Politécnica UFRJ

10-YEAR DEMENTIA PROGNOSIS: APPLYING MACHINE LEARNING MODELS AND EXPLAINABLE AI TECHNIQUES TO IDENTIFY MODIFIABLE RISK FACTORS

Rafael Moreira da Costa Nunes Ribeiro

Projeto de Graduação apresentado ao Curso de Engenharia Eletrônica e de Computação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores:
Ana Luiza Dallora Moraes
Marcello Luiz Rodrigues de Campos

Rio de Janeiro
Julho de 2025

10-YEAR DEMENTIA PROGNOSIS: APPLYING MACHINE LEARNING MODELS AND EXPLAINABLE AI TECHNIQUES TO IDENTIFY MODIFIABLE RISK FACTORS

Rafael Moreira da Costa Nunes Ribeiro

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRÔNICO E DE COMPUTAÇÃO

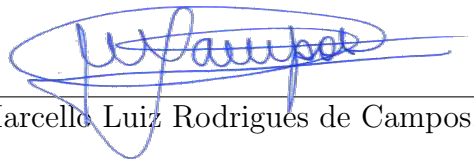
Autor:

Rafael Moreira da Costa Nunes Ribeiro

Orientador:

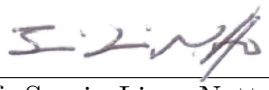
Profa. Ana Luiza Dallora Moraes, Ph. D.

Orientador:



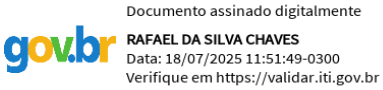
Prof. Marcello Luiz Rodrigues de Campos, Ph. D.

Examinador:



Prof. Sergio Lima Netto, Ph. D.

Examinador:



Prof. Rafael da Silva Chaves, Ph. D.

Rio de Janeiro

Julho de 2025

Declaração de Autoria e de Direitos

Eu, *Rafael Moreira da Costa Nunes Ribeiro* CPF *059.917.097-26*, autor da monografia *10-YEAR DEMENTIA PROGNOSIS: APPLYING MACHINE LEARNING MODELS AND EXPLAINABLE AI TECHNIQUES TO IDENTIFY MODIFIABLE RISK FACTORS*, subscrevo para os devidos fins, as seguintes informações:

1. O autor declara que o trabalho apresentado na disciplina de Projeto de Graduação da Escola Politécnica da UFRJ é de sua autoria, sendo original em forma e conteúdo.

2. Excetuam-se do item 1. eventuais transcrições de texto, figuras, tabelas, conceitos e idéias, que identifiquem claramente a fonte original, explicitando as autorizações obtidas dos respectivos proprietários, quando necessárias.

3. O autor permite que a UFRJ, por um prazo indeterminado, efetue em qualquer mídia de divulgação, a publicação do trabalho acadêmico em sua totalidade, ou em parte. Essa autorização não envolve ônus de qualquer natureza à UFRJ, ou aos seus representantes.

4. O autor pode, excepcionalmente, encaminhar à Comissão de Projeto de Graduação, a não divulgação do material, por um prazo máximo de 01 (um) ano, improrrogável, a contar da data de defesa, desde que o pedido seja justificado, e solicitado antecipadamente, por escrito, à Congregação da Escola Politécnica.

5. O autor declara, ainda, ter a capacidade jurídica para a prática do presente ato, assim como ter conhecimento do teor da presente Declaração, estando ciente das sanções e punições legais, no que tange a cópia parcial, ou total, de obra intelectual, o que se configura como violação do direito autoral previsto no Código Penal Brasileiro no art.184 e art.299, bem como na Lei 9.610.

6. O autor é o único responsável pelo conteúdo apresentado nos trabalhos acadêmicos publicados, não cabendo à UFRJ, aos seus representantes, ou ao(s) orientador(es), qualquer responsabilização/ indenização nesse sentido.

7. Por ser verdade, firmo a presente declaração.

Rafael Moreira da Costa Nunes Ribeiro

UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
Escola Politécnica - Departamento de Eletrônica e de Computação
Centro de Tecnologia, bloco H, sala H-217, Cidade Universitária
Rio de Janeiro - RJ CEP 21949-900

AGRADECIMENTO

Aos meus pais, Jancarla e João Luiz, por sempre me inspirarem, orientarem e apoiarem em todos os momentos da minha vida.

À professora Ana Luiza Dallora e ao professor Marcello Campos, por confiarem em mim e me orientarem ao longo do desenvolvimento deste projeto.

Aos professores Claudio Miceli e Heudson Mirandola, por me acolherem, ensinarem e por abrirem portas dentro e fora da universidade.

À Pauliina, pela compreensão, pelo apoio e pelo incentivo durante toda essa etapa.

À Beatriz, pela constante força, motivação e confiança transmitidas.

Aos meus amigos, pelo companheirismo, incentivo e pelas conversas que tornaram essa jornada mais leve e significativa.

A todos que contribuíram para que eu chegasse até aqui, o meu sincero agradecimento.

RESUMO

A demência engloba uma série de condições degenerativas que comprometem a memória, a cognição e o comportamento, sendo uma das principais causas de dependência entre idosos. Mais de 55 milhões de pessoas vivem atualmente com demência no mundo, com cerca de 10 milhões de novos casos a cada ano. O prognóstico precoce e a identificação de fatores de risco modificáveis são, portanto, cruciais tanto para o planejamento em saúde pública quanto para o cuidado individual. Este estudo avalia se modelos do tipo "caixa-preta", como XGBoost, CatBoost e Balanced Random Forest, podem melhorar a previsão de demência em um horizonte de dez anos no conjunto de dados do Swedish National Study on Aging and Care (SNAC), em comparação com modelos intrinsecamente explicáveis, preservando a interpretabilidade por meio do SHapley Additive exPlanations (SHAP). O fluxo de processamento integra imputação de valores ausentes pelo algoritmo dos K-vizinhos mais próximos (KNN), eliminação recursiva de atributos (RFE) e validação cruzada aninhada para ajuste de hiperparâmetros. Os resultados demonstram uma melhoria significativa em relação ao estudo base: a área sob a Curva Característica de Operação do Receptor (ROC AUC) obetve um aumento de aproximadamente 11 %, e a sensibilidade um aumento de aproximadamente 17 % com o uso do XGBoost. A análise com SHAP também permite identificar os fatores de risco mais influentes e potencialmente modificáveis. Esses resultados confirmam que modelos caixa-preta, quando combinados com técnicas de inteligência artificial explicável (XAI), podem oferecer desempenho preditivo superior sem abrir mão da explicabilidade, um requisito essencial na área médica e cada vez mais exigido por novas regulamentações de Inteligência Artificial (AI) que demandam transparência e responsabilidade em sistemas de apoio à decisão.

Palavras-chave: demência, aprendizado de máquina, inteligência artificial explicável, SHAP.

ABSTRACT

Dementia encompasses a range of degenerative conditions that impair memory, cognition, and behavior, and is one of the leading causes of dependency among the elderly. More than 55 million people currently live with dementia worldwide, with approximately 10 million new cases each year. Early prognosis and the identification of modifiable risk factors are therefore crucial for both public health planning and individual care. This study evaluates whether black-box models such as XGBoost, CatBoost, and Balanced Random Forest can improve ten-year dementia prediction in the Swedish National Study on Aging and Care (SNAC) dataset, compared to intrinsically interpretable models, while preserving interpretability through SHapley Additive exPlanations (SHAP). The pipeline integrates missing value imputation using the K-Nearest Neighbors (KNN) algorithm, Recursive Feature Elimination (RFE), and nested cross-validation for hyperparameter tuning. The results show a significant improvement over the baseline: the area under the receiver operating characteristics curve (ROC AUC) increases by approximately 11 %, and sensitivity increases by around 17 % with the use of XGBoost. SHAP analysis also helps identify the most influential and potentially modifiable risk factors. These findings confirm that black-box models, when combined with Explainable Artificial Intelligence (XAI) techniques, can offer superior predictive performance without sacrificing explainability, an essential requirement in the medical field and increasingly mandated by new Artificial Intelligence (AI) regulations demanding transparency and accountability in decision-making systems.

Key-words: dementia, machine learning, explainable artificial intelligence, SHAP.

Contents

Acronyms

AI Artificial Intelligence

AUC Area Under the Curve

BMI Body Mass Index

DNN Deep Neural Networks

EQ5D EuroQol 5-Dimensions questionnaire

FPR False Positive Rate

G-Mean Geometric Mean

KNN K-Nearest Neighbors

LIME Local Interpretable Model-agnostic Explanations

MCC Matthews correlation coefficient

ML Machine Learning

PCS12 Physical Component Score from the SF-12 health survey

PPV Positive Predictive Value

PR Precision-Recall

RFE Recursive Feature Elimination

ROC Receiver Operating Characteristic

SHAP SHapley Additive exPlanations

SNAC Swedish National Study on Aging and Care

TNR True Negative Rate

TPR True Positive Rate

XAI Explainable Artificial Intelligence

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Dementia, as defined by the World Health Organisation (WHO) [1], is a broad term that includes multiple degenerative conditions affecting memory, cognitive function, and behavior. It is the primary cause of dependency among elders, making caregiving both demanding and expensive.

Currently, more than 55 million people have dementia worldwide [2], and approximately 10 million new cases are recorded each year [1]; numbers that are expected to rise as life expectancy continues to grow. In this sense, prognosis and the identification of lifestyle changes that contribute to delaying or preventing this condition are of great value both in individual and governmental contexts.

A Machine Learning (ML) project is a structured process [3] in which data is analysed and used to develop a model that can make predictions, classify information, or uncover patterns based on input data [4]. The goal of a ML project is to solve a specific problem or improve decision-making by leveraging algorithms, called ML models, that learn from data [5].

One property of those models is their explainability. A given model is considered explainable if its predictions and "thought process" can be understood by either a technical or non technical audience. Explainable ML models are such as decision trees and K-Nearest Neighbors (KNN) algorithms. On the other hand,

non-explainable models, often called "black-box" models, might present a better performance, but can be opaque in their decision-making processes due to their complexity, making it difficult for non-technical users to understand how they arrive at specific conclusions. This behaviour can be seen in Deep Neural Networks (DNN) and ensemble models [3].

XAI is a set of tools and techniques designed to complement and work alongside traditional "black-box" AI and ML models. XAI addresses their lack of explainability by providing mechanisms to interpret, visualise, and explain the outputs of these models without altering their underlying training processes [6]. This is particularly important in critical domains such as healthcare, finance, and law, where understanding the rationale behind predictions or classifications is essential for ethical and responsible use [7].

Moreover, the increasing number of AI regulations, such as the "Roles and Responsibilities Framework for Artificial Intelligence in Critical Infrastructure " by the U.S. Department of Homeland Security [8] and the European Union AI Act, both released in 2024 [9], highlight the importance of explainability by requiring transparency and accountability in AI systems to ensure compliance and foster trust.

In a 2020 study by Dallora et al. [10], a decision tree was used to provide predictions of whether an individual will develop dementia in 10 years, given an initial condition of normal cognition, with the aim of identifying modifiable risk factors over the Swedish National Study on Aging and Care (SNAC) dataset [11]. SNAC is a rich multidisciplinary database derived from an ongoing longitudinal study[1] conducted in Sweden, containing basic information about the subjects, as well as social, medical, and psychological factors.

The decision tree was the selected model in Dallora's work given the project's need for explainability, as doctors and health researchers would need to understand the models' behaviour and predictions, in order to validate the findings and understand its results.

[1]Research method involving repeated observation or data collection from the same subjects over time.

The research presented here extends the study by Dallora et al. [10] by combining black box models with XAI Techinques, therefore removing the restriction for explainable ML models to be used. The models' predictive performance will potentially be enhanced without losing explainability. However, it is important to notice that the post hoc explainability frameworks are based on approximations and statistical methods [12] and might not be as accurate as the natively explainable models.

Thus, the project's justification is based on applying non-explainable models to Dallora's work, which may have better a performance than the chosen Decision Tree, and making them explainable through the use of XAI techniques. With a better performance, the models will yield more reliable insights to the possible modifiable factors related to dementia development.

1.2 Methodology

This work utilizes non-explainable machine learning models, combined with Explainable AI techniques in the context of predicting dementia prognosis over a 10-year window. Initially, a study will be conducted on the SNAC dataset [11] and the nature of the information it contains, using the work by Dallora et al. [10] as the main reference. The SNAC sample used in this work is the same as the one used in Dallora's work. Also, as part of the preliminary studies before implementation, research is conducted on techniques for dealing with imbalanced datasets, missing data, feature selection, and machine learning models appropriate to the characteristics of the data. The performance metrics of the predictive models to be optimized are also defined during the initial phase of the project. Finally, state-of-the-art Explainable AI techniques are analysed, and those most suited to the present problem will be selected.

The Python programming language is used for all proposed experimental tasks, given its extensive use in the Machine Learning field and the availability of pre-implemented Explainable Artificial Intelligence libraries. Once the aforementioned studies are completed, the experimental phase of the work begins. This phase is

divided into four stages:

- Data cleaning and exploratory analysis;
- Data transformations;
- Training and testing of non-explainable predictive models;
- Application of Explainable AI techniques to non-explainable models;

1.3 Goals

This study is focused on using black-box Machine Learning models combined with XAI to predict dementia within a 10-year timeframe, based on data from individuals without a baseline diagnosis of the disorder.

1.4 Structure

This document is structured as follows: we begin by presenting the SNAC dataset and describing its key characteristics. Following this, the theoretical framework chapter outlines the core concepts necessary to understand and approach the problem, with a focus on machine learning model training, evaluation, and explainable AI methodologies. The subsequent chapter details the code implementation and the challenges encountered during development. The Results chapter then presents and discusses the outcomes of the model and XAI analysis. Finally, the Conclusion chapter summarises the work and offers further insights into the results.

Chapter 2

The Dataset

The data used in the present study is a sample of the Swedish National Study on Aging and Care (SNAC). SNAC is a Swedish initiative, supported by research councils and universities, initiated in response to the increasing proportion of older adults in the population [13], from 17.6% in 1993 to 20.6% in 2023 [14]. Figure 2.1 shows the Swedish population age distribution in both moments.

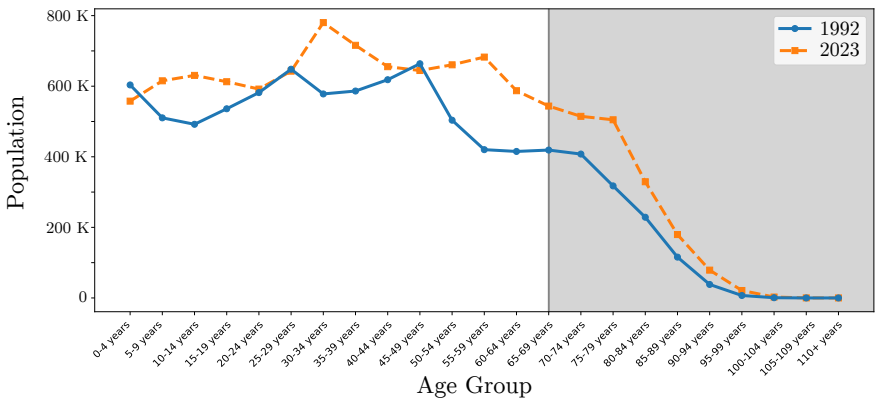


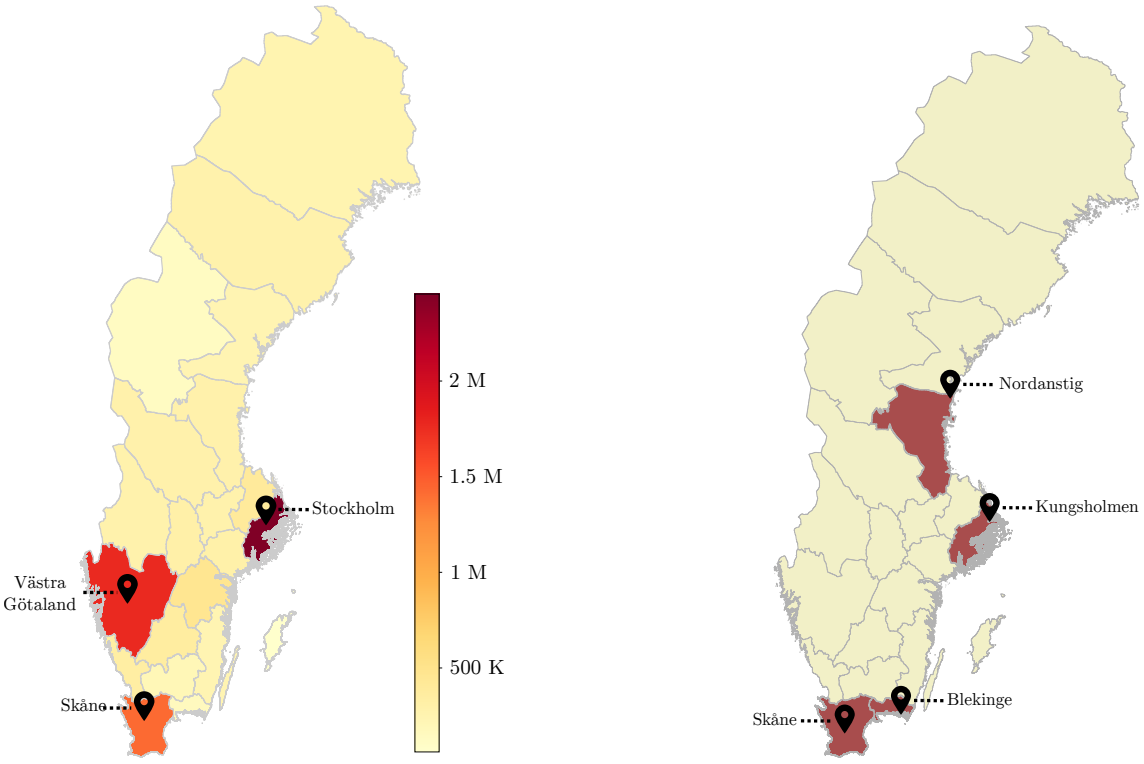
Figure 2.1: Swedish population age distribution: 1992 and 2023 comparison [14]. It is noticeable the stronger presence of 65+ population in 2023, represented by the grey area in the Figure.

The SNAC project employs randomised sampling methods within selected regions of Sweden to ensure that the study population is representative of the broader elderly demographic. SNAC comprises data collected from four distinct regions:

- Area **Skåne**, representing the southern part of the country, contemplating the Eslöv, Hässleholm, Malmö, Osby, and Ystad municipalities;

- Area **Blekinge**, representing the south-east part of the country, contemplating the Karlskrona municipality;
- Area **Kungsholmen**, a district in the city of Stockholm, representing the middle part of the country;
- Area **Nordanstig**, a minicipality of Gävelborg county council, representing the north of the country.

This set of regions was chosen with the aim of achieving a representative sample of the entire country. It is noticeable that the SNAC areas are concentrated in the middle and southern parts of the country. The reason for this choice lies in the fact that, according to the Statistics Sweden Portal (2020) [14], more than 88% of the population resides in urban areas, with these areas being predominantly concentrated in the southernmost part of the country [14].



(a) Swedish population heatmap [15].

(b) SNAC data collection areas [11].

Figure 2.2: Swedish maps, containing a population heatmap and the SNAC data collection ares.

Figure 2.2a illustrates the population heatmap of Sweden by counties in 2023 [15], emphasising the three most populous counties: Stockholm, Västrä Götland, and Skåne, with populations of $2,454,821$; $1,767,016$, and $1,421,781$ inhabitants, respectively. Complementing this, Figure 2.2b highlights the SNAC areas. From the three most populated counties, two are contemplated in the SNAC dataset (areas Kungsholmen ans Skåne), reinforcing the representativeness of the collected data.

In this context, longitudinal studies are key, as they provide a means of tracking individuals over time and understanding how factors such as aging, lifestyle, health conditions, and social changes influence their well-being, needs, and outcomes [16].

Threrefore, SNAC plays a vital role in supporting research on ageing and elderly care by creating long-term consistent databases that serve as a strong foundation for various studies. Its goal is to develop reliable and comparable data that researchers can use to better understand ageing. The information collected spans multiple disciplines that influence the ageing process, including health and illness, social relationships and support, lifestyle choices, economic circumstances, and personal resources. [11]

2.1 Dataset Characteristics

The subset used in this study was the SNAC-Blekinge, also referred to as SNAC-B, which contains 1817 entries and 1643 variables collected from 2001 to 2003 (*SNAC-B - Cohort 1 - Baseline*) characterizing individuals based in an urban south-east part of Sweden. The output variable representing the presence or not of dementia is part of the third follow-up for SNAC-B, collected from 2010 to 2013 (*SNAC-B - Cohort 1 - Follow-up 3*).

In addition, keeping in mind the goal of predicting dementia over a 10-year span, an exclusion criteria [10] was employed to eliminate non-informative entries. This criteria consists on discarding:

1. Subjects who already had dementia in the baseline;

2. Subjects with missing target variable (dementia diagnosis);

3. Subjects with more than 10% of feature values missing;

4. Subjects that passed away before the ten-year mark;

5. Subjects diagnosed with dementia before the ten-year mark;

The purpose of this filtering is to create a dataset that accurately reflects the progression or absence of dementia in a controlled manner. In Criterion 5, matching entries from the SNAC database are excluded because if dementia was diagnosed in less than ten years, the individual may have already experienced significant disease progression at baseline, making the data less relevant to the current study. After applying this criterion, the remaining number of subjects is 726.

2.1.1 Target Variable

Analysing the dataset after the exclusion criteria is applied, the distribution between male and female subjects is 43.10% and 56.90%, respectively. The proportion of subjects diagnosed with dementia at the 10 years mark (`Doc_2010_2013` = 1) is 12.53%. From the group that had a positive diagnosis at the 10-year mark, 31.87% and 68.13% represent the male and female distributions, respectively.

This may suggest that, among those diagnosed with dementia, a larger proportion are female. This finding aligns with previous studies [17, 18] that often indicate a higher dementia prevalence among women. Furthermore, the distribution between genders for positive cases is approximately similar for individuals younger than 70 years old, consistent with the findings of [18]. Table 2.1 shows the distribution of positive and negative cases by age and gender.

Diagnosis	Gender	60	66	72	78	81	84	87	90	96	Total
No Dementia	Male	81	72	47	36	27	19	2	0	0	284
	Female	81	92	67	38	35	27	7	3	1	351
Dementia	Male	1	3	3	5	8	7	2	0	0	29
	Female	1	3	7	12	11	15	12	1	0	62

Table 2.1: Distribution of dementia diagnosis in the ten-year mark by age and gender.

2.1.2 Feature Variables

To address this in the SNAC dataset, 74 feature columns were selected from the original 1,643. These features were selected by specialists in the field of dementia for the study conducted by Dallora et al. [10], and were adopted in the present study due to its compatibility with the proposed methodology and goals.

The features can be separated in 8 groups:

1. **Demographic (2 features)**: Age and gender information;
2. **Social (7 features)**: Education, religion, voluntary work, and more specific information such as social network, support network, and loneliness. The latter three features were derived from a series of standardised questions [10], producing scores ranging from 0 to 4, where 0 indicates a typical individual and 4 indicates a subject with a deficient score;
3. **Lifestyle (9 features)**: Exercise, alcohol consumption, smoking habits, engagement in leisure activities and working state at 65 years of age;
4. **Medical History (23 features)**: This category encompasses a diverse range of medical-related features, including the number of medications, cardiac issues, diabetes, cancer, Parkinson, psychiatric disorders, physical traumas, and sleep disorders;
5. **Blood Test (2 features)**: Hemoglobin Analysis and C-Reactive Protein Analysis;

6. **Physical Examination (13 features)**: Body Mass Index (BMI), recent pain, heart rate, balance and strength tests and teeth analysis;

7. **Psychological (6 features)**: Memory Loss, Memory Decline, Abstract Thinking, Personality Change, Sense of Identity;

8. **Health Instruments (12 features)**: Sense of Coherence [19], Digit Span Test [20], Backwards Digit Span Test [20], Livingston Index [21], EQ5D Test [22], Activities of Daily Living [23], Instrumental Activities of Daily Living [24], Mini-Mental State Examination [25], Clock Drawing Test [26], Mental Composite Score of the SF-12 Health Survey [27], Physical Composite Score of the SF-12 Health Survey [27], Comprehensive psychopathological Rating Scale [28];

Another important aspect to analyse is the presence of missing data in the feature columns. In this study, no feature has more than 8.13% of missing data, as shown in Table A.1, in Appendix A.

Analyzing the column types presented in Table A.1, it is evident that only `INTEGER` and `FLOAT` columns are present, categorized as follows:

- `INTEGER`
  - **Discrete columns**, such as `S_Age`, where the subject's age is recorded as a "rounded" value;
  - **Categorical columns**, such as `S_Gender`, where 1 represents "Male" and 2 represents "Female". In this case, the values do not imply magnitude;
  - **Ordinal columns**, such as `S_Social_Loneliness`, where the values are discrete but carry a sense of magnitude. In this example, 0, 1, and 2 represent a range from "Very High Level of Loneliness" to "Normal Level of Loneliness" [10];
- `FLOAT`
  - **Continuous columns**, such as `S_BMI`, which represents the continuous values of the subjects' Body Mass Index (BMI);

It is important to mention that this reduction to 75 feature variables is only one of the two feature selection steps implemented in this project's Machine Learning pipeline. The second step is explained in detail in Section 3.3, and consists of selecting features separately to each ML model.

We can now proceed to the theoretical framework necessary to understand the next steps of the Machine Learning pipeline.

Chapter 3

Theoretical Framework

In Machine Learning, various techniques can be combined to achieve specific goals [3]. This chapter outlines the main reasons for each framework choice and explains their necessity. Figure 3.1 illustrates the building blocks of the machine learning pipeline used in this study.

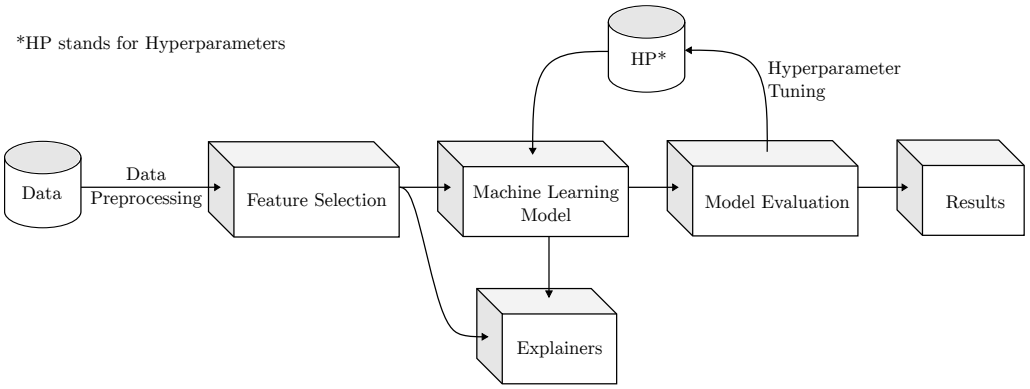


Figure 3.1: Machine Learning with Explainable AI flow adopted in the present study.

In Section 3.2, the "**Data Preprocessing**" part will be covered. This is a crucial part of the majority of Data-driven tasks, and guarantees that the Machine Learning model will have a satisfactory input data, in order to achieve a desirable performance [3].

The "**Feature Selection**" block is responsible for selecting a reduced number out of the 75 available features. The method used in this study, as well as in the baseline reference [10], was the Recursive Feature Elimination (RFE). This method is explained in detail in Section 3.3.

The "**Machine Learning Model**" block is the statistical predictor. Three models will be compared and explained in Section 3.1. In Section 3.4, different ways of evaluating a predictor will be elaborated, covering the "**Model Evaluation**" block. We will then be able to tune the models based on a selected performance metric, as outlined in 3.5.

Finally, the "**Explainers**" block will be discussed in Section 3.6. The explainers are Explainable AI algorithms that are useful after the best model is trained, in order to provide human-friendly interpretations to the predictions.

The methods selected in this study were driven mostly by the available data characteristics, as detailed in Section 2.1, as well as the objective outlined in Section 1.1. The main characteristics that ruled the choices made were:

- The presence of 726 records and 74 feature variables, highlighting the possible need for dimension reduction.
- Imbalanced dataset, with the positive labeled target (with dementia in the ten-year mark) representing only 12.53% of the records.
- Medical context application, in which predicting the minority class (subjects with dementia) is more important than predicting the majority class.

In the following sections, the methods selected for each part of the solution are presented and explained in detail.

3.1 Machine Learning Models

Machine Learning (ML) models are computational systems designed to identify patterns and make decisions based on data, without being explicitly programmed for specific tasks [29]. Unlike traditional programming, where rules and logic are defined manually, ML models learn from examples and generalize their learning to new, unseen data [5, 30].

At their core, ML models are mathematical representations of real-world processes or systems. These models consist of algorithms and statistical methods that process input data (features) to predict outputs (labels or targets) or uncover hidden structures in the data.

In Figure 3.2, a diagram containing a high level abstraction of ML models can be observed. A model $\mathcal{M}$ has the goal of giving an output as an answer a_n for an entry e_n containing features f_n regarding a question $\mathcal{Q}$.

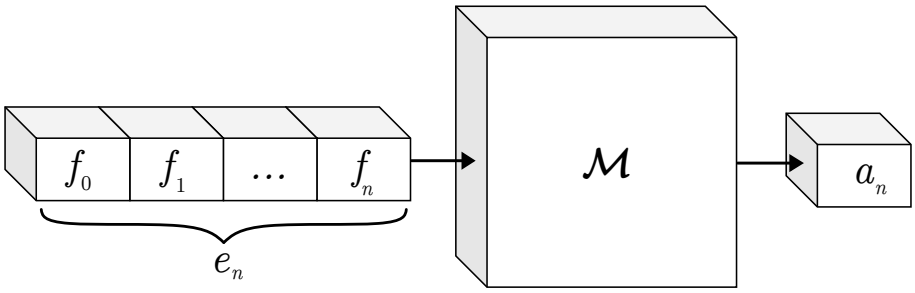


Figure 3.2: Machine Learning Model diagram.

For instance, the problem tackled in this research can have its characteristics defined as:

- Question ($\mathcal{Q}$): What is the subject's dementia diagnosis in a 10-year time?
- Entry (e_n) containing features of the individual ($f_{0...n}$): Features, as presented in Section 2.1.2, such as Age, Body Mass Index and Psychiatric Disorders, for example.
- Model ($\mathcal{M}$): Statistical methods that, given the features $f_{0...n}$, are able to answer question $\mathcal{Q}$. In our study, the models are the Random Forest Classifier, CatBoost Classifier as well as XGBoost Classifier.
- Answer (a):
  - 0: Subject will not have dementia in a ten-year time.
  - 1: Subject will have dementia in a ten-year time.

In order to be capable of giving the desired answers, the model needs to be fitted. This process consists of training the ML model on a dataset containing the correct

target variable (y) previously labeled. During the fitting process, the algorithm adjusts its internal parameters to minimize the error or loss between the predicted outputs and the actual outputs in the training data [3]. The goal is to enable the model to generalise well to unseen data, making accurate predictions based on patterns learned during training.

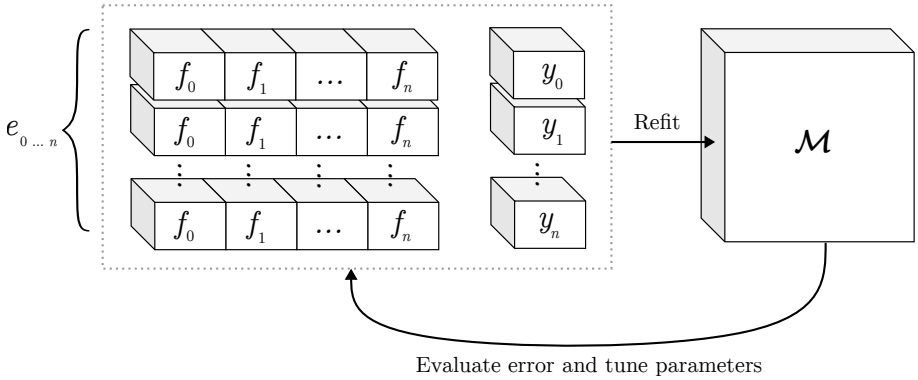


Figure 3.3: Machine Learning Model training diagram.

Figure 3.3 presents the process of fitting the model. Given entries $e_{0...n}$ with labels $y_{0...n}$, the internal ML model parameters are tuned in order to make the model optimal in a certain sense [3].

The aforementioned parameters are internal to the models, and are not modified by the user. The hyperparameters, on the other hand, are inputs of ML algorithms that can be altered by the user, such as tree depth, regularisation function and target metric. The hyperparameters do not change during fitting.

3.2 Missing Data Handling

In real-world datasets, missing data is a prevalent issue that can significantly affect the quality and reliability of subsequent analyses [31]. Missing data arises due to different reasons, including data collection errors, participant non-response, or sensor malfunctions. If not addressed properly, missing values can lead to biased estimates, poor predictive performance, or even invalid conclusions. Consequently, handling missing data is a critical step in any data preprocessing pipeline. In order to address the previously mentioned problems, different techniques might be used.

3.2.1 Data Removal

The most straightforward approach to dealing with missing data is to remove the affected rows or columns. In cases where only a small percentage of the data is missing, removing rows with missing values is a viable solution. If a specific feature contains a high proportion of missing values, removing the entire column may be justified, especially if the feature is not crucial for the analysis. This method, however, may discard potentially valuable information and should be considered only when the missing data is extensive. Furthermore, this approach can lead to an underestimation of uncertainty, resulting in an overstated sense of precision[32].

In the present study, given that there is only a limited amount of records and 8.13% is the largest percentage of missing data over all columns, it stands clear that this is not a concern in this project.

3.2.2 Imputation of Missing Values

Imputation is one of the most widely used strategies for handling missing data in real-world datasets. It involves replacing missing values with estimates that are considered plausible based on the observed data. Imputation techniques vary in complexity, ranging from simple statistical methods to more advanced model-based approaches, each with its own set of assumptions and implications [33]. If used correctly, imputation techniques can preserve the dataset's size and structure, making them preferable over row or column deletion for a variety of problems.

3.2.2.1 Mean, Median, or Mode Imputation

For continuous data, the missing values can be replaced with the mean or median of the observed values in that feature. Those methods are computationally efficient and can work well in different scenarios.

Imputing the mean of a column is suitable for normally distributed datasets, particularly in cases where the dataset contains a large number of records due to its computational efficiency [34]. Despite its simplicity, mean imputation has several limitations. It tends to distort the distribution of the feature by reducing its vari-

ance, leading to underestimation of the variability in the data [34]. Moreover, this technique is sensitive to outliers and skewed distributions [35].

Replacing the values with the column's median brings robustness to outliers and skewed distributions, when compared with the mean [35]. Like mean imputation, median imputation reduces the variability of the data by replacing missing values with a constant value, leading to an underestimation of the feature's variance. In some specific cases, the median can be useful when dealing with discrete ordinal columns. More details regarding this specific use case will be presented later in this section.

For categorical [32] and often ordinal data, the mode can be used. Although selecting the mode of a feature as its imputation strategy is widely used, some drawbacks need to be taken into account. As the previously shown techniques, this method can distort the variance and relationships between features, especially if the missing values represent a large proportion of the dataset [32].

3.2.2.2 K-Nearest Neighbors (KNN) Imputer

Differently from the previously mentioned imputation strategies, the K-Nearest Neighbors Imputer does not generate the filling values based on the whole dataset, but rather in a "curated" subset of it. This subset is generated based on the proximity of each set of features. After selecting the subset, the value to be imputed will be the result of a chosen calculation strategy applied to the neighbors values. The most common strategies are the mean, median and mode. A detailed implementation of the algorithm can be seen in Algorithm 1.

Algorithm 1 KNN Missing Data Imputation

Input: Dataset $\mathcal{D}$ with missing values, number of neighbors k, distance metric d and imputation strategy $\mathcal{S}$

Output: Dataset $\mathcal{D}_{\text{imputed}}$ with missing values imputed

Identify the set of features with missing values $\mathcal{M}$
for each feature $f_m \in \mathcal{M}$ **do**
    for each missing entry x_{ij} in feature f_m **do**
        Compute distances between x_{ij} and all non-missing entries in f_m using d
        Identify the k nearest neighbors based on computed distances
        Retrieve the corresponding values of the k neighbors for f_m
        Impute x_{ij} with $\mathcal{S}$ applied to the k neighbor values
    end for
end for
$\mathcal{D}_{\text{imputed}} \leftarrow \mathcal{D}$ with all missing values replaced
return $\mathcal{D}_{\text{imputed}}$

Algorithm 1 is a direct implementation of the Scikit-Learn Python library [36], which itself is adapted from the approach proposed in a study by Troyanskaya et al. [37]. Additionally, in order to make it possible to calculate the median of the neighbors, tailoring the implementation to the specific requirements of this study, modifications were made directly within the Scikit-Learn source code, as outlined in Chapter 4, taking Torgo's work [38] as the main reference.

To provide a clearer understanding of the KNN Imputer process, we will use the example data presented in Table 3.1. The missing value in `Feature B` will be calculated and imputed step by step.

Specifically, we will work through the example using $K = 3$ nearest neighbors, employing the **Euclidean distance** to identify the most suitable neighbors and the **median** as the imputation strategy.

	Feature A	Feature B	Feature C
x_1	5	10	1
x_{target}	6	NULL	3
x_2	NULL	14	2
x_3	8	12	NULL
x_4	18	21	5

Table 3.1: Example data containing the candidate rows x_1, x_2, x_3, x_4 and the target row x_{target}.

First, we need to calculate the distances between the target row x_{target} and the candidate rows x_1, x_2, x_3, x_4. When dealing with Euclidean distances between a pair where other features containing missing values might be present on both target and candidate rows, it is important to have a clear rule for this calculation.

To compute the distance between two vectors with missing values, a variation of the Euclidean distance [39] is used. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ be two vectors, where some coordinates may have missing values. This distance calculation adjusts for these missing values by ignoring them in the calculation and scaling up the weight of the remaining coordinates. If no missing values are present, the distance will equal the Euclidean distance.

The weight factor, which compensates for the reduced number of available features, is defined as:

$$\text{weight} = \frac{n}{p} \tag{3.1}$$

where n is the total number of features, and p is the number of features that are non-missing in both vectors X and Y.

This version of the Euclidean distance between vectors X and Y, denoted as $d(X, Y)$, is given by:

$$d(X, Y) = \sqrt{\frac{n}{p} \sum_{i \in \text{valid}} (x_i - y_i)^2} \tag{3.2}$$

In this formulation:

- "Valid" is a set containing the feature column indexes where both target and candidate rows are not missing values. In other words, the summation $\sum_{i \in \text{valid}}$ runs over all indices i where both x_i and y_i are non-missing.
- The term $(x_i - y_i)^2$ represents the squared difference for each valid coordinate.
- The factor $\frac{n}{p}$ scales the distance to reflect the missing values, ensuring that it represents the same magnitude as if all features were present.

This adjusted distance measure allows for a comparison between vectors with partially missing data. Calculating for the example in Table 3.1:

$$x_{\text{target}} = \{6, \texttt{NULL}, 3\},\ x_1 = \{5, 10, 1\},\ x_2 = \{\texttt{NULL}, 14, 2\},\ x_3 = \{8, 12, \texttt{NULL}\},\ x_4 = \{18, 21, 5\}$$

$$\begin{aligned} d(x_{\text{target}}, x_1) &= 2.74 \\ d(x_{\text{target}}, x_2) &= 1.73 \\ d(x_{\text{target}}, x_3) &= 3.46 \\ d(x_{\text{target}}, x_4) &= 14.90 \end{aligned} \tag{3.3}$$

We can thus state that the selected neighbors are rows $x_1,\ x_2,\ x_3$, as they have the lowest distances. Their values will be used to calculate the imputed variable, as follows.

The second step involves calculating the imputed value using the chosen metric. This is done by applying the metric to the feature of interest, that is, the missing feature in x_{target}. In this example, we will calculate the median with each neighbor's value. In some cases, however, each neighbor can be weighted by the inverse of its distance to the target point, giving closer neighbors a greater influence than those further away.

$$\text{median}(10, 14, 12) = 12$$

With that, we conclude that 12 will be the imputed value. Applying the algorithm for the other missing values,

	Feature A	Feature B	Feature C
x_1	5	10	1
x_{target}	6	12	3
x_2	6	14	2
x_3	8	12	2
x_4	18	21	5

Table 3.2: Example data containing the values imputed by the KNN Imputer algorithm for $K = 3$, using the Euclidean distance and median as the imputation strategy.

In case of categorical and ordinal feature columns, the median and mode were chosen as imputation methods, respectively, following Torgo's implementation [38]. Further details regarding the KNN Imputer algorithm implementation are be shown in Chapter 4.

3.3 Feature Selection

Feature Selection is a data preprocessing strategy that involves choosing the most important features for a given ML problem. When selecting the most important features, the number of variables is reduced excluding redundant and uninformative variables, thus yielding simpler and more interpretable models, faster to train and often with better performance [40]. This strategy has demonstrated effectiveness across various data mining and ML applications [41] and was employed to address the challenge discussed in this document.

It is important to mention that the 75 available features in the dataset were selected by experts in the field, which itself is another useful feature selection method.

3.3.1 Recursive Feature Elimination (RFE)

Recursive Feature Elimination is a wrapper technique [42] used to systematically eliminate features from model training [43], working as a Feature Selection method.

RFE recursively trains a given model with a subset from the original features and evaluates it in a test set.

The algorithm selects a subset of k features, number chosen by the user, out of the n features in $\mathcal{F}$. It does that by ranking the features by their importance $I(f)$, then excluding the one(s)[1] with the lowest importance score. The algorithm repeats the process until $\mathcal{F}_{\text{selected}}$ contains k features. Algorithm 2 presents the detailed pseudocode of the RFE process.

Algorithm 2 Recursive Feature Elimination (RFE)

Input: Dataset $\mathcal{D}$ with n features, base model M, number of features to select k

Output: Subset of k features $\mathcal{F}_{\text{selected}}$

Initialize feature set $\mathcal{F} \leftarrow \{f_1, f_2, \ldots, f_n\}$
while $length(F) > k$ **do**
    Train model M on features $\mathcal{F}$
    Compute feature importance scores $I(f)$ for all $f \in \mathcal{F}$
    Identify least important feature $f_{\min} \leftarrow \arg\min_{f \in \mathcal{F}} I(f)$
    Remove $f_{\min}$ from $\mathcal{F}$
end while
$\mathcal{F}_{\text{selected}} \leftarrow \mathcal{F}$
return $\mathcal{F}_{\text{selected}}$

Notably, in this research, feature selection was performed individually for each different model $\mathcal{M}$.

3.4 Model Performance Evaluation

A crucial part of Machine Learning problems is the evaluation of a trained model's performance. The choice of performance metrics should align with the specific characteristics of the available data and the project's objectives. Selecting the most

[1] In some cases, multiple features might be eliminated at once to reduce computational costs [43]. In this document's implementation, only one feature was excluded in each round.

appropriate metrics helps provide a more meaningful and reliable assessment of the model's behavior.

The characteristics of the dataset used in this study were discussed in Section 2.1. Drawing on these characteristics, as well as insights from the relevant literature [44, 3, 29], the following performance metrics were selected for analysis:

- Sensitivity
- Specificity
- Precision
- False Positive Rate (FPR)
- Area Under the Receiver Operating Characteristic Curve (ROC AUC)
- Area Under the Precision-Recall Curve (AUC-PR)

3.4.1 Confusion Matrix

The confusion matrix is a standardized framework for analyzing the performance of classifiers. It provides a comprehensive view of a model's predictions by categorizing outcomes into true positives, true negatives, false positives, and false negatives.

This matrix is essential not only for understanding the model's behavior but also for interpreting how each performance metric is derived and what aspects of the model's performance these metrics reflect.

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

Table 3.3: Confusion matrix for a binary classification problem.

3.4.2 Sensitivity

The Sensitivity, also known as Recall or True Positive Rate (TPR), is a performance metric that measures the proportion of correct positive predictions over all the positive labeled data entries.

This metric is considered one of the most important in medical studies [45] because a higher recall indicates better capability in predicting the positive class. In the present study, this metric translates to the capability of predicting dementia in patients.

$$Sensitivity \text{ (Recall or TPR)} = \frac{TP}{FN + TP} \tag{3.4}$$

3.4.3 Specificity

The Specificity, also known as True Negative Rate (TNR), is the rate of the correctly predicted negatives over all actual negative entries (true negatives and false positives).

The intuition behind specificity is the model's capacity to correctly predict the negative class.

$$Specificity \text{ (TNR)} = \frac{TN}{TN + FP} \tag{3.5}$$

3.4.4 Precision

Precision, also known as Positive Predictive Value (PPV), is a performance metric that measures the proportion of correctly predicted positive cases among all instances classified as positive.

This metric is particularly important in applications where false positives carry significant consequences [45]. A higher precision indicates a lower rate of false positive predictions.

$$Precision \text{ (PPV)} = \frac{TP}{TP + FP} \tag{3.6}$$

3.4.5 False Positive Rate (FPR)

The False Positive Rate is the proportion of actual negative cases that are incorrectly classified as positive. It tells us how often a model mistakenly identifies a negative instance as positive.

$$\text{FPR} = \frac{FP}{TN + FP} \tag{3.7}$$

3.4.6 Receiver Operating Characteristic Area Under the Curve (ROC AUC)

The ROC curve is a graphical representation used to evaluate the performance of a binary classifier. It plots the True Positive Rate, against the FPR at various threshold settings. The ROC curve provides a visual representation of the trade-off between sensitivity and specificity as the decision threshold changes.

The intuition behind the ROC curve is that, for each chosen classification probability threshold, a confusion matrix is created. Each of these confusion-matrices yields a *TPR (Sensitivity)* and *False Positive Rate (FPR)* pair, which is used as a point in the ROC curve [3].

An useful and widely used performance metric in imbalanced classification [46] measure when working with the ROC curve is calculating is the Area Under the Curve (AUC). The ROC AUC presents a quantification of how well a given model performs on distinguishing the positive and negative classes, being useful for both balanced and imbalanced ML problems.

To further exemplify the intuition proposed in the beginning of this section, Figure 3.4 shows the ROC for a logistic regression classifier trained over synthetic data containing 1000 records. The ROC AUC is obtained by calculating the area under the blue curve. Points **A**, **B** and **C** are plotted by selecting different classification probability thresholds and analyzing the respective confusion matrices, calculating the desired (*FPR*,*TPR*) pair. The thresholds for points **A**, **B** and **C** are 0.2, 0.5 and 0.8, respectively, and each confusion-matrix can be found in Tables 3.4, 3.5 and 3.6.

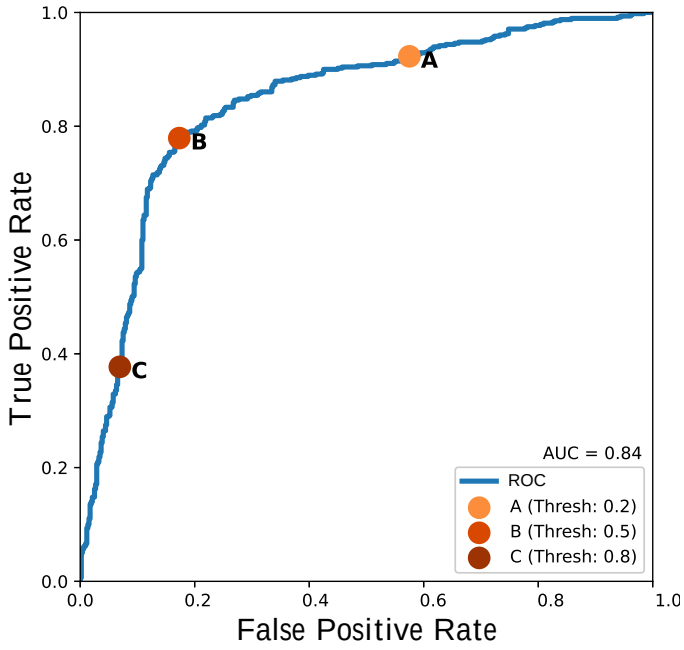


Figure 3.4: ROC with three points in different thresholds.

	Predicted Negative	**Predicted Positive**
Actual Negative	TN: 221	FP: 299
Actual Positive	FN: 37	TP: 443

Table 3.4: Confusion Matrix at Threshold **A** (0.2).

	Predicted Negative	**Predicted Positive**
Actual Negative	TN: 430	FP: 90
Actual Positive	FN: 106	TP: 374

Table 3.5: Confusion Matrix at Threshold **B** (0.5).

	Predicted Negative	**Predicted Positive**
Actual Negative	TN: 484	FP: 36
Actual Positive	FN: 299	TP: 181

Table 3.6: Confusion Matrix at Threshold **C** (0.8).

3.4.7 Area under the Precision-Recall Curve (AUC-PR)

The Precision-Recall (PR) curve is another graphical representation commonly used to evaluate the performance of a binary classifier , especially in the context of imbalanced datasets [47]. It plots Precision (PPV) against Recall (Sensitivity) at various threshold settings. The PR curve provides insight into the trade-off between precision and recall as the decision threshold changes.

The intuition behind the PR curve is similar to that of the ROC curve: for each chosen classification probability threshold, a confusion matrix is generated. Each confusion matrix yields a pair of values of Precision and Recall, which are plotted as points on the PR curve.

Unlike the ROC curve, which illustrates the model's ability to distinguish between classes by plotting TPR and FPR, the PR curve emphasizes the model's performance on the positive class by focusing on how many true positives it identifies (recall) relative to false positives (precision). This makes the PR curve particularly useful for imbalanced classification problems where the minority class is of primary interest, as it directly reflects the balance between capturing positives accurately and minimizing false positives.

A widely used performance metric derived from the PR curve is the Area Under the Precision-Recall Curve (AUC-PR), which quantifies the model's performance in identifying positive instances. Unlike ROC AUC, AUC-PR can provide a more informative [47] evaluation of classifier performance in cases where the positive class is much rarer than the negative class, as it directly considers the precision-recall trade-off.

To illustrate this concept, Figure 3.5 shows the Precision-Recall curve for the same classifier trained in Section 3.4.6. Similarly to the ROC AUC, the AUC-PR is obtained by calculating the area under the blue curve. Points **A**, **B**, and **C** were plotted by selecting the same thresholds from last section's example and analysing their (Precision, Recall) pairs extracted from its confusion-matrices, shown in Tables 3.4, 3.5 and 3.6.

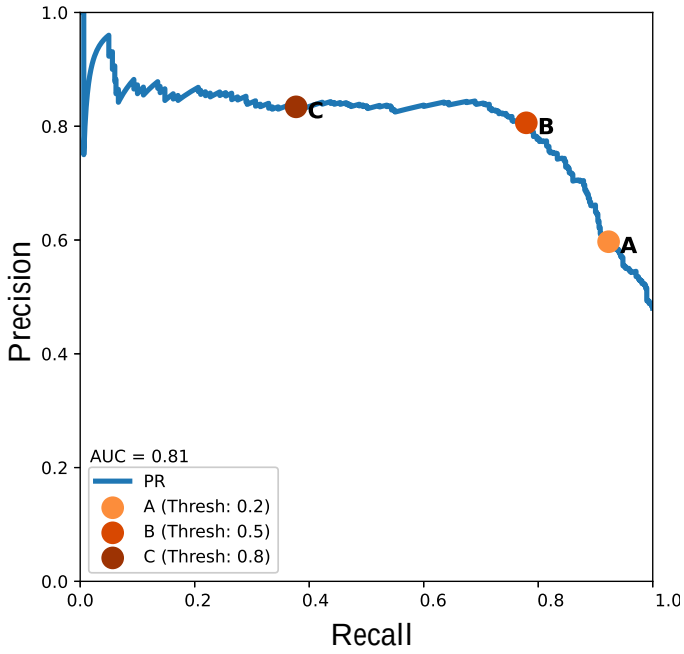


Figure 3.5: PR with three points in different thresholds.

3.5 Model Selection

As explained in Section 3.1, different types of Machine Learning models can be used to address the same problem. Naturally, each model comes with its own set of particularities and tuning options, which result in varying performance for different "model-hyperparameter" pairs, as well as with different input data provided to the model.

In this section, the techniques used to select the best "model-hyperparameter" pair are explained, ensuring robustness to the particularities of the input data. Additionally, key concepts related to this important step in the Machine Learning pipeline are discussed.

3.5.1 Grid Search

As mentioned in the beginning of this Chapter, each Machine Learning Model has its own set of user modifiable parameters, called hyperparameters. For instance, CatBoost Classifier has more than 50 possible hyperparameters to be tuned [48], each one of them being either a binary, integer or floating point variable.

To address the large number of possible hyperparameter combinations, Grid Search provides an automated process for testing all possible combinations. In summary, each possible hyperparameter chosen by the user is instantiated in a "hyperparameter grid", and Grid Search is responsible for training and evaluating the model for each one of those different settings, as presented in Algorithm 3. Examples of "scores" would be the ones presented in Section 3.4.

It is important to note that the hyperparameter tuning step is often time- and resource-intensive, and Grid Search may not always be the most suitable approach. Alternative algorithms, such as Randomized Search and Bayesian Optimization, can be more effective for large datasets or when the hyperparameter grid has many dimensions. These methods do not train the model for all possible combinations but instead focus on selecting more promising hyperparameter configurations based on their respective algorithms.

Algorithm 3 Grid Search for Hyperparameter Tuning

Input: Dataset $\mathcal{D}$, model $\mathcal{M}$, hyperparameter grid $\mathcal{G}$, evaluation metric m

Output: Best model $\mathcal{M}_{\text{best}}$ with optimal hyperparameters

Initialize best score best_score $\leftarrow -\infty$
Initialize best hyperparameters $\theta_{\text{best}} \leftarrow \emptyset$
for each combination of hyperparameters $\theta \in \mathcal{G}$ **do**
    Train model $\mathcal{M}$ with hyperparameters θ on dataset $\mathcal{D}$
    Evaluate model $\mathcal{M}$ using metric m
    if score $>$ best_score **then**
        Update best_score $\leftarrow$ score
        Update $\theta_{\text{best}} \leftarrow \theta$
    end if
end for
$\mathcal{M}_{\text{best}} \leftarrow \mathcal{M}$ with hyperparameters θ_{best}
return $\mathcal{M}_{\text{best}}$

3.5.2 Nested Cross-Validation

Data leakage occurs when information from outside the training dataset is used to create the model, leading to over-optimistic performance estimates and causing the model to perform poorly on unseen data. This issue can happen during both the training and hyperparameter tuning processes, where the model unintentionally gains access to information from the test data or future data points that would not be available in a real-world scenario. Data leakage can result in misleading evaluation metrics and poor model generalisation [3].

Overfitting occurs when a model is over-adjusted to a particular dataset, resulting in poor generalisation to new, unseen data. This often leads to models that perform well on the training set but fail to maintain accuracy on the test set.

Nested Cross-Validation is a model evaluation technique that addresses overfitting and data leakage during model selection [49, 10], providing a more reliable method of selecting the model's hyperparameters and estimating its generalisation performance.

This method consists in two cross-validation setups, called inner and outer cross-validation, as shown in Figure 3.6. The main difference between nested and regular cross-validation is that, when using the nested framework, the test data is unseen in hyperparameter tuning section. The detailed the algorithm's pseudocode can be found in Algorithm 5, and each fold's explanation are shown in the following items.

- **Outer training sets**: A portion of the overall data that is passed to the inner cross-validation setup. Also, this is the data chunk that is used to train the model with the best hyperparameters found in the inner cross-validation via the Grid Search algorithm. Those sets are represented in Figure 3.6 by the portions of o_1 to o_3 that are not coloured.
- **Outer test sets**: A portion of the overall data that remains completely unseen during the entire model selection process. It is used solely for evaluating the performance of the model, which is trained on the outer training set and tuned

using the best hyperparameters selected from the inner cross-validation. Those sets are represented in Figure 3.6 by the portions of o_1 to o_3 that are coloured red.

- **Inner training sets**: Subsets of the outer training set, created through a regular cross-validation process within the inner loop. These inner training sets are used to train the models during hyperparameter tuning, typically within a Grid Search or other hyperparameter optimisation methods. In this work, it is also used for selecting the best features through the Recursive Feature Elimination algorithm. Those sets are represented in Figure 3.6 by the portions of $o_1\ i_1$ to $o_3\ i_3$ that are not coloured blue. It is good to mention that, in the figure, only the inner folds related to the third outer fold (o_3) are presented, to avoid visually polluting the diagram.

- **Inner test sets**: Subsets of the outer training set, created through a regular cross-validation process within the inner loop. The inner test sets are used to evaluate the models inside a Grid Search or other hyperparameter optimisation methods. The aggregation of the results evaluated in this set are called the inner results, which are used for selecting the best hyperparameters. Those sets are represented in Figure 3.6 by the portions of $o_1\ i_1$ to $o_3\ i_3$ that are coloured blue.

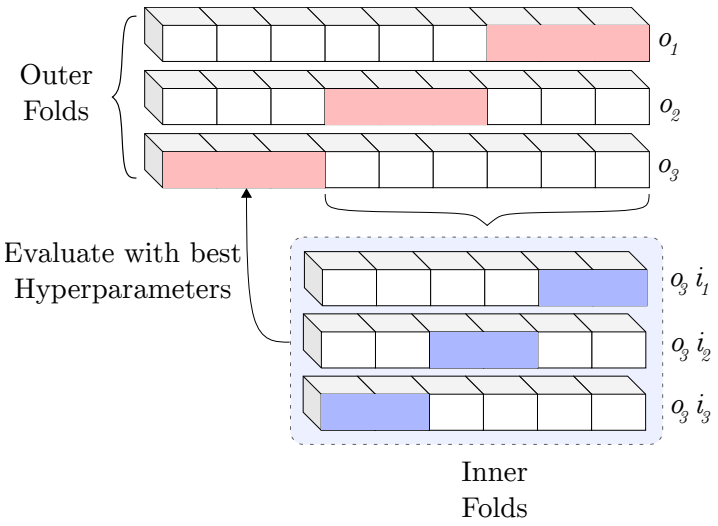


Figure 3.6: Nested Cross-Validation representation for a 3 outer 3 inner cross validation split.

3.6 Explainable AI Techniques

In machine learning, the complexity of some models often lead to a trade-off between performance and explainability. While simpler models, such as decision trees and K-Nearest Neighbors, offer transparent decision-making processes that are easy to understand for both technical and non-technical audiences, more complex models can provide superior performance but lack interpretability. These models, including Deep Neural Networks and ensemble methods, deliver high accuracy but make it difficult to understand the reasoning behind their predictions [3].

To address the lack of explainability, Explainable Artificial Intelligence has emerged as a crucial field in Machine Learning, aiming to make complex models more understandable. By providing methods that shed light on the inner workings of "black box" models, Explainable Artificial Intelligence helps build trust and accountability, ensures fairness [7, 50], and allows for model debugging . These capabilities are critical for improving decision-making and meeting regulatory requirements, especially in high-stakes scenarios.

As the need for explainability started to feature as a hot topic, different methods to help users interpreting models were created, however, their use cases and relationship were not clear [12]. In order to address this issue, the SHAP framework was proposed, unifying six existing Explainable Artificial Intelligence methods and demonstrating that all six aforementioned methods were based on the same equation [12]. Because of that and SHAP's higher consistency to human model interpretations when compared to other methods, it was the chosen XAI framework to be used in this work.

3.6.1 Additive Feature Attribution Methods

The unification of those six methods enable a class of "Additive Feature Attribution Methods" to be created. This characteristic shows that LIME, DeepLIFT, Layer-Wise Relevance Propagation, Shapley regression values, Shapley sampling values and Quantitative Input Influence all follow the same equation [12], shown in Equation 3.8.

The goal is to find a local explanation model to explain the original prediction $f(x)$, where x is a single input and f the original prediction model. The methods that follow Equation 3.8 attribute an effect ϕ_i to each feature and sums the effects of all feature attributions that approximates the output $f(x)$ of the original model.

Explanation models often use simplified inputs x' that map to the original input through a mapping function $x = h(x')$. In this work, the focus is on local methods that explain a prediction based on a single input x [51, 12], through a simplified function g, equivalent to the original model $g(z') \approx f(h(z'))$. The input z' is a subset of the original simplified input x', containing part of its non-null features [52], yielding $z' \approx x'$.

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{3.8}$$

An interesting and useful characteristic of Additive Feature Attribution Methods is the fact that there is an unique solution with three properties: Local Accuracy, Missingness and Consistency.

Property 1 - Local Accuracy: The Local Accuracy is a property that ensures the explainable approximation for a model f should at least match the prediction of the original model for $x' = x$ or, in other words, for the simplified input x' corresponding to the original input x.

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i \tag{3.9}$$

Property 2 - Missingness: The Missingness property states that the features "discarded" after simplifying the input must not impact the prediction. [12]

Property 3 - Consistency: Lastly, the Consistency asserts that if a model is adjusted such that the marginal contribution of a feature increases or remains unchanged, regardless of other inputs, then the value attributed to that feature should not decrease [12]. In other words, given two different models g and g', if the contribution of feature i in the model g' is greater than its contribution in model g [52],

$$g'(z') - g'(z'\backslash i) \geq g(z') - g(z'\backslash i) \tag{3.10}$$

then,

$$\phi_i(g', z') \geq \phi_i(g, z') \tag{3.11}$$

3.6.2 Shapley Value

The Shapely Value, named after Lloyd Shapley who introduced the concept in 1951, comes from the cooperative game theory and represents the surplus generated by the coalition of the selected features in a cooperation game [53].

"Cooperative games" is a branch of game theory, a field of mathematics and economics that studies strategic interactions among rational players. In cooperative games, the focus is on how groups of players, known as coalitions, can work together to achieve a common goal and how to fairly distribute the collective payoff (rewards, benefits, or resources) among the players who contributed to the success [53].

In the context of Explainable Artificial Intelligence and SHAP, cooperative games provide a useful framework for understanding how features (players) collaborate to produce the outcome (prediction) of a machine learning model. Here, the "game" involves the contribution of different subsets of features (coalitions) toward the overall prediction, and the goal is to fairly distribute the "payoff" (model output) among the features based on their individual contributions.

The amount player i should receive is given by [53, 54]

$$\phi_i = \sum_{S \subseteq F \backslash \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} \left[f(S \cup \{i\}) - f(S) \right] \tag{3.12}$$

Breaking down the equation:

- M is the number of players (features) in the game;
- $M!$ represents the number of ways to form a coalition of M players;
- S is the coalition, a subset of the complete set of features F excluding i. S represents a subset of the non-zero entries in x', as explained in Section 3.6.1 ($S = z'$);

- $|S|$ is the number of players in the coalition;
- $|S|!$ is the number of ways coalition z' can be formed;
- $(M - |S| - 1)!$ is the number of ways players can join after player i joined;
- $f(S \cup \{i\}) - f(S)$ is the model output with a coalition (set of features) including feature i subtracted by the model output of the coalition excluding feature i;

The term $\frac{|S|!(M-|S|-1)!}{M!}$ is known as "weight", and $f(S \cup \{i\}) - f(S)$ is the marginal contribution of player i to coalition S. Finally, the *weight* $\cdot$ *marginal contribution* product is calculated for all possible coalitions player i can join, yielding a "fair value" for player i in an M player game. In the XAI context, is how much "importance" the feature has to the whole prediction.

Theorem 1 in Lundberg and Lee (2017) [12] asserts that when an explanation model is defined under the framework of additive feature attribution methods (as specified in Definition 1) and is required to satisfy the properties of Local Accuracy, Missingness, and Consistency, explained in Section 3.8, there is only one model that meets these criteria.

Lundberg and Lee also pointed that Young (1985) [54] demonstrated that Shapley values are the only set of values that satisfy three axioms similar to Property 1 (Local Accuracy) and Property 3 (Consistency), presented in section 3.6.1, and therefore any methods not rooted in Shapley values are likely to compromise either Local Accuracy, Consistency, or both.

3.6.2.1 Numerical Example

Let the set of features be $F = \{A, B, C\}$, and we aim to compute the Shapley value ϕ_A for feature A. The model output values for each subset $S \subseteq F$ (referred to as coalitions in the Shapley framework) are shown in Table 3.7.

Subset S	$f(S)$
$\emptyset$	5
$\{A\}$	8
$\{B\}$	6
$\{C\}$	7
$\{A, B\}$	10
$\{A, C\}$	12
$\{B, C\}$	11
$\{A, B, C\}$	15

Table 3.7: Model output values $f(S)$ for all subsets $S \subseteq \{A, B, C\}$.

We calculate ϕ_A by iterating over all subsets (coalitions) $S \subseteq F \setminus \{A\} = \mathcal{P}(F \setminus \{A\}) = \{\emptyset, \{B\}, \{C\}, \{B, C\}\}$. Using equation 3.12 and expanding the summation,

$$\begin{aligned}
\phi_A = &\frac{0!(2!)}{3!}\left[f(\emptyset \cup \{A\}) - f(\emptyset)\right] \\
&+ \frac{1!(1!)}{3!}\left[f(\{B\} \cup \{A\}) - f(\{B\})\right] \\
&+ \frac{1!(1!)}{3!}\left[f(\{C\} \cup \{A\}) - f(\{C\})\right] \\
&+ \frac{2!(0!)}{3!}\left[f(\{B, C\} \cup \{A\}) - f(\{B, C\})\right] \\
&\approx 3.83
\end{aligned}$$

Therefore, the Shapley value for feature A is approximately $\phi_A = 3.83$. For features B and C, the same process should be executed, yielding their respective Shapley values.

3.6.3 SHapley Additive exPlanations (SHAP)

SHAP is an adaptation of Shapley values to explain individual predictions of ML models in an agnostic way. It provides the unique additive feature importance method that follows the local accuracy, missingness and consistency properties, mentioned in Section 3.6.1, and attributes to each feature a value that corresponds to the

change in the expected model prediction when conditioning on that feature. Moreover, computing exact Shapley values is computationally expensive, especially as the number of features increases [12]. To address this limitation, SHAP introduces efficient approximation methods, such as Kernel SHAP, Deep SHAP, and Tree SHAP, that make it practical to apply Shapley-based explanations to complex models [12].

In order to get to the model's output $f(z)$ from the SHAP values of each feature, we can sum the expected value $E[f(z)]$ to the sum of each SHAP value θ_i, as in Equation (3.13) [12, 55]

$$f(z) = E[f(z)] + \sum_{i=1}^{M} \theta_i \tag{3.13}$$

In Figure 3.7, the aforementioned behaviour is visually explained, showing each variable's contribution to the final result.

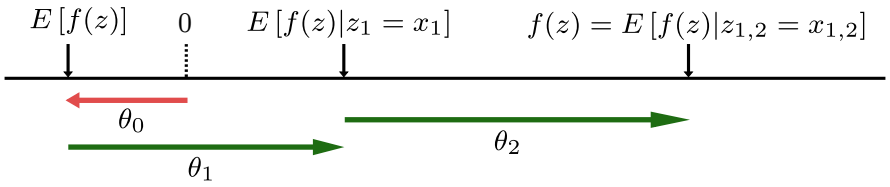


Figure 3.7: SHAP values showing how two variables ($M = 2$) contribute to the model's prediction. This figure is an adaptation from SHAP's paper [12].

To further understand SHAP, the log-odds concept must be introduced. Also known as the logit function, log-odds is a fundamental concept in statistical modeling and Machine Learning, particularly in logistic regression [56]. The log-odds of an event occurring is defined as the natural logarithm of the odds [55], where the odds are the probability of the event happening divided by the probability of it not happening. Mathematically, this is expressed as:

$$\text{log-odds} = \ln\left(\frac{P(y=1)}{1-P(y=1)}\right) \tag{3.14}$$

In classification models, predictions are often made in the log-odds space before being transformed into probabilities using the sigmoid function [57]. While odds combine multiplicatively, meaning that the overall effect is the product of individual

odds contributions, log-odds are additive. This result is expected as the logarithm's product rule states that $\log(ab) = \log(a)+\log(b)$, therefore the total combined result is the sum of individual log-odds contributions.

As shown in Equation 3.13 and the fact that the log-odds space is additive, the final log-odds prediction is obtained by summing the SHAP values of all features along with the expected value of the model. This log-odds formulation allows SHAP to provide consistent, interpretable feature attributions, which can later be converted into probabilities if needed through the inverse-logit (sigmoid) function:

$$P(y = 1) = \frac{1}{1 + e^{-\left(E[f(z)]+\sum_{i=1}^{M}\theta_i\right)}}. \tag{3.15}$$

This relationship between SHAP values and log-odds ensures that the impact of each feature remains linear in log-odds space, preserving interpretability and adherence to Shapley principles [58].

A SHAP Python package containing its value calculation and visual analysis methods is available as an open source project on GitHub[2], and was extensively used in the work presented here, detailed in Section 4.3.

3.6.4 Numerical Example - Kernel SHAP

As previously mentioned in Section 3.6.3, SHAP has different approximation methods to simplify complex computations. To illustrate that, we will present a numerical example from one of the approximations, called Kernel SHAP.

Linear Local Interpretable Model-agnostic Explanations (LIME), presented in 2016 by Ribeiro et al. [51], uses a linear explanation model to locally approximate f with a weighted linear regression.

Kernel SHAP is a combination of Linear LIME [51] and Shapley values. It estimates how each feature contributes to a specific model prediction by evaluating the model on perturbed versions of the input, where only subsets of the features are

[2] `https://github.com/shap/shap`

used and the rest are replaced by a reference value. In SHAP's documentation [58], the reference value is 0.

Similarly to LIME, to assign importance to each feature, Kernel SHAP constructs a linear model that assumes the model output can be approximated as:

$$f(x) \approx \phi_0 + \sum_{i \in S} \phi_i \tag{3.16}$$

where ϕ_0 represents the model's output when no features are known (the base value), and ϕ_i is the SHAP value representing feature i's contribution.

Each coalition S is assigned a weight based on the Shapley kernel function [12]:

$$\pi(S) = \frac{(M-1)}{\binom{M}{|S|} \cdot |S| \cdot (M - |S|)} \tag{3.17}$$

This weighting ensures that subsets of moderate size contribute more significantly to the regression, while subsets of size 0 and M are given extremely large weights to anchor the model at the base value $f(\emptyset)$ and full prediction $f(S)$, where $|S| = M$ [12]. This is important so that $\phi_0 = f(\emptyset)$ and $f(x) = \sum_{i=0}^{M} \phi_i$.

The SHAP values are then obtained by solving a weighted least squares problem, resulting in a unique additive explanation for the prediction. This formulation satisfies key properties of Shapley values, such as local accuracy, missingness, and consistency.

In the following example, the function output values $f(S)$ for all subsets $S \subseteq \{A, B, C\}$ are directly provided, rather than being computed via actual model evaluations. This simplification allows us to focus on illustrating the regression-based mechanism of Kernel SHAP without relying on a specific underlying model. However, it is important to note that, in practice, Kernel SHAP estimates these values by querying the model on perturbed versions of the input [51].

To demonstrate Kernel SHAP with a numerical example, we revisit the output values $f(S)$ for all subsets $S \subseteq \{A, B, C\}$ as presented in Table 3.7. Using these values, Kernel SHAP constructs a weighted regression model and solves for ϕ_A, ϕ_B, ϕ_C and ϕ_0. The weights for each subset from S are

Subset S	$f(S)$	$\pi(S)$
$\emptyset$	5	10000.000
$\{A\}$	8	0.333
$\{B\}$	6	0.333
$\{C\}$	7	0.333
$\{A, B\}$	10	0.333
$\{A, C\}$	12	0.333
$\{B, C\}$	11	0.333
$\{A, B, C\}$	15	10000.000

Table 3.8: Model output values $f(S)$ and Shapley kernel weights $\pi(S)$ for all subsets $S \subseteq \{A, B, C\}$.

This regression problem to be solved is equivalent to minimising the following weighted loss function

$$L(f, \phi, \pi) = \sum_{S \subseteq F} \left[f(S) - X_S \cdot \phi \right]^2 \pi(S), \tag{3.18}$$

where $f(S)$ is the model output for subset S, X_S is the binary indicator vector of active features in S and the intercept term (base value ϕ_0), ϕ is the vector of SHAP values and base value, and $\pi(S)$ is the Shapley kernel weight corresponding to subset S.

To create an easier understanding of the problem, we will present it in matrix notation. By defining a diagonal matrix $\mathbf{D}$ with entries $\sqrt{\pi_i}$, and applying it to both $\mathbf{X}$ (binary indicator matrix) and $\mathbf{y}$ (target vector), in the form of $\mathbf{X_w} = \mathbf{DX}$ and $\mathbf{y_w} = \mathbf{Dy}$, we obtain a simple least squares equation in the form of:

$$L(f, \phi, \pi) = \|\mathbf{y_w} - \mathbf{X_w}\phi\|^2 \tag{3.19}$$

The reasoning behind defining matrix $\mathbf{D}$ as the square root of the weights comes from the following derivation

$$\|\mathbf{y_w} - \mathbf{X_w}\boldsymbol{\phi}\|^2 = \|\mathbf{Dy} - \mathbf{DX}\|^2 \tag{3.20}$$

$$\|\mathbf{y_w} - \mathbf{X_w}\boldsymbol{\phi}\|^2 = (\mathbf{D}(\mathbf{y} - \mathbf{X}\boldsymbol{\phi}))^T(\mathbf{D}(\mathbf{y} - \mathbf{X}\boldsymbol{\phi})) \tag{3.21}$$

$$\|\mathbf{y_w} - \mathbf{X_w}\boldsymbol{\phi}\|^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\phi})^T\mathbf{D}^T\mathbf{D}(\mathbf{y} - \mathbf{X}\boldsymbol{\phi}) \tag{3.22}$$

$$\|\mathbf{y_w} - \mathbf{X_w}\boldsymbol{\phi}\|^2 = (\mathbf{y} - \mathbf{X}\boldsymbol{\phi})^T\mathbf{W}(\mathbf{y} - \mathbf{X}\boldsymbol{\phi}) \tag{3.23}$$

Therefore, $\mathbf{W} = \mathbf{D}^T\mathbf{D} = \text{diag}(\pi_1, \ldots, \pi_n)$, which implies $\mathbf{D} = \sqrt{\mathbf{W}}$. This transformation avoids the need to directly manipulate the full weight matrix $\mathbf{W}$, while preserving the correct influence of each weighted term.

The solution for the least squares problem is

$$\boldsymbol{\phi} = (\mathbf{X_w}^T\mathbf{X_w})^{-1}\mathbf{X_w}\mathbf{y_w}. \tag{3.24}$$

Yielding the values of ϕ_A, ϕ_B, ϕ_C, and ϕ_0 that best explain the model's output as a weighted linear combination of individual feature contributions. The high weights on the empty and full subsets enforce the constraints

$$\phi_0 = f(\emptyset) \tag{3.25}$$

$$\phi_0 + \phi_A + \phi_B + \phi_C = f(\{A, B, C\}) \tag{3.26}$$

ensuring the explanation satisfies the local accuracy property.

The numerical matrices and vectors to be used in the system are shown next.

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} 5 \\ 8 \\ 6 \\ 7 \\ 10 \\ 12 \\ 11 \\ 15 \end{bmatrix} \quad \mathbf{D} = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.577 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.577 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.577 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.577 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.577 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.577 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

$$\boldsymbol{\phi} = \begin{bmatrix} \phi_0 & \phi_A & \phi_B & \phi_C \end{bmatrix}$$

Solving the least squares problem using the Python programming language (Code in Appendix B.4), we have

Feature	Value
A	3.833341
B	2.333341
C	3.833341
Base Value (ϕ_0)	4.999889
Sum of SHAP Values	14.999911
$f(x)$ from table	15.000000

Table 3.9: SHAP values for each feature, base value ϕ_0, and predicted output $f(x)$ for the full input $\{A, B, C\}$.

Table 3.9 presents the SHAP values obtained from the Kernel SHAP regression. Each value represents the estimated contribution of a feature to the model prediction for the full input instance $\{A, B, C\}$. The base value ϕ_0 corresponds to the model output when no features are known, $f(\emptyset) = 5.0$. Feature A contributes approximately 3.83, feature B contributes 2.33, and feature C contributes another 3.83.

The sum of these contributions, along with the base value, yields a total of approximately 15.0, which exactly matches the model output $f(\{A, B, C\})$, result presented in Table 3.7. This confirms that the explanation satisfies the local accuracy property, where the prediction is fully decomposed into the additive effects of each feature.

It is important to note, also, that the Shapley value calculated in Section 3.6.2.1 for feature A is compatible with the one calculated via Kernel SHAP in this section, which is expected.

Although our example involves only a single model and does not allow for a direct empirical verification of the consistency and missingness properties, the SHAP framework guarantees that these properties are theoretically satisfied.

With the theoretical foundation established, we now turn to the implementation chapter, where we explain the code, discuss the challenges encountered, and present the solutions applied.

Chapter 4

Implementation

This Chapter explains the details and challenges of the code implementation, along with the proposed solutions. Code parts containing the basic usage of Python and its libraries will be omitted. Key code blocks can be found in Appendix B.

Figure 4.1 provides an overview of the libraries utilized in each building block of this project's Machine Learning workflow. It is important to mention that the K-Nearest Neighbors is part of the "Data Processing" arrow, and the grey rectangle represents the nested cross-validation building blocks.

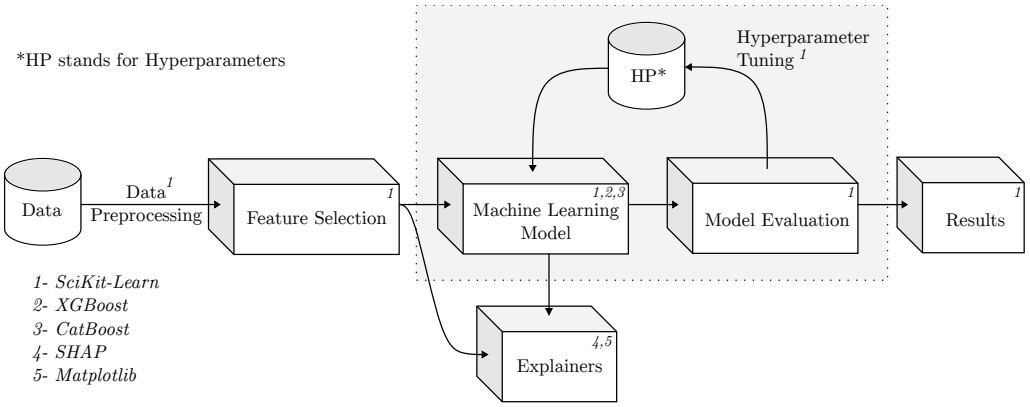


Figure 4.1: Project's workflow with Python libraries corresponding to each block. The grey area corresponds to the nested cross-validation block.

4.1 K-Nearest Neighbors Imputer

The K-Nearest Neighbors Imputer, as detailed in Section 3.2.2.2, is an algorithm responsible for imputing missing values based on the nearest entries of a specific entry in the dataset.

In Python, the SciKit-Learn library has the `KNNImputer` method implemented, that partially meets the needs for this project. This method calculates the Euclidean distance as expected, ranks the nearest neighbors, but only gives the option of aggregating the neighbor values using the mean. The mean aggregation comes as a limitation, as the majority of columns in this project are discrete (ordinal and categorical) and it would possibly yield a result that is not part of the allowed discrete values of the analyzed column. For that reason, the median and mode were chosen as the ordinal and categorical feature columns imputation strategy, respectively.

To address this limitation, the official SciKit-Learn [36] GitHub repository[1] was forked, and modifications were locally made to the `KNNImputer` implementation and Algorithm 1. After applying the necessary changes, the package was rebuilt to integrate the updated version seamlessly.

A key property of the median and mode that guided the implementation of these modifications is the fact that, for an odd number of neighbors, the median and the mode yield the same result in the case of binary values, whereas for ordinal and continuous variables the median still returns the central value. Since binary categorical values were the only case where the mode would have been used, it was determined that implementing the median alone is sufficient for this project.

The primary modification involved adding a new parameter, `imputation_strategy`, allowing users to select different aggregation methods, such as mean and median, depending on the nature of the data. To implement this, changes were made to two key functions within the `KNNImputer` class: `_calc_impute` and `process_chunk`. The

[1] `https://github.com/scikit-learn/scikit-learn`

modified code can be seen in Appendix B, and the base implementation with all the original code on GitHub [2].

The `_calc_impute` function, which computes imputed values based on the selected nearest neighbors, was updated to support different aggregation methods beyond the default mean. When the imputation strategy is set to "median" or "mean", the function now calculates the corresponding statistic instead of automatically averaging the values.

The `process_chunk` function, which processes data in batches to improve computational efficiency, was modified to integrate the `imputation_strategy` parameter. This update ensures that the chosen imputation method is systematically applied to all missing values, maintaining the consistency and integrity of categorical and ordinal data.

After implementing these modifications, the package was rebuilt, allowing the customized `KNNImputer` to be used in the project. This enhanced version ensures more appropriate handling of missing values in discrete feature columns.

In this study, the number of features K that generated the best results was $k = 5$. This number was found running the full ML pipeline for K = 1,3 and 5 and analysing their results.

4.2 Nested Cross-Validation with Grid Search and Recursive Feature Elimination

This phase of the project involves the implementation of Recursive Feature Elimination, nested cross-validation, and grid search. These techniques were applied to reduce the number of variables, assess model performance, and optimize hyperparameters, respectively.

[2] `https://github.com/scikit-learn/scikit-learn/blob/1.5.X/sklearn/impute/_knn.py`

To ensure greater control over each step and due to the absence of pre-implemented nested cross-validation in existing Python libraries, a custom implementation was developed. While Recursive Feature Elimination (RFE) and other essential pre-implemented utility functions were utilized for feature selection and model evaluation, the remaining components were implemented from scratch to enhance flexibility and control in the evaluation process.

4.2.1 Recursive Feature Elimination Implementation

While developing the code to implement the aforementioned steps, computational cost emerged as a significant bottleneck, especially in the RFE step. In a direct implementation, the RFE algorithm would be executed in every iteration over the outer and inner folds for each hyperparameter set under evaluation, using the hyperparameters in the model to be used inside the RFE. This process meant that for each combination of outer fold, inner fold, and hyperparameter set, the feature selection process (RFE) had to be repeated multiple times, which led to a large number of model training and testing cycles. As a result, the computational load became very high, making it time-consuming to evaluate the models efficiently. To address this, optimisations were necessary to speed up the process.

This challenge introduced a tradeoff between the number of hyperparameter configurations to be tested and the feature elimination technique to be used, as they represented the most time consuming parts of the code. To mitigate this issue, an alternative implementation was adopted, leveraging the base model classifier without incorporating hyperparameter tuning through grid search. This approach, detailed in Algorithm 4, eliminated the need to rerun RFE for every hyperparameter set, significantly reducing computational complexity while preserving the integrity of the feature selection process. It is important to notice that the splits are stratified, taking into account the fact that the dataset is imbalanced.

Algorithm 4 Storing Best Columns per outer and Inner Folds pair

Input: Dataset $\mathcal{D}$, model $\mathcal{M}$, inner cross-validation $\mathcal{I}$, outer cross-validation $\mathcal{O}$

Output: Selected columns dictionary $\mathcal{S}$

1: Initialize cols per fold $\mathcal{S} \leftarrow \{k : [] \text{ for } k \in \{0, 1, \ldots, \text{number of outer folds} - 1\}\}$
2: Initialize current fold cols $\mathcal{S}_{fold} \leftarrow 0$
3: **for** each outer fold $\omega \in \mathcal{O}$ from $\mathcal{D}$ **do**
4: **for** each inner fold $\iota \in \mathcal{I}$ from ω **do**
5: $\mathcal{S}_{fold} \leftarrow$ `RFE` columns over ι with model $\mathcal{M}$
6: $\mathcal{S}$`[outer fold index][inner fold index]` $\leftarrow \mathcal{S}_{fold}$
7: **end for**
8: **end for**
9: **return** $\mathcal{S}$

The only difference between both of the possible mentioned implementations is that, in Algorithm 4, the model $\mathcal{M}$ to be optimized during the RFE uses the default hyperparameters and stores the selected features, whereas in the direct implementation, the model would make use of different hyperparameters each round and possibly get different features for the same inner and outer fold pair. In addition to reducing the computational cost, avoiding the use of the same hyperparameters both in the RFE and the grid search can prevent overfitting [59] and yield more generalizable results, as shown in a small deviation between the inner and outer folds, to be presented in Chapter 5.

4.2.2 Nested Cross-validation with Grid Search

The code implementation in this project is adapted from the one present in the book "Introduction to Machine Learning with Python" by Müller and Guido [60], and followed the previously explained concepts in Sections 3.5.1 and Section 3.5.2. The main difference between this project's implementation and the one present in the book is that here more metrics were exported, and the fact that features used for each outer and inner fold pair are different, as explained in Section 4.2.1.

Algorithm 5 Nested Cross-Validation with Grid Search for Model Selection

Input: Dataset $\mathcal{D}$, model $\mathcal{M}$, hyperparameter grid $\mathcal{G}$, columns per fold dictionary $\mathcal{S}$, Outer cross-validation $\mathcal{O}$ with k folds, inner cross-validation $\mathcal{I}$ with j folds, evaluation metrics m

Output: Performance metrics and best hyperparameters θ_{best}

for each outer fold $(\mathcal{X}_{train}, \mathcal{X}_{test})$ in $\mathcal{O}(\mathcal{D})$ **do**
    for each hyperparameter set θ in $\mathcal{G}$ **do**
        for each inner fold $(\mathcal{X}_{inner\_train}, \mathcal{X}_{inner\_val})$ in $\mathcal{I}(\mathcal{X}_{train})$ **do**
            $\mathcal{S}_{pair} \leftarrow \mathcal{S}$`[outer fold index][inner fold index]`
            Train $\mathcal{M}$ with θ on $\mathcal{X}_{inner\_train}$ with features $\mathcal{S}_{pair}$
            Evaluate $\mathcal{M}$ on $\mathcal{X}_{inner\_val}$ using m
        end for
        Compute mean inner validation score
    end for
    Select best hyperparameters θ_{best} based on highest inner score
    Train $\mathcal{M}$ with θ_{best} on $\mathcal{X}_{train}$ with features $\mathcal{S}_{pair}$ ▷ With $\mathcal{S}_{pair}$ from the best inner fold
    Evaluate $\mathcal{M}$ on $\mathcal{X}_{test}$ using m
end for
return Outer and inner fold performance metrics and best hyperparameters

4.3 Explainable Artificial Intelligence

The Explainable Artificial Intelligence implementation of this study consisted of applying pre-built methods of the SHAP library [61] to the chosen model, trained with the selected hyperparameter and best features, all of those found through the nested cross-validation and grid search combination. Figure 4.2 represents the logical flow for the XAI implementation in Python.

The "Data" block in the diagram represents the SNAC dataset after imputing missing values using the KNN Imputer, as described in Sections 3.2.2.2 and 4.1. Following imputation, the dataset was filtered to retain only the selected features, resulted from the Recursive Feature Elimination algorithm, along with the target

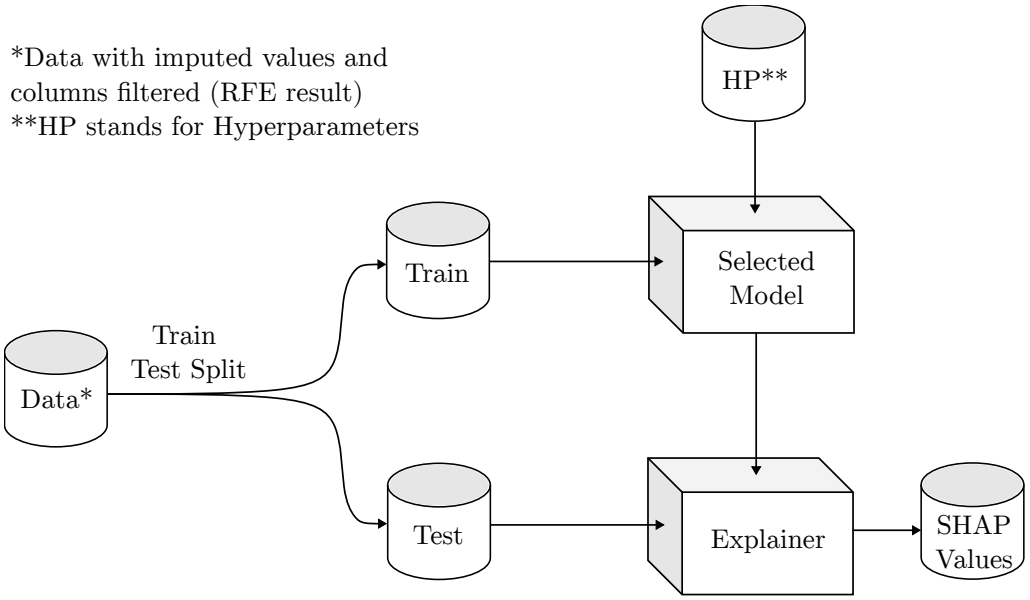


Figure 4.2: Explainable Artificial Intelligence implementation diagram.

column.

In order to keep the analysis consistent and generalisable, the data was split into train and test sets, using the `train_test_split` method from the Scikit-Learn Library [36], stratified by the target column to take the imbalance into account. The "Train" block is then fed into the chosen model, along with the hyperparameters, represented by "HP" in the diagram. In the training step, the models are tested and their results analysed, to ensure a satisfactory model to be "explained".

As part of the implementation, SHAP's `TreeExplainer` method is utilized to generate the "explainer," which takes the trained model and test data as inputs. This method is used to explain the output of ensemble tree models but, in case a model-agnostic explainer is needed, the raw "Explainer" method is useful. The aforementioned process is represented by the "Explainer" block in the workflow. Once the explainer is created, SHAP values can be computed for the test data, enabling further analysis.

In this study, all required SHAP plotting methods relied solely on the SHAP values, produced by the explainer, as the primary input. Therefore, the extracted SHAP values provide sufficient information for the complete XAI analysis and visualization implemented in this work.

The specific methods used and the plots will be presented in the following chapter, in Section 5.3.

Chapter 5

Results

In this chapter, we will present the top-performing results for each trained Machine Learning model, and apply the Explainable Artificial Intelligence tools discussed in Section 3.6 to enhance our understanding of its predictions.

Furthermore, the results will be compared to those reported in the reference study by Dallora et al.[10], allowing for a comprehensive evaluation of improvements or differences in ROC AUC, sensitivity, and other key metrics.

With that, we will be finally able to extract possible modifiable risk factors from dementia using Machine Learning and Explainable Artificial Intelligence Techniques and enlighten the contributions of using black-box models combined with XAI techniques.

5.1 Model Selection

The models implemented in this study were the XGBoost Classifier (`XGB`) [62], CatBoost Classifier (`CBC`) [63], and Balanced Random Forest Classifier (`RFC`) [64]. Those models were chosen because of their strong predicting power and native tools, in the form of hyperparameters, to handle class imbalance. We utilised nested cross-validation to assess the performance of each model, ensuring that the results are robust and generalisable.

Section 3.5 provided a detailed explanation of the nested cross-validation framework, emphasising its role in ensuring reliable and generalisable model evaluation metrics, along with its advantages and trade-offs. Building on this foundation, our focus now shifts to selecting the best-tuned model for each of the three classifiers.

To start this analysis, the methodology for choosing the best tuned model will be discussed. Secondly, we will explain how to decide which of the three models is the best for the present challenge.

As stated in Section 3.5.2, the nested cross-validation is responsible for assuring that no data is leaked during feature selection, as well as to analyse overfitting for each model-hyperparameter combination. In order to be able to assure that the model is not overfitted, one should compare the inner and outer fold results for a given fold. In this step, an overfitted model would present a good result for the inner fold and a poor result for the outer fold. This behaviour would mean the model "learned" the data it has been trained with, but is not generalised enough to perform well on data other than the trained one. On the other hand, if a model shows a small deviation between the inner and outer fold results, it gives a clue that the model is generalisable and not overfitted.

As shown in Table 5.1, inner and outer fold results have a maximum absolute deviation $|\frac{Outer-Inner}{Outer}|$ of 5.32 % for the XGBoost model, 4.35 % for the CatBoost and 5.83 % for the Balanced Random Forest in the ROC AUC metric, suggesting that the models are not overfitted and are acceptable generalisations [65].

Fold	**XGBoost**	**CatBoost**	**Bal. Random Forest**
`Fold_0`	3.47 %	0.05 %	0.74 %
`Fold_1`	4.17 %	2.75 %	5.83 %
`Fold_2`	5.32 %	3.37 %	5.19 %
`Fold_3`	5.03 %	4.35 %	5.16 %
`Fold_4`	3.01 %	0.39 %	1.85 %

Table 5.1: ROC AUC deviation values for each model and fold.

Tables 5.2, 5.3 and 5.4 represent the full results for the XGBoost, CatBoost and Balanced Random Forest models, respectively.

Fold	Acc.	ROC AUC	PR AUC	Sens.	Spec.	Prec.
Outer_0	0.637	0.816	0.489	0.842	0.606	0.242
Outer_1	0.648	0.777	0.363	0.778	0.630	0.230
Outer_2	0.669	0.831	0.426	0.944	0.630	0.266
Outer_3	0.593	0.821	0.332	0.944	0.543	0.227
Outer_4	0.600	0.810	0.292	0.889	0.559	0.222
Inner_0	0.634	0.787	0.379	0.824	0.608	0.250
Inner_1	0.637	0.809	0.411	0.884	0.601	0.262
Inner_2	0.620	0.787	0.378	0.846	0.587	0.245
Inner_3	0.602	0.779	0.393	0.834	0.567	0.233
Inner_4	0.625	0.785	0.396	0.847	0.590	0.244

Table 5.2: Outer and Inner fold results for the XGBoost Classifier model.

Fold	Acc.	ROC AUC	PR AUC	Sens.	Spec.	Prec.
Outer_0	0.685	0.812	0.503	0.737	0.677	0.255
Outer_1	0.724	0.794	0.506	0.722	0.724	0.271
Outer_2	0.759	0.842	0.366	0.889	0.740	0.327
Outer_3	0.703	0.840	0.393	0.889	0.677	0.281
Outer_4	0.697	0.803	0.259	0.778	0.685	0.259
Inner_0	0.724	0.812	0.362	0.749	0.723	0.300
Inner_1	0.725	0.816	0.414	0.734	0.724	0.304
Inner_2	0.709	0.814	0.402	0.744	0.705	0.291
Inner_3	0.713	0.803	0.393	0.735	0.708	0.290
Inner_4	0.720	0.806	0.404	0.734	0.717	0.291

Table 5.3: Outer and Inner fold results for the CatBoost Classifier model.

Fold	Acc.	ROC AUC	PR AUC	Sens.	Spec.	Prec.
Outer_0	0.610	0.784	0.430	0.789	0.583	0.221
Outer_1	0.669	0.749	0.355	0.667	0.669	0.222
Outer_2	0.669	0.833	0.402	0.889	0.638	0.258
Outer_3	0.545	0.736	0.231	0.889	0.496	0.200
Outer_4	0.621	0.788	0.269	0.889	0.583	0.232
Inner_0	0.638	0.789	0.381	0.806	0.613	0.249
Inner_1	0.640	0.793	0.397	0.843	0.610	0.256
Inner_2	0.630	0.790	0.388	0.835	0.602	0.252
Inner_3	0.640	0.774	0.347	0.820	0.612	0.253
Inner_4	0.640	0.774	0.360	0.806	0.612	0.248

Table 5.4: Outer and Inner fold results for the Balanced Random Forest Classifier model.

Proceeding with the model selection, we will now investigate how to select the fold that will represent each model and, after that, compare the three selected models. In order to choose the most reliable results from each model, aiming to have a fair and realistic comparison between the models, the model will be chosen based on its median Receiver Operating Characteristic Area Under the Curve outer fold results.

It is important to mention that each fold was trained with their particular hyperparameters and selected variables, as explained in Section 4.2 and shown in Appendix C.

The ROC AUC is a powerful metric when comparing models [3] as it represent the models' ability to distinguish between classes and take both false positives and false negatives into account, an important characteristic especially when dealing with imbalanced data. Moreover, as one of the goals is to compare the present results the baseline study by Dallora et. al [10], it makes sense for a fair comparison to use the same model selection methodology.

In Table 5.5, the chosen fold for each model is presented, following the aforementioned median ROC AUC framework.

Model	Acc.	ROC AUC	PR AUC	Sens.	Spec.	Prec.
`CatBoost`	0.685	0.812	0.503	0.737	0.677	0.255
`XGBoost`	0.637	0.816	0.489	0.842	0.606	0.242
`Random Forest`	0.610	0.784	0.430	0.789	0.583	0.221

Table 5.5: Median outer folder scores for each model.

The results presented in Table 5.5 highlight a "tie" between the CatBoost and XGBoost Classifiers, and the fact that both outperform the Balanced Random Forest Classifier. It is safe to say it is a tie as there is a 0.5% deviation between the two models' results. Furthermore, here is a noticeable sensitivity versus specificity tradeoff. In CatBoost, the model is more conservative towards the positive class, fact shown by the lower recall, higher specificity and precision, while XGBoost presents itself as a more agressive one, with a considerably higher recall, lower specificity and lower precision. The "agressiveness" of each model could be changed varying its decision threshold as detailed in 3.1.

Model	Acc.	ROC AUC	PR AUC	Sens.	Spec.	Prec.
XGBoost	0.029	0.018	0.070	0.063	0.036	0.016
CatBoost	0.026	0.019	0.093	0.073	0.026	0.026
Balanced RF	0.046	0.034	0.076	0.088	0.059	0.019

Table 5.6: Standard deviation of outer fold results for each model across all performance metrics.

Table 5.6 presents the standard deviation of each performance metric across the outer folds for all evaluated models. This analysis is essential as it highlights the consistency and robustness of each model when tested on unseen data. Among the compared models, XGBoost demonstrates notably lower variability across most metrics, particularly in ROC AUC and Sensitivity, which are critical for the problem

under investigation. This suggests that XGBoost may offer more stable and reliable performance in real-world scenarios.

To mitigate the influence of the sensitivity-specificity tradeoff in the final model selection, we rely on a set of aggregated performance metrics. These include the F1-score, Balanced Accuracy, Matthews correlation coefficient (MCC), Cohen's kappa, and Geometric Mean (G-Mean), as they provide a more robust evaluation by incorporating both class-wise performance and the balance between precision and recall. By considering these comprehensive metrics, we ensure that the final selection is based on a well-rounded assessment rather than being disproportionately influenced by the sensitivity-specificity tradeoff.

Model	**F1**	**Bal. Acc.**	**MCC**	**Kappa**	**G-Mean**
`CatBoost`	0.373	0.703	0.281	0.222	0.702
`XGBoost`	0.376	0.724	0.303	0.219	0.715
`Random Forest`	0.345	0.686	0.251	0.178	0.678

Table 5.7: Median outer folder combined metrics scores for each model.

Upon evaluating the selected performance metrics, XGBoost outperformed CatBoost in four out of the five aggregated metrics (F1-score, Balanced Accuracy, MCC, and G-Mean) while CatBoost only showed a slight advantage in Cohen's Kappa. This consistent superiority across multiple evaluation criteria indicates that XGBoost provides a more reliable performance for the given task. Therefore, based on this comprehensive assessment, XGBoost is selected as the final model for this project, as it demonstrates the best overall balance between predictive power and generalization capability.

5.2 Comparison with the Baseline Study

In order to compare the XGBoost results with the ones found through an explainable decision tree in the baseline study by Dallora et al. [10], the performances of the median ROC AUC outer fold results will be analysed.

Model	ROC AUC	Sensitivity	Precision
`Baseline`	0.735	0.722	0.289
`XGBoost`	0.816	0.842	0.242

Table 5.8: Decision Tree (Baseline) and XGBoost comparison.

From Table 5.8, the comparison between XGBoost and the baseline Decision Tree model reveals a notable improvement in classification performance. XGBoost achieves a higher ROC AUC (0.816 vs. 0.735), indicating a better overall discrimination ability between classes.

Additionally, it exhibits higher sensitivity (0.842 vs. 0.722), meaning it correctly identifies more positive cases than the baseline model. However, this comes at the cost of a slightly lower precision (0.242 vs. 0.289), suggesting that while XGBoost captures more true positives, it also produces more false positives. Despite this expected tradeoff, already explained in Section 5.1, the overall improvement in ROC AUC and sensitivity demonstrates XGBoost's superior predictive power when compared to the baseline approach.

5.3 Explainable Artifical Intelligence

After having decided on the best model, including its features and hyperparameters, it is possible to apply XAI techniques in order to obtain insights on the "thought process" behind every prediction. Despite a superior performance when compared to natively explainable models, until this point the model's reasoning behind the predictions is still not possible to be extracted.

In this section, the SHapley Additive exPlanations (SHAP) Explainable Artificial Intelligence method will be applied in XGBoost, the model chosen in Section 5.1. The primary objective is to identify potential factors that may impact the risk of developing dementia. SHAP enables us to gain insight into these influences in a model-agnostic manner when analysing the features, allowing predictions from any black-box model to be explained.

To start the analysis, the mean of the absolute SHAP values was plotted in Figure 5.1 for each of the 15 features used to train the XGBoost algorithm. The SHAP bar plot demonstrates the global importance of each input feature by visualising the mean absolute SHAP values, a measure that quantifies how much each feature, on average, contributes to the model's output. Features are ranked in descending order of importance, with the most influential features at the top of the plot.

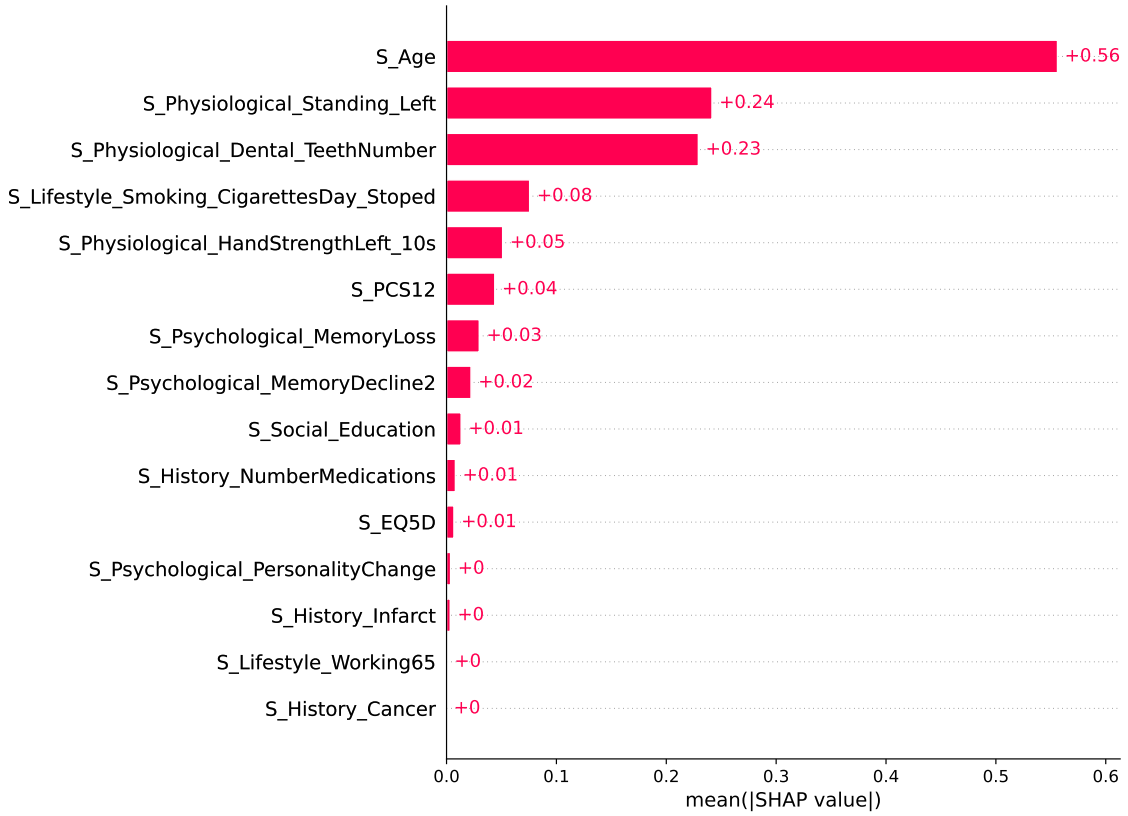


Figure 5.1: Absolute mean SHAP values plot for each of the 15 selected features .

From this graph, we can extract that the most important feature is `S_Age`, which is expected since the risk of dementia is widely known to increase with age [66]. The second most important feature is `S_Physiological_Standing_Left`, that represents the longest time, in seconds, out of three tries, that the subject can stand with the left leg without support. This result follows the findings from the baseline study by Dallora et al. [10], where in the resulting decision tree, for subjects younger than 75 years of age, that feature was present as the next decision node (`S_Physiological_Standing_Left` $\geq 7.8\ s$) after the `S_Age` < 75 split.

Furthermore, `S_Physiological_Dental_TeethNumber` also demonstrates a meaningful contribution to the model, supporting previous findings [67] that associate a higher number of teeth with a lower risk of dementia and increased life expectancy. In order to confirm that, SHAP's bar plot hierarchical clustering feature was used. This feature was primarily developed to test feature redundancy, training a XGBoost model for each pair of input features and comparing the model's loss [58]. In this analysis, the clustering was implemented with a 0.8 cutoff, which means that features with at least 20% redundancy were included. This value was chosen as, except for the age and standing in one leg features that had more than 50% redundancy (cutoff 0.5) when compared, the other pairs only started showing for redundancies less or equal than 20% (cutoff 0.8).

The results of this analysis, shown in Figure 5.2, confirmed the hypothesized correlation. Age, the ability to stand on one leg, and the number of teeth were clustered together, indicating that these features shared at least 20% redundancy and thus were significantly correlated with each other [58].

Another interesting observation that also agrees on the baseline study is that the approximate amount of cigarettes smoked per day until stopped, represented by the feature column named `S_Lifestyle_Smoking_CigarettesDay_Stoped`, plays a significant role in the predictions. In the baseline study by Dallora et al. [10], this feature is one edge away from the root node, which means that, for subjects younger than 75 years of age, a branch goes to subjects that smoked less than 7.5 cigarettes per day on average and the other split to the ones that smoked 7.5 or more.

The SHAP results presented in Figure 5.1 corroborate with several studies that affirm that physical factors are intrinsically related to developing dementia [68, 69], and that lifestyle [70, 71] and smoking [72] are linked with the disease's progress as well.

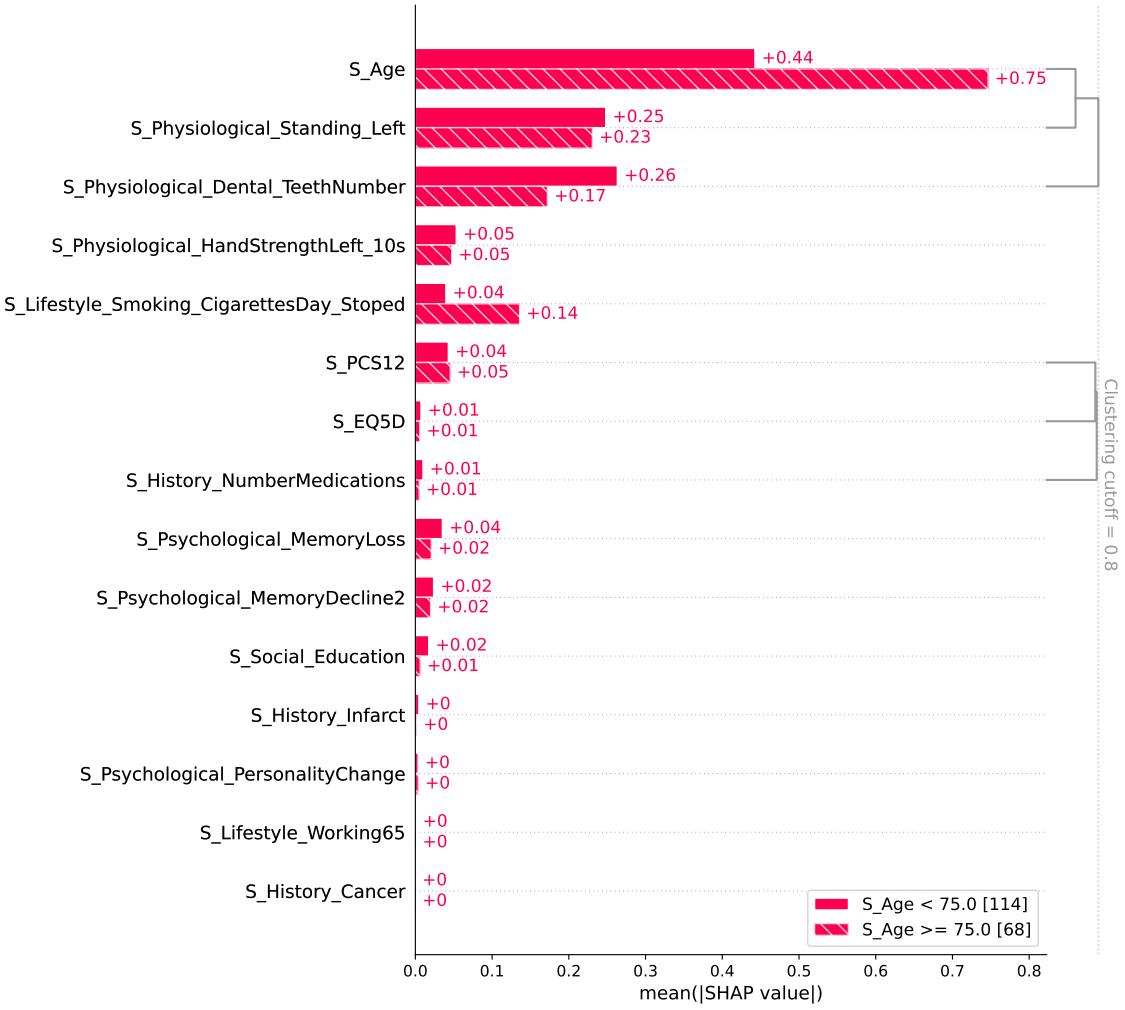


Figure 5.2: Absolute mean SHAP values plot for two cohorts with clustering.

In addition to that, Figure 5.2 shows the use of "cohorts", another feature of the bar plots present in the SHAP's library implementation [61]. In this feature, calling the bar plot with "N" as the cohort argument will create "N" groups that optimally separate the SHAP values of the instances using the Decision Tree Regressor from Python's Scikit-Learn Library [36]. For instance, the boundary between the two cohorts was determined to be an age threshold of 75 years of age, separating individuals older or younger than this value. This value is also the first decision tree split (root node) in Dallora's study [10], indicating that to be a compatible value.

With this segmented analysis, it stands clear that for older individuals the most impactful feature is their age. On the other hand, we also notice that for individuals younger than 75 years the other features play a more significant role, except from the smoking, that showed to have more impact in older subjects, and the others

that did not have an expressive difference after the age cutoff.

Another useful source of information is the "Beeswarm" plot, introduced in Figure 5.3. It allows us to analyze not only the magnitude of SHAP values for each feature, but also their direction of influence on each prediction. The color indicates the subject's feature value, while the position along the x-axis shows whether the feature has a positive or negative impact on the prediction for each instance.

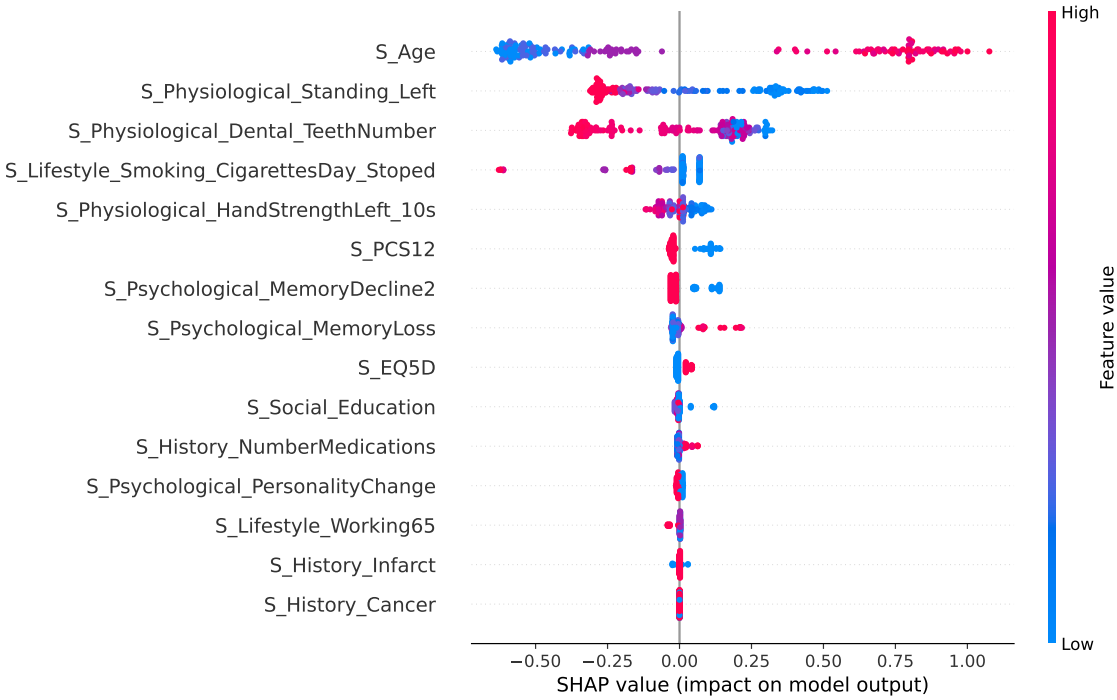


Figure 5.3: Beeswarm plot over all test data.

From Figure 5.3, we can extract that all the physiological features presented herein have an impact on the dementia diagnosis not only in absolute value, but also in their contribution direction, which once again corroborates with the baseline study [10] and the literature [68, 69]. The beeswarm plot also makes it possible to identify that the `S_PCS12`, `S_Psychological_MemoryDecline2` and `S_EQ5D` had a clear split in the SHAP value = 0 vertical line. The absence of mixed colors in each side of the line shows that those features had a clear contribution to the final dementia diagnosis, and despite their relatively low magnitude (low module of the SHAP value), they can be helpful specially when combined to other analysis.

Specifically, `S_PCS12` (Physical Component Score from the SF-12 health survey [27]), a framework to assess both physical and mental health status and `S_EQ5D` (EuroQol 5-Dimensions questionnaire [22]), that evaluates physical, mental, and social well-being are interesting cases. These assessments, that use carefully designed questionnaires that respondents complete, aim to provide a standardized "index of health" [22]. From Figure 5.3, those features showed to be correlated to the dementia diagnosis in a 10-year span, This highlights how physical decline, psychological state, and social functioning are interconnected and have the potential to collectively serve as early indicators of dementia risk.

Following the analysis, the "heatmap" plot is displayed in Figure 5.4. It allows for a condensed "entry-centered" study, in which each of the 182 test individuals is a thin column. In this case, entries are organized from left to right in descending order of log odds sum $f(x)$ for a dementia diagnosis, and features are sorted from top to bottom in terms of importance to the model. As presented in Equation 3.14 from Section 3.6.3, since probability is derived from the logistic function of the log odds [56], a decrease in log odds corresponds to a lower probability of dementia. Note that red and blue mean a positive and a negative contribution to dementia, respectively.

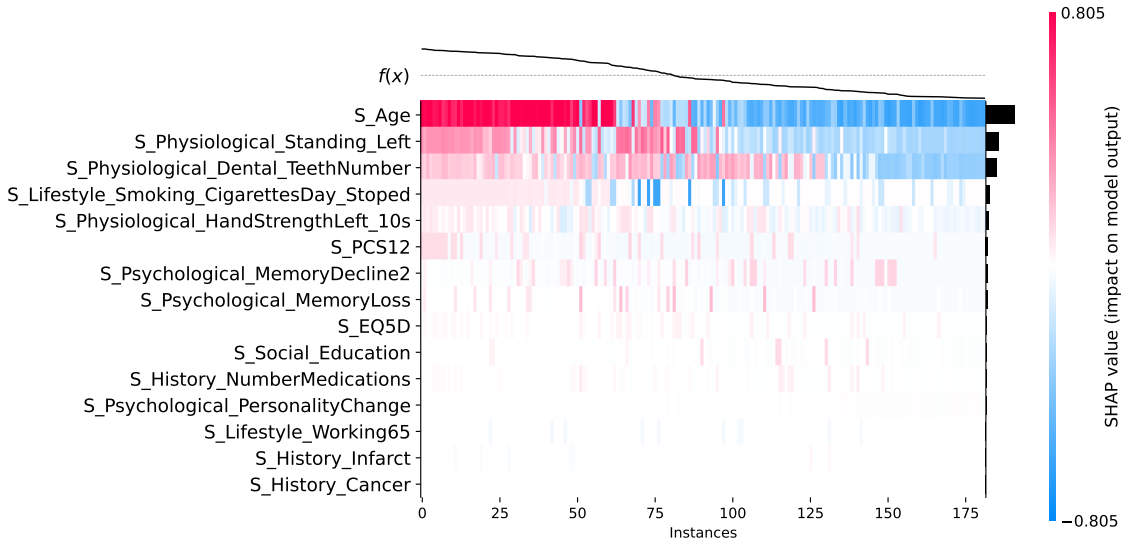


Figure 5.4: Heatmap of the feature importance for all individuals, sorted by their SHAP values log odds sum $f(x)$.

An insight that arises from this plot that again confirms the findings by Dallora et al. [10] and previous analysis is that, for younger individuals (blue tone in `S_Age`), the `S_Physiological_Standing_Left` and `S_Physiological_Dental_TeethNumber` columns appeared as deciding factors for the individual to be diagnosed with dementia ($f(x) > 0$). This is shown in the graph as a blue tone in the age column and red tones in the other columns previously mentioned.

Lastly, the SHAP package also allows for an individual analysis of the results. For instance, the waterfall plot presented in Figure 5.5 shows, for a particular subject, how each feature contributed to the final prediction, from the expected value without any other features $E[f(x)]$ to the curent model output $f(x)$, summing one contribution at a time (in log-odds). In the left-hand side, we are able to see the analised variables in descending order of magnitude for each contribution, as well as their values for specific subject (in grey). We are able to see that the young age (66) and good standing left result (60 s) were the main contributors for the negative prediction, and the number of teeth (19) was the main source of contribution to a positive prediction to dementia diagnosis.

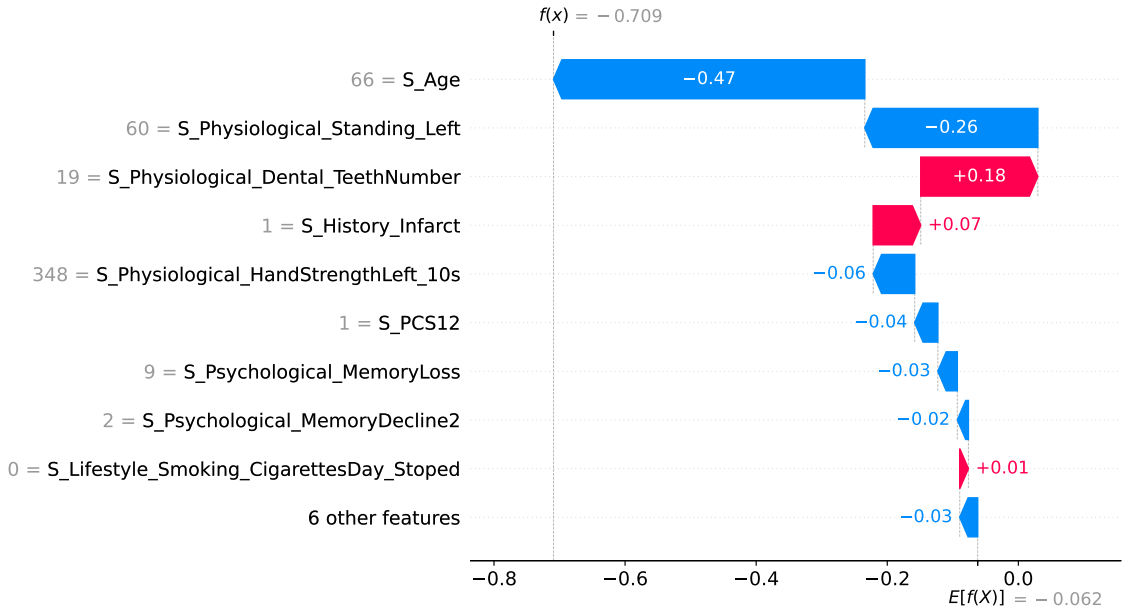


Figure 5.5: Waterfall plot for a correctly predicted subject with no dementia diagnosis at the 10-year mark.

On the other hand, Figure 5.6 highlights a correctly predicted individual with dementia. It is clear to see that, despite its relatively young age (72), the poor one leg stand result (7.02 s) and the low number of teeth (16) influenced the predictor to a positive result for the dementia diagnosis.

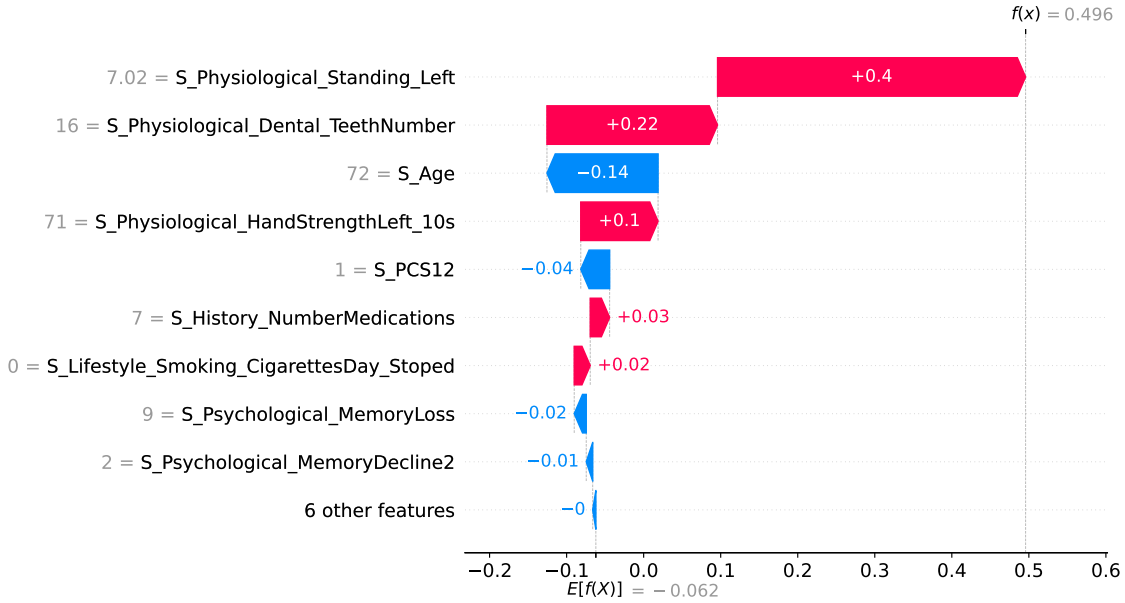


Figure 5.6: Waterfall plot for a correctly predicted subject with positive dementia diagnosis at the 10-year mark.

The application of SHAP within the XAI framework has demonstrated its effectiveness in interpreting complex machine learning models, particularly in dementia risk prediction. By analyzing SHAP values, it is possible to quantify the contribution of each feature to the model's predictions, enabling a comprehensive assessment of feature importance. The versatility of SHAP extends to various analytical approaches, including global importance rankings, cohort-based segmentations, and individual-level explanations through visualizations such as beeswarm plots, heatmaps, and waterfall plots. These methods allow for a detailed examination of how input features influence predictions, providing a systematic approach to identifying correlations, dependencies, and redundancies within the dataset. Additionally, the ability to perform cohort-based analysis highlights the adaptability of SHAP in uncovering nuanced patterns across different subgroups. The results obtained reinforce the potential of XAI techniques in enhancing model transparency and facilitating a structured evaluation of predictive factors, making them essentially useful

for analyzing and interpreting data-driven models.

Chapter 6

Conclusion

Dementia remains one of the leading causes of dependency among the elderly, significantly impacting individuals and healthcare systems worldwide [1]. This study sought to enhance dementia prognosis using non-explainable Machine Learning models while maintaining interpretability through Explainable Artificial Intelligence techniques. By leveraging data from the Swedish National Study on Aging and Care, the proposed methodology aimed to improve predictive accuracy while providing meaningful insights into modifiable risk factors.

The key findings of this research confirm that black-box Machine Learning models, when combined with Explainable Artificial Intelligence methods such as SHAP, can outperform traditional explainable models like Decision Trees in predictive performance while preserving interpretability in a model-agnostic manner. Through a comparative analysis of models including XGBoost, CatBoost, and Balanced Random Forest, it was observed that the integration of explainability tools and with black-box models enables the extraction of insights with superior predictive performance, when compared to problems restricted to explainable models.

Using the SHAP values and the tools provided by the SHAP library implementation, we were able to confirm findings from the baseline study [10], such as the relationship of smoking habits and the standing in one leg test to the dementia diagnosis. Moreover, the results brought to light new insights about the Physical Component Score from the SF-12 health survey and the EuroQol 5-Dimensions questionnaire, that figured as the 6^{th} and 11^{th} most impactful features to the dementia

diagnosis at the 10-year mark.

The ability for experts to analyse individual predictions, as shown in Section 5.3, can also be valuable to fine tune the models and allowing healthcare professionals to interpret model outcomes effectively.

Furthermore, the implementation of Recursive Feature Elimination (RFE) and Nested Cross-Validation with Grid Search ensured that the models were both robust and generalisable. The small deviation between inner and outer folds confirmed that overfitting was mitigated, enhancing the reliability of predictions.

While this research successfully addressed the initial problem statement, certain limitations remain. The dataset's demographic bias, reliance on structured health records, and missing data issues could influence the generalisability of the findings.

Also, an important limitation of this study lies in the interpretative differences between inherently explainable models such as decision trees and post hoc explainability tools like SHAP. While decision tree models provide direct and transparent decision-making paths based on how variables split data to reduce classification error, SHAP offers an approximation of each feature's marginal contribution to a prediction, derived after the model has been trained. As such, SHAP values indicate which features influenced a specific prediction most strongly, but they do not imply causality or direct predictive power in a traditional statistical sense. Therefore, the "top contributors" identified by SHAP should be interpreted as impactful within the context of the trained model rather than as definitive indicators of dementia risk.

Despite not intrinsically being an indicator of dementia due to its approximations, this limitation is mitigated by the fact that the key variables highlighted by SHAP in this study — such as age, quality of life score (EQ5D), and physical and cognitive components of the SF-12 — align closely with those identified by Dallora et al. [10]. This consistency supports the validity of the model's interpretability and reinforces confidence in the insights derived from the XAI techniques applied.

Future work could explore, as stated in Dallora's study, other SNAC datasets to reduce the demographic bias and increase the number of subjects, increasing generalisability and potentially the models' performances. Moreover, SHAP is not limited to tabular data, its library also includes a built-in implementation for XAI in images, making it applicable to fields like medical imaging and computer vision.

In conclusion, this study demonstrates that combining black-box machine learning models with XAI techniques presents a viable pathway for improving dementia prognosis while ensuring transparency. These findings contribute to the growing body of knowledge on AI applications in healthcare, paving the way for more effective early intervention strategies, personalized treatment plans and finding unknown risk factors.

Bibliography

[1] World Health Organization, *Global action plan on the public health response to dementia 2017–2025*. Geneva, World Health Organization, 2017.

[2] Alzheimer's Disease International, *World Alzheimer Report 2024: Global changes in attitudes to dementia*. London, Alzheimer's Disease International, Sep. 2024.

[3] BURKOV, A., *Machine Learning Engineering*. True Positive Incorporated, 2020.

[4] SARKER, I. H., "Machine Learning: Algorithms, Real-World Applications and Research Directions", *SN Computer Science*, v. 2, n. 3, pp. 160, Mar. 2021.

[5] GOODFELLOW, I., BENGIO, Y., COURVILLE, A., *Deep Learning*. MIT Press, Nov. 2016.

[6] HOLZINGER, A., SARANTI, A., MOLNAR, C., *et al.*, "Explainable AI Methods - A Brief Overview". In: Holzinger, A., Goebel, R., Fong, R., *et al.* (eds.), *xxAI - Beyond Explainable AI: International Workshop, Held in Conjunction with ICML 2020, July 18, 2020, Vienna, Austria, Revised and Extended Papers*, Cham, Springer International Publishing, pp. 13–38, 2022.

[7] BARREDO ARRIETA, A., DíAZ-RODRíGUEZ, N., DEL SER, J., *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI", *Information Fusion*, v. 58, pp. 82–115, Jun. 2020.

[8] "Roles and Responsibilities Framework for Artificial Intelligence in Critical Infrastructure", U.S. Department of Homeland Security.

[9] "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act) (Text with EEA relevance)", Jun. 2024, Legislative Body: CONSIL, EP.

[10] DALLORA, A. L., MINKU, L., MENDES, E., *et al.*, "Multifactorial 10-Year Prior Diagnosis Prediction Model of Dementia", *International Journal of Environmental Research and Public Health*, v. 17, n. 18, pp. 6674, Sep. 2020.

[11] LAGERGREN, M., FRATIGLIONI, L., HALLBERG, I. R., *et al.*, "A longitudinal study integrating population, care and social services data. The Swedish National study on Aging and Care (SNAC)", *Aging Clinical and Experimental Research*, v. 16, n. 2, pp. 158–168, Apr. 2004.

[12] LUNDBERG, S. M., LEE, S.-I., "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems*, v. 30, 2017.

[13] LUTZ, W., SANDERSON, W., SCHERBOV, S., "The coming acceleration of global population ageing", *Nature*, v. 451, n. 7179, pp. 716–719, Feb. 2008. Publisher: Nature Publishing Group.

[14] Statistics Sweden, "Population density per sq. km, population and land area by region and sex. Year 1991 - 2023", `http://www.statistikdatabasen.scb.se/pxweb/en/ssd/START__BE__BE0101__BE0101C/BefArealTathetKon/`, Accessed: 2024-11-25.

[15] Statistics Sweden, "Population by age and sex. Year 1860 - 2023", `http://www.statistikdatabasen.scb.se/pxweb/en/ssd/START__BE__BE0101__BE0101A/BefolkningR1860N/`, Accessed: 2024-10-24.

[16] CARUANA, E. J., ROMAN, M., HERNáNDEZ-SáNCHEZ, J., *et al.*, "Longitudinal studies", *Journal of Thoracic Disease*, v. 7, n. 11, pp. E537–E540, Nov. 2015.

[17] CASTRO-ALDRETE, L., MOSER, M. V., PUTIGNANO, G., *et al.*, "Sex and gender considerations in Alzheimer's disease: The Women's Brain Project contribution", *Frontiers in Aging Neuroscience*, v. 15, Mar. 2023. Publisher: Frontiers.

[18] HENDRIKS, S., PEETOOM, K., BAKKER, C., *et al.*, "Global Prevalence of Young-Onset Dementia: A Systematic Review and Meta-analysis", *JAMA Neurology*, v. 78, n. 9, pp. 1080–1090, Sep. 2021.

[19] ANTONOVSKY, A., "The structure and properties of the sense of coherence scale", *Social Science & Medicine*, v. 36, n. 6, pp. 725–733, Mar. 1993.

[20] SAKLOFSKE, D. H., RUM, R., SCHOENBERG, M. R., "Wechsler Adult Intelligence Scale (All Versions)". In: Kreutzer, J., DeLuca, J., Caplan, B. (eds.), *Encyclopedia of Clinical Neuropsychology*, Cham, Springer International Publishing, pp. 1–11, 2017.

[21] LIVINGSTON, G., BLIZARD, B., MANN, A., "Does sleep disturbance predict depression in elderly people? A study in inner London", *The British Journal of General Practice: The Journal of the Royal College of General Practitioners*, v. 43, n. 376, pp. 445–448, Nov. 1993.

[22] BROOKS, R., "EuroQol: the current state of play", *Health Policy*, v. 37, n. 1, pp. 53–72, Jul. 1996.

[23] KATZ, S., "Assessing Self-maintenance: Activities of Daily Living, Mobility, and Instrumental Activities of Daily Living", *Journal of the American Geriatrics Society*, v. 31, n. 12, pp. 721–727, 1983. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1532-5415.1983.tb03391.x.

[24] LAWTON, M. P., BRODY, E. M., "Assessment of Older People: Self-Maintaining and Instrumental Activities of Daily Living1", *The Gerontologist*, v. 9, n. 3_Part_1, pp. 179–186, Oct. 1969.

[25] FOLSTEIN, M. F., FOLSTEIN, S. E., MCHUGH, P. R., ""Mini-mental state": A practical method for grading the cognitive state of patients for the clinician", *Journal of Psychiatric Research*, v. 12, n. 3, pp. 189–198, Nov. 1975.

[26] AGRELL, B., DEHLIN, O., "The clock-drawing test", *Age and Ageing*, v. 27, n. 3, pp. 399–403, May 1998.

[27] JENKINSON, C., LAYTE, R., "Development and Testing of the UK SF-12", *Journal of Health Services Research & Policy*, v. 2, n. 1, pp. 14–18, Jan. 1997. Publisher: SAGE Publications.

[28] MONTGOMERY, S. A., ÅSBERG, M., "A New Depression Scale Designed to be Sensitive to Change", *The British Journal of Psychiatry*, v. 134, n. 4, pp. 382–389, Apr. 1979.

[29] BISHOP, C. M., *Pattern recognition and machine learning*, Information science and statistics. New York, Springer, 2006.

[30] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J. H., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009. Google-Books-ID: eBSgoAEACAAJ.

[31] RUBIN, D. B., "Inference and Missing Data", *Biometrika*, v. 63, n. 3, pp. 581–592, 1976. Publisher: [Oxford University Press, Biometrika Trust].

[32] MEMON, S. M., WAMALA, R., KABANO, I. H., "A comparison of imputation methods for categorical data", *Informatics in Medicine Unlocked*, v. 42, pp. 101382, Jan. 2023.

[33] LI, J., GUO, S., MA, R., *et al.*, "Comparison of the effects of imputation methods for missing data in predictive modelling of cohort study datasets", *BMC Medical Research Methodology*, v. 24, n. 1, pp. 41, Feb. 2024.

[34] ZHANG, Z., "Missing data imputation: focusing on single imputation", *Annals of Translational Medicine*, v. 4, n. 1, pp. 9, Jan. 2016.

[35] ACUñA, E., RODRIGUEZ, C., "The Treatment of Missing Values and its Effect on Classifier Accuracy". In: Banks, D., McMorris, F. R., Arabie, P., *et al.* (eds.), *Classification, Clustering, and Data Mining Applications*, pp. 639–647, Berlin, Heidelberg, 2004.

[36] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., *et al.*, "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, n. 85, pp. 2825–2830, 2011.

[37] TROYANSKAYA, O., CANTOR, M., SHERLOCK, G., *et al.*, "Missing value estimation methods for DNA microarrays", *Bioinformatics*, v. 17, n. 6, pp. 520–525, Jun. 2001.

[38] TORGO, L., *Data Mining with R: Learning with Case Studies*. New York, Chapman and Hall/CRC, Jul. 2011.

[39] DIXON, J. K., "Pattern Recognition with Partly Missing Data", *IEEE Transactions on Systems, Man, and Cybernetics*, v. 9, n. 10, pp. 617–621, Oct. 1979. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[40] GUYON, I., ELISSEEFF, A., "An Introduction to Variable and Feature Selection", *The Journal of Machine Learning Research*, , 2003.

[41] LI, J., CHENG, K., WANG, S., *et al.*, "Feature Selection: A Data Perspective", *ACM Comput. Surv.*, v. 50, n. 6, pp. 94:1–94:45, Dec. 2017.

[42] KOHAVI, R., JOHN, G. H., "Wrappers for feature subset selection", *Artificial Intelligence*, v. 97, n. 1, pp. 273–324, Dec. 1997.

[43] GUYON, I., WESTON, J., BARNHILL, S., *et al.*, "Gene Selection for Cancer Classification using Support Vector Machines", *Machine Learning*, v. 46, n. 1, pp. 389–422, Jan. 2002.

[44] LUQUE, A., CARRASCO, A., MARTíN, A., *et al.*, "The impact of class imbalance in classification performance metrics based on the binary confusion matrix", *Pattern Recognition*, v. 91, pp. 216–231, Jul. 2019.

[45] HICKS, S. A., STRüMKE, I., THAMBAWITA, V., *et al.*, "On evaluation metrics for medical applications of artificial intelligence", *Scientific Reports*, v. 12, pp. 5979, Apr. 2022.

[46] NATOLE, M., YING, Y., LYU, S., "Stochastic Proximal Algorithms for AUC Maximization", *35th International Conference on Machine Learning (ICML)*, v. 80, 2018.

[47] SAITO, T., REHMSMEIER, M., "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets", *PLOS ONE*, v. 10, n. 3, pp. e0118432, Mar. 2015.

[48] CatBoost Documentation, "Common Parameters", `https://catboost.ai/docs/en/references/training-parameters/common`, Accessed: 2025-01-14.

[49] KRSTAJIC, D., BUTUROVIC, L. J., LEAHY, D. E., *et al.*, "Cross-validation pitfalls when selecting and assessing regression and classification models", *Journal of Cheminformatics*, v. 6, n. 1, pp. 10, Dec. 2014.

[50] CHADDAD, A., PENG, J., XU, J., *et al.*, "Survey of Explainable AI Techniques in Healthcare", *Sensors*, v. 23, n. 2, pp. 634, Jan. 2023. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[51] RIBEIRO, M. T., SINGH, S., GUESTRIN, C., ""Why Should I Trust You?": Explaining the Predictions of Any Classifier", Aug. 2016, arXiv:1602.04938 [cs, stat].

[52] CREMADES, A., HOYAS, S., VINUESA, R., "Additive-feature-attribution methods: a review on explainable artificial intelligence for fluid dynamics and heat transfer", Sep. 2024, arXiv:2409.11992 [physics].

[53] GALLIC, E., "Machine Learning and Statistical Learning - Explainable machine learning model with SHAP", `https://egallic.fr/Enseignement/M2_EBDS/Course/Week_4/shap.pdf`, Accessed: 2025-01-07.

[54] YOUNG, H. P., "Monotonic solutions of cooperative games", *International Journal of Game Theory*, v. 14, n. 2, pp. 65–72, Jun. 1985.

[55] MOLNAR, C., "Interpretable Machine Learning", `https://christophm.github.io/interpretable-ml-book/`, 2019, `https://christophm.github.io/interpretable-ml-book/`.

[56] KLEINBAUM, D., KLEIN, M., "Logistic Regression: A Self-Learning Text | SpringerLink", 2010.

[57] Google for Developers, "Logistic regression: Calculating a probability with the sigmoid function | Machine Learning", `https://developers.`

google.com/machine-learning/crash-course/logistic-regression/sigmoid-function, Acessed: 2025-03-06.

[58] SHAP Documentation, "An introduction to explainable AI with Shapley values — SHAP latest documentation", https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html, 2025, Accessed: 2025-03-06.

[59] BISCHL, B., BINDER, M., LANG, M., *et al.*, "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges", *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, v. 13, 2021.

[60] MüLLER, A., GUIDO, S., *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2016.

[61] SHAP, "SHAP's GitHub Repsitory", https://github.com/shap/shap, Dec. 2024, Accessed: 2024-12-18.

[62] CHEN, T., GUESTRIN, C., "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, Aug. 2016.

[63] PROKHORENKOVA, L., GUSEV, G., VOROBEV, A., *et al.*, "CatBoost: unbiased boosting with categorical features". In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 6639–6649, Red Hook, NY, USA, Dec. 2018.

[64] CHEN, C., BREIMAN, L., LIAW, A., "Using Random Forest to Learn Imbalanced Data", University of California, Berkeley, 2004.

[65] CAWLEY, G. C., TALBOT, N. L. C., "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation", *The Journal of Machine Learning Research*, v. 11, 2010.

[66] JIA, L., DU, Y., CHU, L., *et al.*, "Prevalence, risk factors, and management of dementia and mild cognitive impairment in adults aged 60 years or older in

China: a cross-sectional study", *The Lancet Public Health*, v. 5, n. 12, pp. e661–e671, Dec. 2020. Publisher: Elsevier.

[67] KIUCHI, S., MATSUYAMA, Y., TAKEUCHI, K., *et al.*, "Number of Teeth and Dementia-free Life Expectancy: A 10-Year Follow-Up Study from the Japan Gerontological Evaluation Study", *Journal of the American Medical Directors Association*, v. 25, n. 11, Nov. 2024. Publisher: Elsevier.

[68] CUI, M., ZHANG, S., LIU, Y., *et al.*, "Grip Strength and the Risk of Cognitive Decline and Dementia: A Systematic Review and Meta-Analysis of Longitudinal Cohort Studies", *Frontiers in Aging Neuroscience*, v. 13, Feb. 2021. Publisher: Frontiers.

[69] PETERMANN-ROCHA, F., LYALL, D. M., GRAY, S. R., *et al.*, "Associations between physical frailty and dementia incidence: a prospective study from UK Biobank", *The Lancet Healthy Longevity*, v. 1, n. 2, pp. e58–e68, Nov. 2020. Publisher: Elsevier.

[70] KUIPER, J. S., ZUIDERSMA, M., OUDE VOSHAAR, R. C., *et al.*, "Social relationships and risk of dementia: A systematic review and meta-analysis of longitudinal cohort studies", *Ageing Research Reviews*, v. 22, pp. 39–57, Jul. 2015.

[71] WANG, H.-X., KARP, A., WINBLAD, B., *et al.*, "Late-Life Engagement in Social and Leisure Activities Is Associated with a Decreased Risk of Dementia: A Longitudinal Study from the Kungsholmen Project", *American Journal of Epidemiology*, v. 155, n. 12, pp. 1081–1087, Jun. 2002.

[72] ANSTEY, K. J., VON SANDEN, C., SALIM, A., *et al.*, "Smoking as a Risk Factor for Dementia and Cognitive Decline: A Meta-Analysis of Prospective Studies", *American Journal of Epidemiology*, v. 166, n. 4, pp. 367–378, Aug. 2007.

Appendix A

Data Description

Table A.1: DataFrame Info with Missing Data

Index	Column	Null Count	Type	Percentage
0	S_Gender	0	Int64	0.00%
1	S_Age	0	Int64	0.00%
2	S_BMI	3	Float64	0.41%
3	S_Social_Education	13	Int64	1.79%
4	S_Social_Religion	20	Int64	2.75%
5	S_Social_ReligiousActivities	9	Int64	1.24%
6	S_Social_VoluntaryAssociation	16	Int64	2.20%
7	S_Social_SocialNetwork	11	Int64	1.52%
8	S_Social_SupportNetwork	17	Int64	2.34%
9	S_Social_Loneliness	25	Int64	3.44%
10	S_E_Lifestyle_Exercise_Light	59	Int64	8.13%
11	S_Lifestyle_AlcoholFrequency	8	Int64	1.10%
12	S_Lifestyle_AlcoholQuantity	47	Int64	6.47%
13	S_Lifestyle_Working65	0	Int64	0.00%
14	S_Lifestyle_Smoking	5	Int64	0.69%
15	S_Lifestyle_Smoking_CigarettesDay_Stoped	26	Int64	3.58%
16	S_Lifestyle_SocialActivities	35	Int64	4.82%
17	S_Lifestyle_PhysicallyDemandingActivities	29	Int64	3.99%

Continued on next page

Index	Column	Null Count	Type	Percentage
18	S_Lifestyle_LeisureActivities	44	Int64	6.06%
19	S_History_NumberMedications	0	Int64	0.00%
20	S_History_FamilyHistory	1	Int64	0.14%
21	S_History_Infarct	8	Int64	1.10%
22	S_History_Arrhythmia	8	Int64	1.10%
23	S_History_HeartFailure	0	Int64	0.00%
24	S_History_Stroke	5	Int64	0.69%
25	S_History_TIARIND	5	Int64	0.69%
26	S_History_Diabetes1	0	Int64	0.00%
27	S_History_Diabetes2	2	Int64	0.28%
28	S_History_ThyroidDisease	4	Int64	0.55%
29	S_History_Cancer	0	Int64	0.00%
30	S_History_Epilepsy	0	Int64	0.00%
31	S_History_AtrialFibrillation	8	Int64	1.10%
32	S_History_IschemicSigns	7	Int64	0.96%
33	S_History_Parkinsons	1	Int64	0.14%
34	S_History_Depression	0	Int64	0.00%
35	S_History_OtherPsychiatricDiseases	3	Int64	0.41%
36	S_History_Snoring	2	Int64	0.28%
37	S_History_SleepApnea	5	Int64	0.69%
38	S_History_HipFracture	4	Int64	0.55%
39	S_History_HeadTrauma	5	Int64	0.69%
40	S_History_DevelopmentalDisabilities	0	Int64	0.00%
41	S_History_HighBloodPressure	9	Int64	1.24%
42	S_BloodTest_HB	3	Int64	0.41%
43	S_BloodTest_CRP	9	Float64	1.24%
44	S_Physiological_Pain	12	Int64	1.65%
45	S_Physiological_HeartRate_Sitting	8	Int64	1.10%
46	S_Physiological_HeartRate_Lying	5	Int64	0.69%
47	S_Physiological_BloodPressure_right	0	Int64	0.00%

Continued on next page

Index	Column	Null Count	Type	Percentage
48	S_Physiological_HandStrengthRight_10s	13	Int64	1.79%
49	S_Physiological_HandStrengthLeft_10s	15	Int64	2.07%
50	S_Physiological_Rise_Safe	23	Int64	3.17%
51	S_Physiological_Rise_How	28	Int64	3.86%
52	S_Physiological_WeightLoss_3months	5	Int64	0.69%
53	S_Physiological_Standing_Right	31	Float64	4.27%
54	S_Physiological_Standing_Left	31	Float64	4.27%
55	S_Physiological_Dental_Prothesis	2	Int64	0.28%
56	S_Physiological_Dental_TeethNumber	33	Int64	4.55%
57	S_Psychological_MemoryLoss	32	Int64	4.41%
58	S_Psychological_MemoryDecline	1	Int64	0.14%
59	S_Psychological_MemoryDecline2	7	Int64	0.96%
60	S_Psychological_AbstractThinking	19	Int64	2.62%
61	S_Psychological_PersonalityChange	7	Int64	0.96%
62	S_Psychological_Identity	2	Int64	0.28%
63	S_SOC_FinalTotalScore	6	Float64	0.83%
64	S_DigitSpan_Correct	3	Int64	0.41%
65	S_DigitSpanBack_Correct	5	Int64	0.69%
66	S_Livingston	4	Int64	0.55%
67	S_EQ5D	24	Int64	3.31%
68	S_Index_Katz	15	Int64	2.07%
69	S_IADL	17	Int64	2.34%
70	S_MMSE	0	Int64	0.00%
71	S_ClockTest_Sum	9	Int64	1.24%
72	S_MCS12	38	Int64	5.23%
73	S_PCS12	38	Int64	5.23%
74	S_CPRS	2	Int64	0.28%
75	Doc_2010_2013	0	Int64	0.00%

Appendix B

Code Implementation

```
1
2     _parameter_constraints: dict = {
3         **_BaseImputer._parameter_constraints,
4         "n_neighbors": [Interval(Integral, 1, None, closed="left")
    ],
5         "weights": [StrOptions({"uniform", "distance"}), callable,
    Hidden(None)],
6         "metric": [StrOptions(set(_NAN_METRICS)), callable],
7         "copy": ["boolean"],
8         "imputation_strategy": [StrOptions({"mean", "median"}),
    callable, Hidden(None)]
9     }
10
11 def _calc_impute(self, dist_pot_donors, n_neighbors, fit_X_col,
    mask_fit_X_col):
12         """Helper function to impute a single column."""
13         # Get donors (indices of n_neighbors closest donors)
14         donors_idx = np.argpartition(dist_pot_donors, n_neighbors -
     1, axis=1)[:, :n_neighbors]
15
16         # Debugging: Print selected donor indices
17         print("Selected donor indices (n_neighbors):")
18         print(donors_idx)
19
20         # Get weight matrix from distance matrix
21         donors_dist = dist_pot_donors[np.arange(donors_idx.shape
    [0])[:, None], donors_idx]
```

```
22
23          # Debugging: Print distances of the selected donors
24          print("\nDistances of selected donors:")
25          print(donors_dist)
26
27          weight_matrix = _get_weights(donors_dist, self.weights)
28
29          # fill nans with zeros in weight matrix
30          if weight_matrix is not None:
31              weight_matrix[np.isnan(weight_matrix)] = 0.0
32
33          # Retrieve donor values based on the selected donor indices
34          donors = fit_X_col.take(donors_idx)
35          donors_mask = mask_fit_X_col.take(donors_idx)
36          donors = np.ma.array(donors, mask=donors_mask)
37
38          # Debugging: Print the actual donor values for the selected
     indices
39          print("\nDonor values for the selected neighbors:")
40          print(donors)
41
42          # Apply the selected imputation strategy (mean or median)
43          if self.imputation_strategy == 'mean':
44              return np.ma.average(donors, axis=1, weights=
    weight_matrix).data
45          elif self.imputation_strategy == 'median':
46              return np.ma.median(donors, axis=1).data
47          else:
48              raise ValueError(f"Invalid imputation strategy: {self.
    imputation_strategy}")
49
50          def process_chunk(dist_chunk, start):
51              row_missing_chunk = row_missing_idx[start : start + len
    (dist_chunk)]
52
53              # Find and impute missing by column
54              for col in range(X.shape[1]):
55                  # print(f"\n\nColumn ID: {col}\n")
56                  if not valid_mask[col]:
```

```
                    # column was all missing during training
                    continue

                col_mask = mask[row_missing_chunk, col]
                if not np.any(col_mask):
                    # column has no missing values
                    continue

                (potential_donors_idx,) = np.nonzero(
non_missing_fix_X[:, col])

                # receivers_idx are indices in X
                receivers_idx = row_missing_chunk[np.flatnonzero(
col_mask)]

                # distances for samples that needed imputation for
column
                dist_subset = dist_chunk[dist_idx_map[receivers_idx
] - start][
                    :, potential_donors_idx
                ]

                # receivers with all nan distances impute with mean
 or median
                all_nan_dist_mask = np.isnan(dist_subset).all(axis
=1)
                all_nan_receivers_idx = receivers_idx[
all_nan_dist_mask]

                if all_nan_receivers_idx.size:
                    if self.imputation_strategy == "mean":
                        col_stat = np.ma.array(
                            self._fit_X[:, col], mask=mask_fit_X[:,
 col]
                        ).mean()
                    elif self.imputation_strategy == "median":
                        col_stat = np.ma.array(
                            self._fit_X[:, col], mask=mask_fit_X[:,
 col]
```

```
                        ).median()
                    else:
                        raise ValueError(f"Invalid imputation strategy: {self.imputation_strategy}")

                    X[all_nan_receivers_idx, col] = col_stat

                    if len(all_nan_receivers_idx) == len(receivers_idx):
                        # all receivers imputed with mean or median
                        continue

                    # receivers with at least one defined distance
                    receivers_idx = receivers_idx[~all_nan_dist_mask]
                    dist_subset = dist_chunk[dist_idx_map[receivers_idx] - start][
                        :, potential_donors_idx
                    ]

                n_neighbors = min(self.n_neighbors, len(potential_donors_idx))
                value = self._calc_impute(
                    dist_subset,
                    n_neighbors,
                    self._fit_X[potential_donors_idx, col],
                    mask_fit_X[potential_donors_idx, col],
                )
                X[receivers_idx, col] = value
```

Code B.1: Modified Sklearn KNN Imputer

```
def run_rfe(X, y, clf, params, step=1, n_features_to_select=15, default_params=False, verbose=True):

    logger.info(f'{clf.__name__} - Starting RFE')
    if default_params:
        if clf.__name__ == "CatBoostClassifier":
            print("CBC")
            model = clf(silent= True, random_seed= 3577)
```

```
        elif clf.__name__ == "BalancedRandomForestClassifier":
            print("BRFC")
            model = clf(**{'bootstrap': False,
                           'class_weight': 'balanced',
                           'random_state': 3577,
                           'replacement': False})
        else:
            print("XGB")
            model = clf(seed = 3577)
    else:
        model = clf(**params)

    rfe = RFE(estimator=model, n_features_to_select=
   n_features_to_select, step=step, verbose=verbose)
    rfe.fit(X, y)
    selected_cols = X.iloc[:, rfe.support_].columns
    logger.info(f'Number of features: {rfe.n_features_}')
    logger.info(f'Selected columns: {selected_cols}')

    return selected_cols


# Storing best columns per inner fold to reduce computational costs
   .
rfe_cv = StratifiedKFold(n_splits=5, random_state=3577, shuffle=
   True)

# RFECV selected columns for each inner fold of an outer fold.
cols_per_fold = {k: [] for k in range(rfe_cv.get_n_splits())}

for outer_id, (training_samples, test_samples) in enumerate(
   outer_cv.split(X, y)):
    for inner_id,(inner_train, inner_test) in enumerate(inner_cv.
   split(X[training_samples], y[training_samples])):

        cols = run_rfe(X_features.iloc[inner_train],
                       y[inner_train],
                       clf,
                       params={}, # Base parameters,
```

```
                                step=1,
                                n_features_to_select=15,
                                default_params=True,
                                verbose=False )

        cols_per_fold[outer_id].append(cols)
```

Code B.2: Optimized Recursive Feature Elimination

```
def nested_cv(X, y, inner_cv, outer_cv, clf, parameter_grid,
    verbose=True):

# X dataframe with feature names
X_features = X.copy()

# X and y numpy arrays
X = X.to_numpy()
y = y.to_numpy().ravel()

# Model name string
model_name = clf.__name__

# Initialize lists to store the outer and inner CV metrics
outer_metrics = {
    'Accuracy': [], 'ROC AUC': [], 'PR AUC': [], 'Sensitivity': [],
    'Specificity': [],
    'Precision': [], 'F1': [], 'Balanced Accuracy': [], 'MCC': [],
    'Kappa': [], 'G-Mean': [],
    'Confusion Matrix': [], 'HP':[], 'Features':[]
}
inner_metrics = {
    'Accuracy': [], 'ROC AUC': [],'PR AUC': [], 'Sensitivity': [],
    'Specificity': [],
    'Precision': [], 'F1': [], 'Balanced Accuracy': [], 'MCC': [],
    'Kappa': [], 'G-Mean': [],
    'Confusion Matrix': [], 'HP':[], 'Features':[]
}

# Verbose information
counter = 1
```

```
total_combinations = len(parameter_grid)*outer_cv.n_splits*inner_cv.n_splits
logger.info(f'{model_name} - Total Combinations: {total_combinations}')


# For each split of the data in the outer cross-validation
for outer_id, (training_samples, test_samples) in enumerate(outer_cv.split(X, y)):

    best_params = {}
    best_params_per_fold = []
    best_score = -np.inf
    best_inner_metrics = {k: [] for k in inner_metrics.keys()}

    parameter_combinations = len(parameter_grid)

    # Inner cross-validation to select the best parameters
    for parameter_id, parameters in enumerate(parameter_grid):

        # Cross-validation scores list. Will be used when choosing the best model.
        cv_scores = []
        current_inner_metrics = {k: [] for k in inner_metrics.keys()}

        # Previous GridSearch metric result
        previous_result = 0
        best_inner_fold = -1



        for inner_id,(inner_train, inner_test) in enumerate(inner_cv.split(X[training_samples], y[training_samples])):
            if verbose:
                logger.info(f'{model_name} - OF {outer_id + 1}/{outer_cv.n_splits}  IF {inner_id + 1}/{inner_cv.n_splits}  HP {parameter_id + 1}/{parameter_combinations} COUNTER: {counter}/{total_combinations}')
```

```
                # print(f"OF {outer_id + 1}/{outer_cv.n_splits}  IF {inner_id + 1}/{inner_cv.n_splits}  HP {parameter_id + 1}/{parameter_combinations} TOTAL Counter: {counter}/{total_combinations}")
                # print(f"Inner fold {inner_id + 1}")
            counter+=1

            # Retrieving the RFE selected columns for the given (outer, inner) folds pair.
            cols = cols_per_fold[outer_id][inner_id]

            # Threshold tuning
            if model_name == "BalancedRandomForestClassifier":
                threshold = 0.65
            else:
                threshold = 0.5

            clf_ = clf(**parameters)
            clf_.fit(X_features.iloc[inner_train][cols], y[inner_train])

            # Get predicted probabilities for the positive class
            y_pred_prob_inner = clf_.predict_proba(X_features.iloc[inner_test][cols])[:, 1]

            # Apply threshold to decide positive (1) or negative (0) predictions
            y_pred_inner = (y_pred_prob_inner >= threshold).astype(int)

            # Calculate various metrics for inner fold
            current_inner_metrics['Accuracy'].append(accuracy_score(y[inner_test], y_pred_inner))
            current_inner_metrics['ROC AUC'].append(roc_auc_score(y[inner_test], y_pred_prob_inner))
            current_inner_metrics['Sensitivity'].append(recall_score(y[inner_test], y_pred_inner))
            current_inner_metrics['Precision'].append(precision_score(y[inner_test], y_pred_inner))
```

```
            current_inner_metrics['F1'].append(f1_score(y[
inner_test], y_pred_inner))
            current_inner_metrics['Balanced Accuracy'].append(
balanced_accuracy_score(y[inner_test], y_pred_inner))
            current_inner_metrics['MCC'].append(matthews_corrcoef(y
[inner_test], y_pred_inner))
            current_inner_metrics['Kappa'].append(cohen_kappa_score
(y[inner_test], y_pred_inner))
            [tn_inner, fp_inner, fn_inner, tp_inner] =
confusion_matrix(y[inner_test], y_pred_inner).ravel()
            specificity_inner = tn_inner / (tn_inner + fp_inner)
            current_inner_metrics['Confusion Matrix'].append(list(
confusion_matrix(y[inner_test], y_pred_inner).ravel()))
            current_inner_metrics['Specificity'].append(
specificity_inner)
            g_mean_inner = np.sqrt(recall_score(y[inner_test],
y_pred_inner) * specificity_inner)
            current_inner_metrics['G-Mean'].append(g_mean_inner)
            precision, recall, _ = precision_recall_curve(y[
inner_test], y_pred_prob_inner)
            current_inner_metrics['PR AUC'].append(auc(recall,
precision))
            current_inner_metrics['Features'].append(list(cols))

            # Calculate the recall (sensitivity)
            score = recall_score(y[inner_test], y_pred_inner)


            pr_auc = auc(recall, precision)
            roc_auc = roc_auc_score(y[inner_test],
y_pred_prob_inner)
            precision = precision_score(y[inner_test], y_pred_inner
)
            recall = recall_score(y[inner_test], y_pred_inner)

            logger.debug(f"INNER ({inner_id}) SCORE: {score}")

            # Check score to decide if the current is the best fold
.
```

```
110             if score > previous_result:
111                 best_inner_fold = inner_id
112 
113             cv_scores.append(score)
114 
115         mean_score = np.mean(cv_scores)
116 
117         # Store the metrics for the inner fold if they are better
    than the previous ones.
118         if mean_score > best_score:
119 
120             best_score = mean_score
121             best_params = parameters # Parameters for this round
122             best_inner_metrics = current_inner_metrics
123             best_inner_metrics['HP'] = best_params
124             best_inner_metrics['Features'] = current_inner_metrics[
    'Features'][best_inner_fold]
125 
126         best_params_per_fold.append(best_params)
127 
128     # Store the best inner metrics for this outer fold except the
    confusion matrix and the Hyperparameters
129     for metric in list(inner_metrics.keys())[:-3]:
130         inner_metrics[metric].append(np.mean(best_inner_metrics[
    metric]))
131 
132     # The internal result will be given by the mean of the inner
    metrics for each outer fold,
133     # but the confusion matrix will be the separate confusion
    matrices for each inner fold of
134     # the best parameters.
135     inner_metrics['Confusion Matrix'].append(best_inner_metrics['
    Confusion Matrix'])
136     inner_metrics['HP'].append(best_inner_metrics['HP'])
137     inner_metrics['Features'].append(best_inner_metrics['Features'
    ])
138 
139     logger.debug(f"{model_name} - Best metrics from inner cv found"
    )
```

```
140
141     # Build classifier on the outer training set with best
        parameters
142     clf_ = clf(**best_params)
143
144     # Fit the classifier for the outer training split with the
        selected features from the RFE
145     clf_.fit(X_features.iloc[training_samples][list(
        best_inner_metrics['Features'])], y[training_samples])
146
147     logger.debug(f"{model_name} - Outer fold trained with best
        metrics from inner cv")
148
149     y_pred_prob = clf_.predict_proba(X_features.iloc[test_samples][
        list(best_inner_metrics['Features'])])[:, 1]
150
151     y_pred = (y_pred_prob >= threshold).astype(int)
152
153     logger.debug(f"{model_name} - Outer fold predictions")
154
155     # Calculate various metrics for outer fold
156     outer_metrics['Accuracy'].append(accuracy_score(y[test_samples
        ], y_pred))
157     outer_metrics['ROC AUC'].append(roc_auc_score(y[test_samples],
        y_pred_prob))
158     outer_metrics['Sensitivity'].append(recall_score(y[test_samples
        ], y_pred))
159     outer_metrics['Precision'].append(precision_score(y[
        test_samples], y_pred))
160     outer_metrics['F1'].append(f1_score(y[test_samples], y_pred))
161     outer_metrics['Balanced Accuracy'].append(
        balanced_accuracy_score(y[test_samples], y_pred))
162     outer_metrics['MCC'].append(matthews_corrcoef(y[test_samples],
        y_pred))
163     outer_metrics['Kappa'].append(cohen_kappa_score(y[test_samples
        ], y_pred))
164
165     [tn, fp, fn, tp] = confusion_matrix(y[test_samples], y_pred).
        ravel()
```

```
166     specificity = tn / (tn + fp)
167     outer_metrics['Confusion Matrix'].append([tn, fp, fn, tp])
168     outer_metrics['Specificity'].append(specificity)
169     g_mean = np.sqrt(recall_score(y[test_samples], y_pred) *
        specificity)
170     outer_metrics['G-Mean'].append(g_mean)
171     outer_metrics['HP'].append(best_params)
172     outer_metrics['Features'].append(list(best_inner_metrics['
        Features']))
173     precision, recall, _ = precision_recall_curve(y[test_samples],
        y_pred_prob)
174     outer_metrics["PR AUC"].append(auc(recall, precision))
175     logger.debug(f"{model_name} - Outer fold metrics calculated")
176
177 # Return both inner and outer metrics as dictionaries
178 return {'Outer': outer_metrics, 'Inner': inner_metrics}
```

Code B.3: Optimized Recursive Feature Elimination

```
1  # Code adapted from SHAP's original documentation
2  # https://shap.readthedocs.io/en/latest/example_notebooks/
       tabular_examples/model_agnostic/Simple%20Kernel%20SHAP.html
3
4  import numpy as np
5  import itertools
6  import scipy.special
7  import pandas as pd
8
9  # Define powerset
10 def powerset(iterable):
11     s = list(iterable)
12     return list(itertools.chain.from_iterable(itertools.
       combinations(s, r) for r in range(len(s)+1)))
13
14 # Define Shapley kernel
15 def shapley_kernel(M, s):
16     if s == 0 or s == M:
17         return 10000
18     return (M - 1) / (scipy.special.comb(M, s) * s * (M - s))
19
```

```
# Model outputs
subset_values = {
    (): 5,
    ('A',): 8,
    ('B',): 6,
    ('C',): 7,
    ('A', 'B'): 10,
    ('A', 'C'): 12,
    ('B', 'C'): 11,
    ('A', 'B', 'C'): 15
}

# Feature setup
features = ['A', 'B', 'C']
M = len(features)
num_subsets = 2 ** M

# Build X and y
X = np.zeros((num_subsets, M + 1))  # includes intercept
X[:, -1] = 1  # base value term
y = np.zeros(num_subsets)
weights = np.zeros(num_subsets)

all_subsets = powerset(range(M))
for i, subset in enumerate(all_subsets):
    row = [0] * M
    for j in subset:
        row[j] = 1
    X[i, :-1] = row
    y[i] = subset_values[tuple(features[j] for j in subset)]
    weights[i] = shapley_kernel(M, len(subset))

# Weighted least squares
D = np.diag(np.sqrt(weights))
X_w = D @ X
y_w = D @ y
phi = np.linalg.pinv(X_w.T @ X_w) @ X_w.T @ y_w  # Pseudoinverse in
    case of non-invertible matrices
# Results
```

```
shap_values = phi[:-1]
base_value = phi[-1]
total = np.sum(phi)
f_x = subset_values[('A', 'B', 'C')]

# Output
df = pd.DataFrame({
    "Feature": features + ["Base Value", "Sum of SHAP Values", "f(x) from table"],
    "Value": np.append(shap_values, [base_value, total, f_x])
})
print(df)
```

Code B.4: Simple Kernel SHAP implementation, adapted from SHAP's original documentation

Appendix C

Selected Hyperparameters and Features

C.1 XGBoost

Parameter	Value
colsample_bytree	0.8
eval_metric	aucpr
gamma	0
learning_rate	0.01
max_depth	2
min_child_weight	2
n_estimators	150
scale_pos_weight	12
seed	3577
subsample	0.8

Table C.1: Hyperparameter configuration for the XGBoost model.

Selected Features
S_Age
S_Social_Education
S_Lifestyle_Working65
S_Lifestyle_Smoking_CigarettesDay_Stoped
S_History_NumberMedications
S_History_Infarct
S_History_Cancer
S_Physiological_HandStrengthLeft_10s
S_Physiological_Standing_Left
S_Physiological_Dental_TeethNumber
S_Psychological_MemoryLoss
S_Psychological_MemoryDecline2
S_Psychological_PersonalityChange
S_EQ5D
S_PCS12

Table C.2: Selected features for the XGBoost model.

C.2 CatBoost

Parameter	Value
auto_class_weights	Balanced
depth	4
eval_metric	AUC
iterations	100
l2_leaf_reg	5
learning_rate	0.01
random_seed	3577
silent	True

Table C.3: Hyperparameter configuration for the CatBoost model.

Selected Features
S_Age
S_BMI
S_Social_VoluntaryAssociation
S_Lifestyle_AlcoholQuantity
S_Lifestyle_Smoking_CigarettesDay_Stoped
S_Lifestyle_SocialActivities
S_History_NumberMedications
S_BloodTest_HB
S_Physiological_HandStrengthRight_10s
S_Physiological_HandStrengthLeft_10s
S_Physiological_Standing_Left
S_Physiological_Dental_Prothesis
S_Physiological_Dental_TeethNumber
S_Psychological_MemoryDecline2
S_ClockTest_Sum

Table C.4: Selected features for the CatBoost model.

C.3 Balanced Random Forest

Parameter	Value
bootstrap	False
class_weight	balanced
max_depth	5
min_samples_leaf	3
min_samples_split	2
n_estimators	200
random_state	3577
replacement	True
sampling_strategy	not minority

Table C.5: Hyperparameter configuration for the Balanced Random Forest model.

Selected Features
S_Age
S_BMI
S_History_NumberMedications
S_BloodTest_HB
S_Physiological_HeartRate_Sitting
S_Physiological_HeartRate_Lying
S_Physiological_HandStrengthRight_10s
S_Physiological_HandStrengthLeft_10s
S_Physiological_Standing_Right
S_Physiological_Standing_Left
S_Physiological_Dental_TeethNumber
S_Psychological_MemoryLoss
S_SOC_FinalTotalScore
S_DigitSpan_Correct
S_MMSE

Table C.6: Selected features for the Balanced Random Forest model.