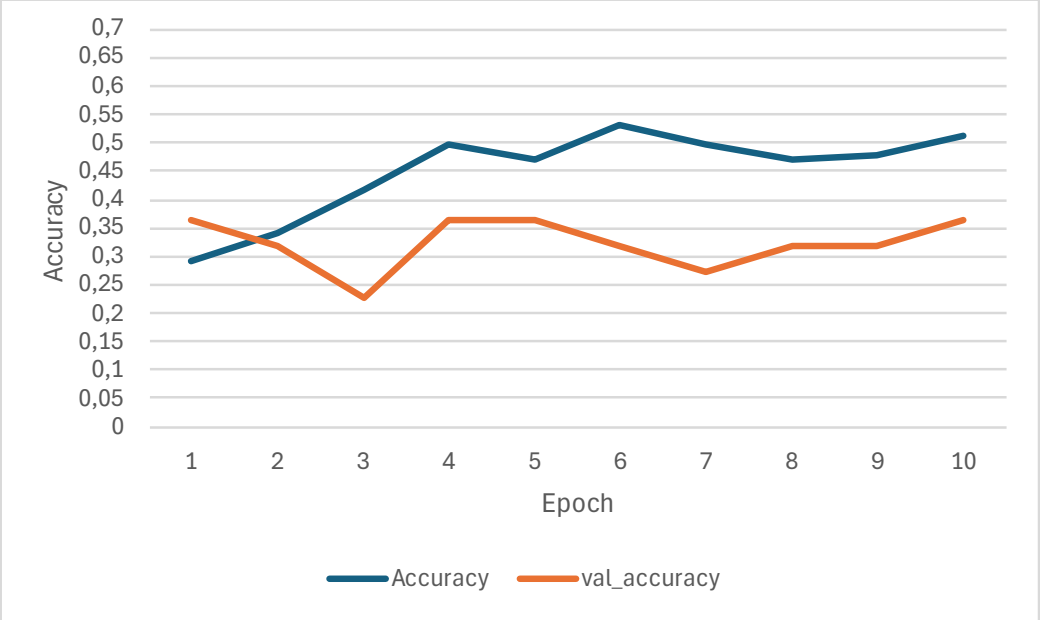
4.7 LSTM model

The LSTM model's performance exhibited a clear correlation between the complexity of the target variable and its ability to learn effectively. When faced with the full target values, the model achieved a low initial training accuracy of around 19%, only

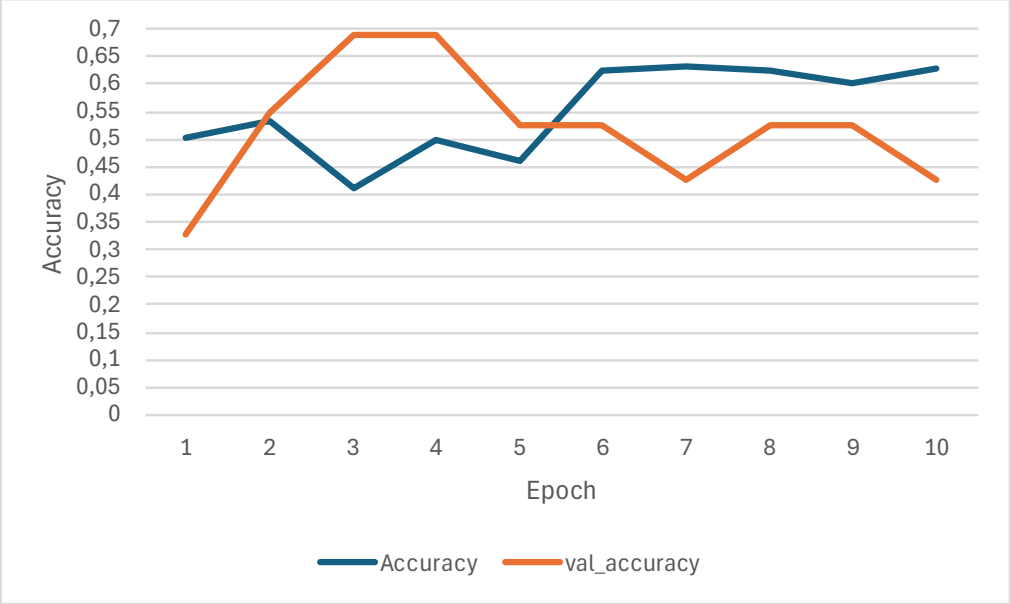
increasing to 41% by the end of 10 epochs. Similarly, validation accuracy remained stagnant, fluctuating between 15.62% and 18.75%. This lack of improvement was mirrored in the loss function. Training loss decreased slightly from 1.4204 to 1.3095, and validation loss showed minimal change from 1.4169 to 1.3609. These metrics suggest the model struggled with the complexity of the full target task. Conversely, reducing the number of targets led to a significant improvement in performance. With three target values, training accuracy rose from 29.06% to 51.28%, with validation accuracy also demonstrating a positive trend, ending at 36.36%. Training and validation loss followed a similar pattern, indicating the model was successfully learning the relationships within the data. The most successful application was observed in the binary classification task. Here, the model achieved the highest accuracy, starting at 50% and reaching 61.40% after 10 epochs. Validation accuracy also improved significantly, peaking at 68.18%, although it exhibited some fluctuations. The loss function mirrored this trend, with training loss decreasing from 0.7310 to 0.6763. While validation loss showed an increase at the end, these metrics demonstrate the model's superior performance on the binary task. By analysing these results, we can determine the appropriate scenarios where this LSTM model can be effectively applied (i.e., simpler classification problems) and potentially explore methods to improve its performance on more complex tasks with a large number of target variables.



(a)



(b)



(c)

Figure 26: Training and testing sessions of the LSTM model for (a) the full dataset (dataset 1), (b)for the three-target values dataset (dataset 2) and (c) for the binary dataset (dataset 3). Accuracy refers to the accuracy achieved during the training session, while val_accuracy refers to the accuracy achieved using the validation set.

4.8 Best model and feature importance

Examining the full target dataset (dataset 1), where categorical features were addressed using the dummy approach, the SVM emerged as the most effective model for this classification task. Its accuracy of 0.67 surpassed the runner-up, the

Random Forest model, by a margin of 0.03 (detailed model accuracies can be found in Table 8).

Intriguingly, for the SVM model the most impactful feature, emerging buy compering the f_values from the ANOVA test that is used by SelectKBest, was SF-36_GH-N, which corresponds to the general health section within the SF-36 health questionnaire (Figure 28). The second most important feature was SF-36_BP-N (bodily pain), and the third most important feature was the HADS anxiety questionnaire.

Table 8: Summary table of model accuracies for the dataset 1. SVM: Support Vector Machine, RF: Random Forest, DT: Decision Tree, KNN: K- Nearest Neighbours.

ACCURACY SUMMARY TABLE FOR DATASET 1						
Non-dummy handling approach						
SVM	RF	DT	KNN	XGBoost	CNN	LSTM
0.63	0.55	0.46	0.54	0.53	0.18	0.19

Dummy handling approach				
SVM	RF	DT	KNN	XGBoost
0.67	0.64	0.51	0.60	0.59

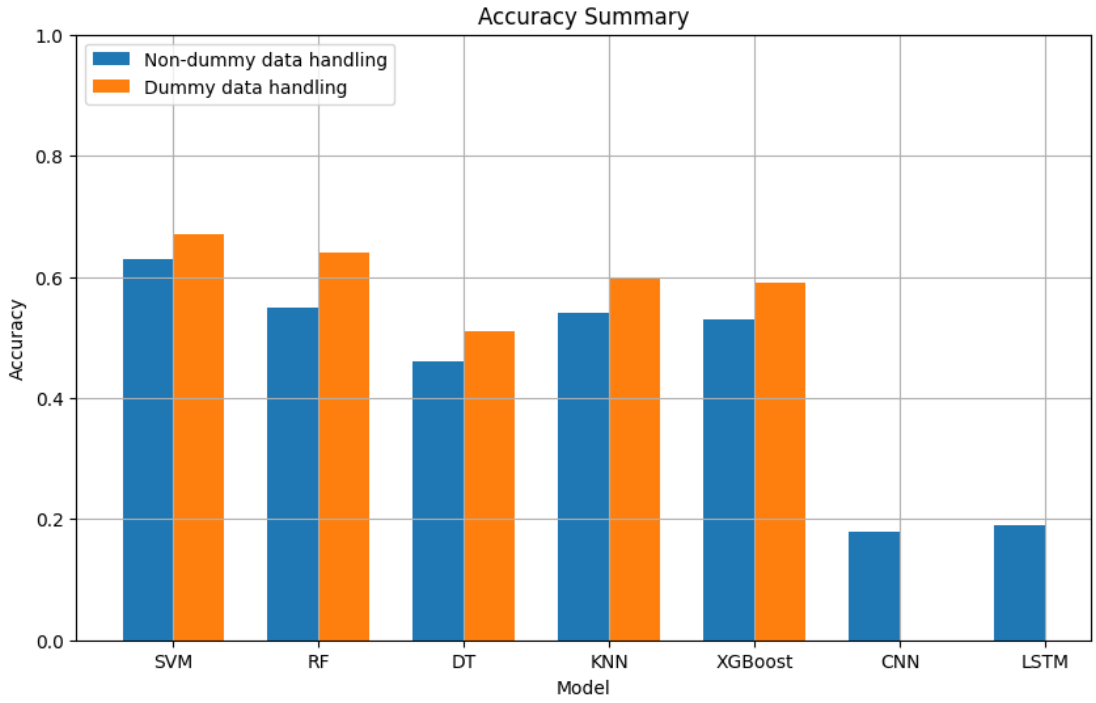


Figure 27: Visual representation of the Table 12. SVM: Support Vector Machine, RF: Random Forest, DT: Decision Tree, KNN: K- Nearest Neighbours.

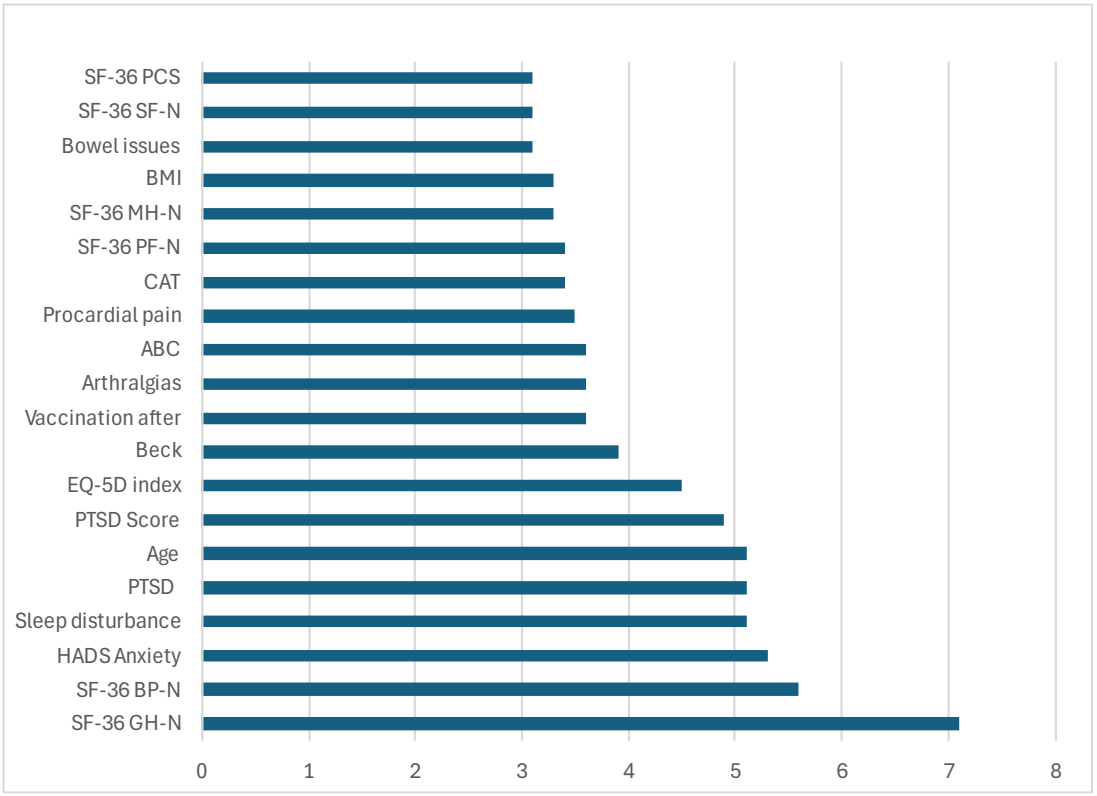


Figure 28: Important features selected by the SVM model when missing values handled with the dummy approach.

4.9 Metric scores per class for the best model

The evaluation metrics, alongside the confusion matrix, see Figure 29, reveal interesting class-specific performance variations in the model. Class 0.0, the class referring to patients labelled as they do not need any further specialist consultation, stands out with the highest precision (0.69) as shown in table 9. This indicates a strong ability to correctly identify relevant instances of class 0.0 and limit false positives, which can be further confirmed by its high recall of 0.81. Analysing the confusion, we can see that a significant portion of class 0.0 instances were indeed classified correctly.

Conversely, class 2.0 shows the lowest performance. Despite a moderate precision of 0.56 (indicating the model isn't including a substantial number of irrelevant instances in this class), the recall of 0.42 is concerning. This low recall, evident from the confusion matrix , suggests the model is missing a significant number of relevant class 2.0 examples.

Classes 1.0 and 3.0 showcase a trade-off between precision and recall. Class 1.0 exhibits balanced performance with a precision of 0.57 and a recall of 0.58. This suggests the model performs moderately well for this class, as reflected in the confusion matrix (likely a balance between correct classifications on the diagonal and misclassifications elsewhere). It identifies relevant instances of class 1.0 while avoiding a high number of irrelevant ones.

On the other hand, class 3.0 achieves the highest recall (0.87), indicating a strong ability to capture most of the relevant examples, as confirmed by the high value on the diagonal in the confusion matrix for class 3.0. However, the precision of 0.81 suggests there might be some inclusion of irrelevant instances in this category. Depending on the specific context of the thesis and the relative importance of precision versus recall, this trade-off for class 3.0 may require further consideration.

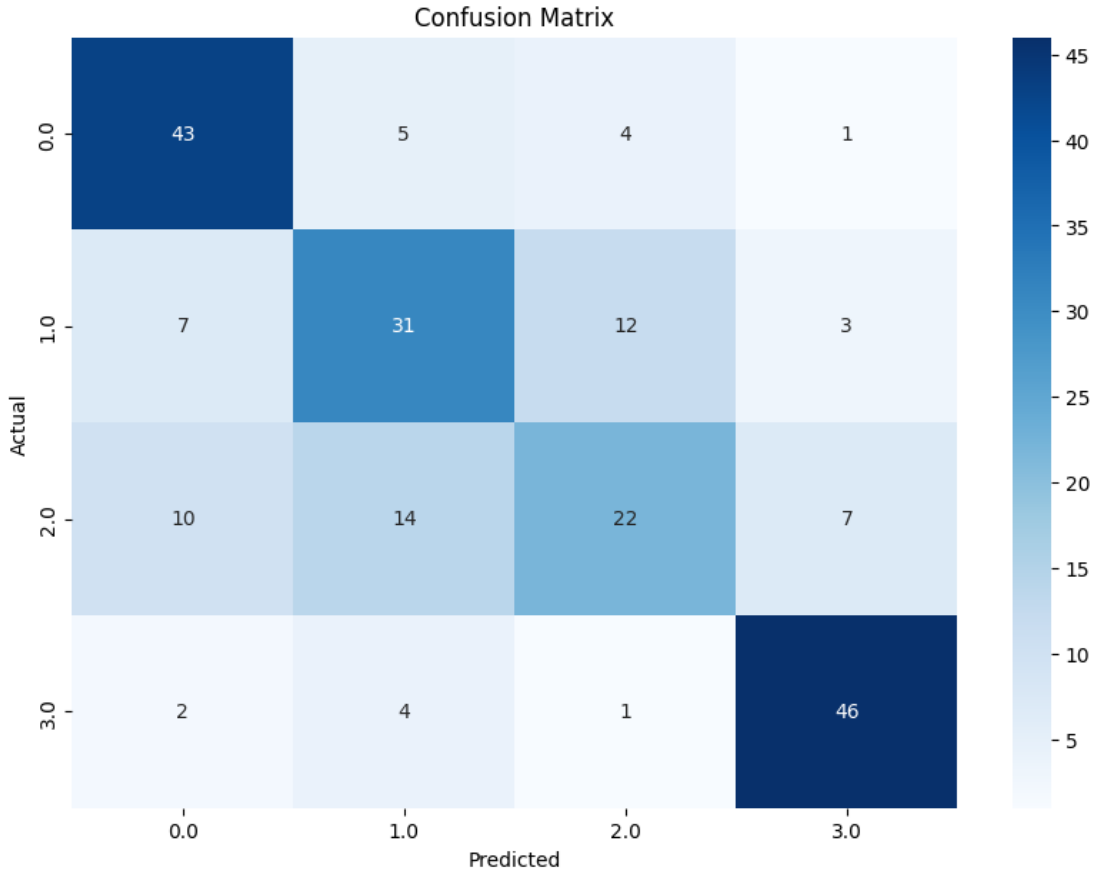


Figure 29: Confusion matrix of the SVM model.

Table 9: Metric scores per class for the SVM model.

Class	Precision	Recall	F1-score
0.0	0.69	0.81	0.74
1.0	0.57	0.58	0.57
2.0	0.56	0.42	0.48
3.0	0.81	0.87	0.84

5. Discussion and conclusion

In this thesis, the interrogated dataset was acquired at the Long Covid clinic of Evangelismos General Hospital, as part of an ongoing research project that examines holistically Long Covid patients. The analysis showed that different models were the most appropriate for different datasets. All models that emerged as the best, used the dummy variables data handling approach to address the categorical features. This indicates that categorical features should be converted to dummy variables and not being used as they are [65]

A trend emerged where model complexity mirrored dataset complexity. For Datasets 2 and 3, exhibiting lower complexity, the Random Forest model performed most effectively. Conversely, the SVM algorithm provided a promising foundation for Long Covid patient classification in the more intricate Dataset 1. Importantly, we cannot attribute the model discrepancies to model parameter variations between the different datasets. The most likely explanation for the differential performance lies in the inherent complexity of the datasets themselves.

Analysis of various features associated with Long Covid reveals a hierarchy of importance for patient classification, as illustrated in Figure 28. Here, we delve deeper into these prominent features.

A potential high link between BMI and Long Covid severity emerges from Figure 28, aligning with research by Wu et al. (2021) [66]. This suggests that BMI may serve as an indicator of overall health, potentially influencing the intensity of Long Covid symptoms.

Figure 28 emphasizes the significant impact of vaccination after a Covid-19 infection on Long Covid development and severity. This finding is further corroborated by research from Mumtaz et al. (2022) [67]. It underscores the critical role of vaccination in mitigating the risk and severity of Long Covid.

The prominence of gastrointestinal problems and sleep disturbances in Long Covid patients is highlighted in Figure 28. Studies by Tedjasukmana et al. (2023) and Álvarez-Santacruz et al. (2024) provide deeper insights into these specific aspects [68], [69]. Additionally, Vélez et. al (2023) contributes valuable knowledge in this domain [70].

The HADS questionnaire, as showed in Figure 28, was identified as an important feature in classifying patients. The association between anxiety and Long Covid was studied by Mazza et al. (2022) [71]. The study shows a high association between anxiety and Long Covid development.

Also, there is a high association between arthralgias and Long Covid, as showed in Figure 28. Research by Nalbandian et al. (2021) [72] supports this association.

Moreover, the SF-36 family of questinnaires, depicted in Figure 28, provides valuable insights into various mental and physical health aspects relevant to Long Covid. A study by Rodríguez-Galán et al. (2022) demonstrates the utility of SF-36 for assessing Long Covid patients [73]. Scores on specific subscales like Mental Health (MH-N),

Physical Functioning (PF-N), and General Health (GH-N) could be particularly informative.

By prioritizing these key features and leveraging insights from the cited studies, we can develop more robust classification models, ultimately leading to improved diagnosis and management strategies for Long Covid.

While some features like fatigue and breathlessness have emerged as strong indicators of Long Covid, others require a closer look. Let's delve into two such features:

One of the features that need a closer look is chest pain. While chest pain is a recognized symptom of Long Covid, it might not be as specific as others. Focusing on the exact type of chest pain experienced by patients could be more informative. For instance, studies by Ziauddeen et al. (2022) offer valuable insights into the characteristics of chest pain in Long Covid patients [74].

Another feature is PTSD. The relevance of PTSD scores and specific PTSD subcategories for Long Covid classification needs further investigation. Studies by Aceituno et al. (2024) explore the relationship between these conditions in Covid-19 patients [75].

While the SVM model, using the dummy handling approach achieved a promising accuracy of 0.67, further exploration can lead to significant improvements. Additional investigation is warranted for features like specific types of chest pain (building on Ziauddeen et al., 2022 [74]) and particular PTSD subcategories (as explored by Aceituno et al., 2024 [75]) to potentially enhance model performance. Integrating data from electronic health records, imaging tests, and genetics could also provide a more holistic view of patient health, potentially leading to even more robust patient classification.

Also, after discussing the above features with our collaborators at the Evangelismos General Hospital, we concluded that patients should have the Covid-19 vaccine, since it is the only way to reduce the risk for developing Long Covid. The rest of the symptoms emerge because Long Covid impacts multiple physiological systems. We cannot reduce the risk for Long Covid by preventing different symptoms. For example, trying to sleep better will not prevent Long Covid.

This study acknowledges some limitations. The current accuracy of 0.67 indicates room for improvement. Future research should explore techniques like hyperparameter tuning and feature engineering to refine the model. Additionally, the generalisability of the findings may be limited by the specific patient population and data sources used. Further research with diverse populations and healthcare settings is crucial for broader applicability.

Our model aims to become a standard process assessing patients at their first visit at the Long Covid clinic. This can be done by creating an application with a simple questionnaire containing the most important features, as described in section 4.8. After the patient completes the questionnaire, the application will send the results proposing the medical specialty that is best suited for the patients' needs. This way we can streamline the triaging process in the clinic.

Several exciting possibilities emerge for future research. Data integration from various sources, including electronic health records, imaging, and genetic information, could offer a more comprehensive picture of patient health. Broader population studies across different geographical locations and healthcare systems are necessary to verify the generalizability of the findings. Continuous model improvement through techniques like hyperparameter tuning and feature engineering is essential to maintain optimal performance in a constantly evolving environment.

One of the challenges in developing machine learning models is the need for large, robust datasets. This thesis proposes a novel approach to address this challenge by leveraging advancements in Artificial Intelligence. A recent study by Nikolopoulos et al. (2024) demonstrates the effectiveness of Wasserstein Generative Adversarial Networks (WGANs) in generating synthetic patient data [76]. WGANs offer a significant advantage in terms of speed, allowing for the creation of a large volume of data in a shorter timeframe. Furthermore, WGANs are designed to capture the underlying distribution of the real patient data, ensuring statistical similarity between the real and synthetic populations. This approach not only expedites data acquisition but also mitigates the risk of type II errors, which occur when a true effect goes undetected due to insufficient data points. Utilising a combination of real-world and virtual patient data can lead to a robust dataset size for training and validating the model.

This thesis contributes significantly to the growing body of research on the application of machine learning in healthcare. By demonstrating the feasibility and effectiveness of machine learning for prioritising Long Covid patients, this study paves the way for more efficient and targeted approaches to managing this complex condition. As future research refines the model and expands its generalisability, ML has the potential to revolutionize Long Covid patient care, leading to improved outcomes for patients and optimised resource utilisation within healthcare systems. This approach ultimately empowers clinicians to deliver more focused and timely interventions, ensuring increased safety for Long Covid patients.

REFERENCES

[1] J. Coste *et al.*, “Prevalence of long COVID in the general adult population according to different definitions and sociodemographic and infection characteristics. A nationwide random sampling survey in France in autumn 2022,” *Clinical Microbiology and Infection*, Mar. 2024, doi: 10.1016/j.cmi.2024.03.020.

[2] WHO, “Origin of SARS-CoV-2 .” Accessed: Mar. 27, 2024. [Online]. Available: https://apps.who.int/iris/bitstream/handle/10665/332197/WHO-2019-nCoV-FAQ-Virus_origin-2020.1-eng.pdf

[3] Centers for Disease Control and Prevention, “Coronavirus Disease 2019 (COVID-19) – Symptoms.” Accessed: Jun. 25, 2024. [Online]. Available: https://www.cdc.gov/coronavirus/2019-ncov/symptoms-testing/symptoms.html

[4] S. J. R. da Silva *et al.*, “Two Years into the COVID-19 Pandemic: Lessons Learned,” *ACS Infect Dis*, vol. 8, no. 9, pp. 1758–1814, Sep. 2022, doi: 10.1021/acsinfecdis.2c00204.

[5] Centers for Disease Control and Prevention, “Long Covid or Post-Covid Conditions.” Accessed: Jun. 25, 2024. [Online]. Available: www.cdc.gov/coronavirus/2019-ncov/long-term-effects/index.html

[6] M.-S. Katsarou *et al.*, “The Greek Collaborative Long COVID Study: Non-Hospitalized and Hospitalized Patients Share Similar Symptom Patterns,” *J Pers Med*, vol. 12, no. 6, p. 987, Jun. 2022, doi: 10.3390/jpm12060987.

[7] NIH COVID-19 Research, “Long COVID.” Accessed: Jun. 25, 2024. [Online]. Available: https://covid19.nih.gov/covid-19-topics/long-covid

[8] M. Kobusiak-Prokopowicz *et al.*, “Cardiovascular, Pulmonary, and Neuropsychiatric Short- and Long-Term Complications of COVID-19,” *Cells*, vol. 11, no. 23, p. 3882, Dec. 2022, doi: 10.3390/cells11233882.

[9] www.caymanchem.com, “Research Tools for Long COVID Syndrome.” Accessed: Apr. 12, 2024. [Online]. Available: https://www.caymanchem.com/news/research-tools-for-long-covid-syndrome

[10] M. Jha, R. Gupta, and R. Saxena, “A Precise Method to Detect Post-COVID-19 Pulmonary Fibrosis Through Extreme Gradient Boosting,” *SN Comput Sci*, vol. 4, no. 1, p. 89, Dec. 2022, doi: 10.1007/s42979-022-01526-x.

[11] E. R. Pfaff *et al.*, “Identifying who has long COVID in the USA: a machine learning approach using N3C data,” *Lancet Digit Health*, vol. 4, no. 7, pp. e532–e541, Jul. 2022, doi: 10.1016/S2589-7500(22)00048-6.

[12] S. Jiang, J. Loomba, S. Sharma, D. Brown, RECOVER consortium, and N3C consortium, “Vital Measurements of Hospitalized COVID-19 Patients as a Predictor of Long COVID: An EHR-based Cohort Study from the RECOVER Program in N3C.,” *ArXiv*, Nov. 2022.

[13] C. H. Sudre *et al.*, “Attributes and predictors of long COVID,” *Nat Med*, vol. 27, no. 4, pp. 626–631, Apr. 2021, doi: 10.1038/s41591-021-01292-y.

[14] R. Subramanian, R. Rubi, M. Dedeepya, and S. Gorugantu, “Quantitative Progression analysis of Post-Acute Sequelae of COVID-19, Pulmonary Fibrosis (PASC-PF) and Artificial Intelligence driven CT scoring of Lung involvement in Covid-19 infection using HRCT-Chest images,” *Med Res Arch*, vol. 10, no. 10, 2022, doi: 10.18103/mra.v10i10.3145.

[15] M. Binka *et al.*, “An Elastic Net Regression Model for Identifying Long COVID Patients Using Health Administrative Data: A Population-Based Study,” *Open Forum Infect Dis*, vol. 9, no. 12, Dec. 2022, doi: 10.1093/ofid/ofac640.

[16] E. Hill *et al.*, “Risk Factors Associated with Post-Acute Sequelae of SARS-CoV-2 in an EHR Cohort: A National COVID Cohort Collaborative (N3C) Analysis as part of the NIH RECOVER program.,” *medRxiv*, Aug. 2022, doi: 10.1101/2022.08.15.22278603.

[17] O. Moreno-Pérez *et al.*, “Post-acute COVID-19 syndrome. Incidence and risk factors: A Mediterranean cohort study,” *Journal of Infection*, vol. 82, no. 3, pp. 378–383, Mar. 2021, doi: 10.1016/j.jinf.2021.01.004.

[18] M. A. Patel *et al.*, “Organ and cell-specific biomarkers of Long-COVID identified with targeted proteomics and machine learning,” *Molecular Medicine*, vol. 29, no. 1, p. 26, Feb. 2023, doi: 10.1186/s10020-023-00610-z.

[19] H. Zhang *et al.*, “Data-driven identification of post-acute SARS-CoV-2 infection subphenotypes,” *Nat Med*, vol. 29, no. 1, pp. 226–235, Jan. 2023, doi: 10.1038/s41591-022-02116-3.

[20] Y. Zhu *et al.*, “Using natural language processing on free-text clinical notes to identify patients with long-term COVID effects,” in *Proceedings of the 13th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, New York, NY, USA: ACM, Aug. 2022, pp. 1–9. doi: 10.1145/3535508.3545555.

[21] L. Wang, D. Foer, E. MacPhaul, Y.-C. Lo, D. W. Bates, and L. Zhou, “PASCLex: A comprehensive post-acute sequelae of COVID-19 (PASC) symptom lexicon derived from electronic health record clinical notes,” *J Biomed Inform*, vol. 125, p. 103951, Jan. 2022, doi: 10.1016/j.jbi.2021.103951.

[22] J. M. Banda *et al.*, “Characterization of long-term patient-reported symptoms of COVID-19: an analysis of social media data,” *medRxiv*, p. 2021.07.13.21260449, Jan. 2021, doi: 10.1101/2021.07.13.21260449.

[23] J. Bell, “What Is Machine Learning?,” in *Machine Learning and the City*, Wiley, 2022, pp. 207–216. doi: 10.1002/9781119815075.ch18.

[24] A. L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers,” *IBM J Res Dev*, vol. 3, no. 3, pp. 210–229, Jul. 1959, doi: 10.1147/rd.33.0210.

[25] Tom M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[26] Sandhya N. dhage and Charanjeet Kaur Raina, “A review on Machine Learning Techniques,” *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 4, no. 3, pp. 395–399, Mar. 2016.

[27] T. Chen and C. Guestrin, “XGBoost,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[28] J. R. Quinlan, “Induction of Decision Trees,” *Mach Learn*, vol. 1, no. 1, pp. 81–106, 1986, doi: 10.1023/A:1022643204877.

[29] J. Pickard, “Decision Trees,” Geek Culture. Accessed: Jun. 25, 2024. [Online]. Available: medium.com/geekculture/decision-trees-adb32c47c8ef

[30] Matthew N. Bernstein, “Random Forests.”

[31] S. Li, J. Y. C. Ting, and A. S. Barnard, “Optimization-Free Inverse Design of High-Dimensional Nanoparticle Electrocatalysts Using Multi-target Machine Learning,” 2022, pp. 307–318. doi: 10.1007/978-3-031-08754-7_39.

[32] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach Learn*, vol. 20, no. 3, pp. 273–297, Sep. 1995, doi: 10.1007/BF00994018.

[33] A. Brital, “Introduction to Support Vector Machine,” Medium. Accessed: May 05, 2024. [Online]. Available: medium.com/@AnasBrital98/introduction-to-support-vector-machine-2a2091401858

[34] L. A. Garcia-Escudero and A. Gordaliza, “Robustness Properties of k Means and Trimmed k Means,” *J Am Stat Assoc*, vol. 94, no. 447, p. 956, Sep. 1999, doi: 10.2307/2670010.

[35] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE Trans Inf Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967, doi: 10.1109/TIT.1967.1053964.

[36] D. Roi. Yehoshua, “K-Nearest Neighbors (KNN): A Comprehensive Guide,” AI Made Simple. Accessed: May 05, 2024. [Online]. Available: medium.com/ai-made-simple/k-nearest-neighbors-knn-a-comprehensive-guide-7add717806ad

[37] Kevin P. Murphy, *Machine Learning : A Probabilistic Perspective*. Cambridge, Massachussets: The MIT Press, 2012.

[38] M. M. Taye, “Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions,” *Computation*, vol. 11, no. 3, p. 52, Mar. 2023, doi: 10.3390/computation11030052.

[39] F. Emmert-Streib, Z. Yang, H. Feng, S. Tripathi, and M. Dehmer, “An Introductory Review of Deep Learning for Prediction Models With Big Data,” *Front Artif Intell*, vol. 3, Feb. 2020, doi: 10.3389/frai.2020.00004.

[40] K. D, “Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning,” Medium. Accessed: May 05, 2024. [Online]. Available:

medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48

[41] P. Sharma, “Different Types of Cross-Validations in Machine Learning,” Analytics Vidhya. Accessed: Jun. 25, 2024. [Online]. Available: www.analyticsvidhya.com/blog/2022/02/different-types-of-cross-validations-in-machine-learning/

[42] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York, NY: Springer US, 2021. doi: 10.1007/978-1-0716-1418-1.

[43] Aurélien Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. O’Reilly Media, Inc., 2019.

[44] O. P. Barus, Romindo, and Jefri Junifer Pangaribuan, “Classification of Hearing Loss Degrees with Naive Bayes Algorithm,” *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 7, no. 4, pp. 751–757, Aug. 2023, doi: 10.29207/resti.v7i4.4683.

[45] Margherita Grandini, Enrico Bagli, and Giorgio Visani, “Metrics for Multi-Class Classification: an Overview,” Aug. 2020, doi: https://doi.org/10.48550/arXiv.2008.05756.

[46] developers google, “Classification: ROC Curve and AUC,” developers google. Accessed: Jun. 25, 2024. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

[47] S. Narkhede, “Understanding AUC - ROC Curve,” Medium. Accessed: May 06, 2024. [Online]. Available: , towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

[48] L. Lins and F. M. Carvalho, “SF-36 total score as a single measure of health-related quality of life: Scoping review,” *SAGE Open Med*, vol. 4, p. 205031211667172, Jan. 2016, doi: 10.1177/2050312116671725.

[49] I. Michopoulos *et al.*, “Hospital Anxiety and Depression Scale (HADS): validation in a Greek general hospital sample,” *Ann Gen Psychiatry*, vol. 7, no. 1, p. 4, Dec. 2008, doi: 10.1186/1744-859X-7-4.

[50] M. Hagströmer, P. Oja, and M. Sjöström, “The International Physical Activity Questionnaire (IPAQ): a study of concurrent and construct validity,” *Public Health Nutr*, vol. 9, no. 6, pp. 755–762, Sep. 2006, doi: 10.1079/PHN2005898.

[51] N. Gusi, P. R. Olivares, and R. Rajendram, “The EQ-5D Health-Related Quality of Life Questionnaire,” in *Handbook of Disease Burdens and Quality of Life Measures*, New York, NY: Springer New York, 2010, pp. 87–99. doi: 10.1007/978-0-387-78665-0_5.

[52] Mike Polkey, Claus Vogelmeier, and Mark Dransfield, “CAT User Guide & FAQs.” May 06, 2022.

[53] E. ERTAN YAZAR, E. Y. NIKSARLIOGLU, B. YIGITBAS, and M. BAYRAKTAROGLU, “How to Utilize CAT and mMRC Scores to Assess Symptom Status of Patients with COPD in Clinical Practice?,” *Medeni Med J*, vol. 37, no. 2, pp. 173–179, Jun. 2022, doi: 10.4274/MMJ.galenos.2022.06787.

[54] G. B. Neuberger, “Measures of fatigue: The Fatigue Questionnaire, Fatigue Severity Scale, Multidimensional Assessment of Fatigue Scale, and Short Form-36 Vitality (Energy/Fatigue) Subscale of the Short Form Health Survey,” *Arthritis Rheum*, vol. 49, no. S5, pp. S175–S183, Oct. 2003, doi: 10.1002/art.11405.

[55] G. Dautzenberg, J. Lijmer, and A. Beekman, “Diagnostic accuracy of the Montreal Cognitive Assessment (MoCA) for cognitive screening in old age psychiatry: Determining cutoff scores in clinical practice. Avoiding spectrum bias caused by healthy controls,” *Int J Geriatr Psychiatry*, vol. 35, no. 3, pp. 261–269, Mar. 2020, doi: 10.1002/gps.5227.

[56] L. E. Powell and A. M. Myers, “The Activities-specific Balance Confidence (ABC) Scale,” *J Gerontol A Biol Sci Med Sci*, vol. 50A, no. 1, pp. M28–M34, Jan. 1995, doi: 10.1093/gerona/50A.1.M28.

[57] J. Chilcot, S. Norton, M. E. Kelly, and R. Moss-Morris, “The Chalder Fatigue Questionnaire is a valid and reliable measure of perceived fatigue severity in multiple sclerosis,” *Multiple Sclerosis Journal*, vol. 22, no. 5, pp. 677–684, Apr. 2016, doi: 10.1177/1352458515598019.

[58] K. Webster, D. Cella, and K. Yost, “The Functional Assessment of Chronic Illness Therapy (FACIT) Measurement System: properties, applications, and interpretation,” *Health Qual Life Outcomes*, vol. 1, no. 1, p. 79, 2003, doi: 10.1186/1477-7525-1-79.

[59] P. K. McCutchan, M. C. Freed, E. C. Low, B. E. Belsher, and C. C. Engel, “Rescaling the Post-Traumatic Stress Disorder Checklist for Use in Primary Care,” *Mil Med*, vol. 181, no. 9, pp. 1002–1006, Sep. 2016, doi: 10.7205/MILMED-D-15-00444.

[60] A.-M. Aalto, M. Elovainio, M. Kivimäki, A. Uutela, and S. Pirkola, “The Beck Depression Inventory and General Health Questionnaire as measures of depression in the general population: A validation study using the Composite International Diagnostic Interview as the gold standard,” *Psychiatry Res*, vol. 197, no. 1–2, pp. 163–171, May 2012, doi: 10.1016/j.psychres.2011.09.008.

[61] M. Tsekoura, K. Fousekis, E. Billis, Y. Dionyssiotis, and E. Tsepis, “Cross-cultural adaptation of the Greek version of post-COVID-19 Functional Status Scale: assessment of non-hospitalised post-COVID-19 survivors,” *Eur J Transl Myol*, vol. 33, no. 2, Jun. 2023, doi: 10.4081/ejtm.2023.11328.

[62] Miriam Santos, “A Data Scientist’s Essential Guide to Exploratory Data Analysis,” Towards Data Science. Accessed: Jun. 26, 2024. [Online]. Available: https://towardsdatascience.com/a-data-scientists-essential-guide-to-exploratory-data-analysis-25637eee0cf6

[63] Clay Ford, “Understanding QQ Plots,” University of Virginia: Library. Accessed: Jun. 26, 2024. [Online]. Available: https://library.virginia.edu/data/articles/understanding-q-q-plots

[64] Cory Maklin, “Synthetic Minority Over-sampling TEchnique (SMOTE),” Medium. Accessed: Jun. 27, 2024. [Online]. Available: https://medium.com/@corymaklin/synthetic-minority-over-sampling-technique-smote-7d419696b88c

[65] R. Nisbet, G. Miner, and K. Yale, “Data Understanding and Preparation,” in *Handbook of Statistical Analysis and Data Mining Applications*, Elsevier, 2018, pp. 63–64. doi: 10.1016/B978-0-12-416632-5.00004-9.

[66] X. Wu *et al.*, “Association of body mass index with severity and mortality of COVID-19 pneumonia: a two-center, retrospective cohort study from Wuhan, China,” *Aging*, vol. 13, no. 6, pp. 7767–7780, Mar. 2021, doi: 10.18632/aging.202813.

[67] A. Mumtaz *et al.*, “COVID-19 Vaccine and Long COVID: A Scoping Review,” *Life*, vol. 12, no. 7, p. 1066, Jul. 2022, doi: 10.3390/life12071066.

[68] R. Tedjasukmana, A. Budikayanti, W. R. Islamiyah, A. M. A. L. Witjaksono, and M. Hakim, “Sleep disturbance in post COVID-19 conditions: Prevalence and quality of life,” *Front Neurol*, vol. 13, Jan. 2023, doi: 10.3389/fneur.2022.1095606.

[69] C. Álvarez-Santacruz, S. D. Tyrkalska, and S. Candel, “The Microbiota in Long COVID,” *Int J Mol Sci*, vol. 25, no. 2, p. 1330, Jan. 2024, doi: 10.3390/ijms25021330.

[70] Christopher D. Vélez, “Can long COVID affect the gut?,” Harvard Health Publishing: Harvard Medical School. Accessed: Jun. 26, 2024. [Online]. Available: https://www.health.harvard.edu/blog/can-long-covid-affect-the-gut-202303202903

[71] M. G. Mazza, M. Palladini, S. Poletti, and F. Benedetti, “Post-COVID-19 Depressive Symptoms: Epidemiology, Pathophysiology, and Pharmacological Treatment,” *CNS Drugs*, vol. 36, no. 7, pp. 681–702, Jul. 2022, doi: 10.1007/s40263-022-00931-3.

[72] A. Nalbandian *et al.*, “Post-acute COVID-19 syndrome,” *Nat Med*, vol. 27, no. 4, pp. 601–615, Apr. 2021, doi: 10.1038/s41591-021-01283-z.

[73] I. Rodríguez-Galán, N. Albaladejo-Blázquez, N. Ruiz-Robledillo, J. F. Pascual-Lledó, R. Ferrer-Cascales, and J. Gil-Carbonell, “Impact of COVID-19 on Health-Related Quality of Life: A Longitudinal Study in a Spanish Clinical Sample,” *Int J Environ Res Public Health*, vol. 19, no. 16, p. 10421, Aug. 2022, doi: 10.3390/ijerph191610421.

[74] N. Ziauddeen *et al.*, “Characteristics and impact of Long Covid: Findings from an online survey,” *PLoS One*, vol. 17, no. 3, p. e0264331, Mar. 2022, doi: 10.1371/journal.pone.0264331.

[75] H. Aceituno, “Long COVID: Association of fear and loneliness with anxiety, depression, and post-traumatic stress disorder: A case-control study,” *Gac Med Caracas*, vol. 132, no. Supl. 1, Jan. 2024, doi: 10.47307/GMC.2024.132.s1.14.

[76] A. Nikolopoulos and V. D. Karalis, “Implementation of a Generative AI Algorithm for Virtually Increasing the Sample Size of Clinical Studies,” *Applied Sciences*, vol. 14, no. 11, p. 4570, May 2024, doi: 10.3390/app14114570.