Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Magistrale in Informatica

AI-based Models for Cognitive Impairment Detection in Cancer Treatment: Analysis of Preclinical Data within the AI4ChemoBrain Project

Relatore:

Chiar.mo Prof.
DANILO MONTESI

Controrelatore:

Chiar.mo Prof.
MAURIZIO GABBRIELLI

Correlatore:

Prof.
FLAVIO BERTINI

Presentata da:

DELIA CAVALCA

Sessione I
Anno Accademico 2024/2025

Sommario

Negli ultimi decenni, i progressi nella chemioterapia hanno contribuito ad aumentare la sopravvivenza dei pazienti oncologici, ponendo tuttavia l'attenzione su una nuova priorità clinica: il miglioramento della qualità della vita nei sopravvissuti, spesso compromessa da effetti collaterali persistenti. Tra questi, fino al 70% dei pazienti segnala difficoltà cognitive, note come *chemobrain*, che includono deficit di concentrazione, ragionamento, apprendimento e memoria, rendendo difficile lo svolgimento delle attività quotidiane. Nonostante la rilevanza del fenomeno, attualmente non esistono strategie efficaci per la sua prevenzione o trattamento.

In questo contesto, il progetto AI4ChemoBrain intende sviluppare un modello predittivo di rischio individuale dell'insorgenza di *chemobrain*, attraverso l'impiego di tecnologie di apprendimento automatico e intelligenza artificiale.

Il presente studio si inserisce nella fase iniziale del progetto, dedicata all'analisi di un dataset storico contenente dati comportamentali ottenuti da test cognitivi effettuati su modelli murini. L'obiettivo è lo sviluppo di un classificatore in grado di prevedere la presenza di deterioramento cognitivo nei soggetti. A tal fine, sono stati addestrati e valutati diversi algoritmi supervisionati di machine learning, che hanno portato all'identificazione di modelli con un'accuratezza superiore al 95%. Tali sistemi costituiscono uno strumento affidabile per l'individuazione dei soggetti con deficit cognitivi, risultando utili per le fasi successive del progetto.

È stata inoltre condotta un'analisi esplorativa mediante algoritmi non supervisionati, volta all'identificazione di eventuali fattori predittivi alternativi. Tuttavia, i risultati non hanno evidenziato nei dati caratteristiche coerenti con la presenza di disfunzioni cognitive, in linea con quanto rilevato dai modelli di classificazione supervisionata.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This research is part of the AI4ChemoBrain project, which aims to develop a machine learning-based tool to predict cognitive impairment in cancer patients undergoing chemotherapy. The focus of this study is on the analysis of historical preclinical data to create supervised and unsupervised machine learning models to detect cognitive impairment and identify potential novel predictive factors.

1.1 Motivation

Cognitive impairment is a debilitating side effect experienced by patients with cancer treated with systemically administered anticancer therapies. With around 19.3 million new cases of cancer worldwide in 2020 and the five year survival rate growing from 50% in 1970 to 67% in 2013, an urgent need exists to understand enduring side effects with severe implications for quality of life. Among these, cognitive impairment is one of the most challenging, as it severely affects daily functioning and general well-being of individuals.

Often referred to as "chemobrain" or "chemofog", cognitive impairment associated with cancer treatment is characterized by a decline in performance in cognitive function related to learning, attention, executive functions, memory, multitasking, and processing speed, and is associated with chemotherapy, hormone therapy, immunotherapy, and targeted therapies. Prevalence of clinically significant cognitive impairment varies between

17% and 78% with self-reported measures, and is approximately 33% using objective neurocognitive testing in post-chemotherapy patients with breast cancer. Although cognitive impairment is a key factor in preventing patients from regaining their previous quality of life, no management strategies or clinical guidelines are available [1].

In this context, the AI4ChemoBrain project aims to generate a tool for clinical use to predict the onset of chemobrain based on machine learning (ML) and artificial intelligence (AI) technologies. The objective of the project is the development of a predictive demonstrator for cognitive complications during chemotherapy based on subject-derived phenotype and omics data.

The model will exploit the tools of machine learning and artificial intelligence and will be trained with historical datasets collected over the last 15 years from preclinical models of cognitive decline and omics data, already validated through statistical analysis and scientific publication. The ML/AI model will be tested with a dataset derived from a preclinical chemobrain model, in order to assess the predictive ability of the chemobrain. The project will then validate the predictive model with internal and external cohorts to ensure its robustness, effectiveness, and translatability in the clinical environment. The ultimate goal is the transfer of the model to the clinical setting to promote adjuvant therapies and support personalised medicine, helping to improve the empowerment of cancer patients and reduce the economic burden of treatment side effects.

The AI4ChemoBrain project is co-funded by the Emilia-Romagna ERDF Regional Programme (ERDF RP) 2021-2027, ACTION 1.1.2 [2].

1.2 Research objectives

Within the AI4ChemoBrain project, which is still in its early stages, my contribution focuses on the analysis of historical data related to preclinical models of cognitive decline. These data, provided by the lead partner of the project IRET Foundation, consist of behavioural measures from murine models collected through well-established cognitive tests.

The primary objective of the analysis is to develop a supervised machine learning model capable of detecting cognitive impairment, based on the validated historical data-

set. This model will serve as a reliable tool for classifying samples in new datasets generated during the later stages of the project, providing a solid foundation for the assessment of cognitive decline in subsequent phases. To achieve this goal, various ML algorithms are trained and evaluated, in order to identify the most accurate model. Also, an analysis is performed to understand the impact of specific characteristics in predicting cognitive impairment, finding the most influential ones.

In addition, an exploratory unsupervised approach is used to investigate potential alternative predictive factors of cognitive impairment. By applying clustering techniques, the goal is to uncover novel patterns or factors that may not be immediately evident from traditional predictive models.

1.3 Thesis structure

The thesis is structured into the following chapters:

- Chapter 1: This chapter introduces the motivations and context in which the study was conducted and outlines the main goals of the research.

- Chapter 2: This chapter reviews the state of the art, defines key concepts, and provides the necessary definitions to understand the framework of the research, focusing on cognitive impairment in murine models and the behavioural tests used to assess it.

- Chapter 3: This chapter provides a detailed description of the dataset used in the study, including its origin, structure, and integration process. It also presents an exploratory data analysis to examine attribute types, distributions, and key patterns through statistical and visualization techniques.

- Chapter 4: This chapter describes the data preprocessing phase, a critical step in obtaining high-quality data. It presents the methods used to clean, normalize, and transform the raw data into a suitable format for analysis.

- Chapter 5: This chapter provides an overview of machine learning models, describing concepts and mechanisms of the supervised and unsupervised techniques used.

- Chapter 6: This chapter gives a description of the modelling phase for the classification task of the project. It reports the results of the trained models and provides a comparison between them. Performance evaluation and interpretation of the classifiers are also provided for a complete understanding of the results obtained.

- Chapter 7: This chapter describes the unsupervised analysis objectives and the results obtained from the clustering techniques used. It gives an interpretation of the models trained, exploring the correlation between the identified clusters and the true labels associated with the samples.

- Chapter 8: This chapter provides a comprehensive understanding of the results obtained by both the supervised and unsupervised approaches, within the specific context of the project. It draws conclusions from the study and outlines future research directions.

Chapter 2

State of the Art

Animal models are essential for studying both normal and disordered cognitive processes. In particular, murine models provide a fundamental framework for evaluating cognitive performance, offering valuable insights into learning and memory deficits. Behavioural tests serve as key tools for the assessment of cognitive impairment.

2.1 Cognition in murine models

Cognition is the mental action or process of acquiring knowledge and understanding through thought, experience, and the senses. It encompasses all aspects of intellectual functions and processes such as perception, attention, learning, memory, comprehension and production of language, intelligence, and reasoning [3].

Specifically, memory is defined as the ability to store, maintain, and retrieve information, while learning is the acquisition of information that changes behaviour and memory. In addition, memory is classified according to different criteria, including duration (short-term memory and long-term memory), function (working memory and reference memory), and content (explicit memory and conceptual memory) [4].

The task of understanding deficits in memory and learning in humans is daunting due to the complexity of neural and cognitive mechanisms in the nervous system. This job is made more difficult for clinicians and researchers by the fact that many techniques used to research memory are not ethically acceptable or technically feasible for use in

humans [4]. However, humans and animals exhibit significant similarities in cognitive abilities, such as episodic memory and spatial orientation skills. In particular, murine models (mice and rats) share numerous neurological and brain dynamics similarities with humans. For instance, in both humans and rodents, the ability to orient toward specific reference points and to create cognitive spatial maps is essential for survival and is mediated by common brain structures, such as the hippocampus [5]. These shared mechanisms make murine models a valuable tool for exploring the fundamental processes of cognition and its dysfunctions, such as deficits in memory and learning.

Therefore, to improve the study of human pathological conditions, simulations are conducted using animal models. Cognitive decline can be induced in rodents through various methodologies, including brain lesions, genetic manipulations, or pharmacological treatments.

Among the various models, the **Tg2576** mouse is widely used to investigate learning and memory. It is a transgenic model for the study of **Alzheimer's disease** (AD), a neurodegenerative disorder characterized by memory loss and personality changes, leading to dementia.

The Tg2576 model expresses a mutation in the APP (Amyloid Precursor Protein) gene, leading to the formation of amyloid plaques in the brain, one of the key pathological features of AD. Tg2576 mice develop behavioural manifestations similar to those observed in humans affected by Alzheimer's disease, such as deficits in memory and learning. These animals exhibit difficulties in navigating complex environments, mirroring episodic memory loss and impaired executive functions seen in patients with senile dementia. They present an impairment of short-term memory at about 9 months of age, while long-term memory shows a marked deterioration around 12 months [6]. These characteristics make Tg2576 mice an ideal model for investigating the mechanisms underlying cognitive impairment, as well as for testing potential pharmacological therapies.

2.2 Cognitive impairment assessment

Cognitive impairment is assessed through behavioural tests that evaluate specific cognitive abilities, such as memory, spatial learning, attention, and problem-solving skills.

Over the decades, different tests have been developed to analyse learning and memory disorders in animals. Among the most used are the Morris Water Maze, Novel Object Recognition, Y-Maze, Open Field, Touch-Screen Systems, and Contextual Fear Conditioning. Each of these tests explores different brain domains, requires different technological support, and has its own strengths and limitations. Three of these tests used in the study of cognition in laboratory rodents are described below.

2.2.1 Morris Water Maze

The Morris Water Maze (MWM) is a spatial learning test for rodents, developed by the neuroscientist Richard Morris. Spatial learning refers to the animal's ability to learn the location of a reward. The MWM assesses the ability to learn to find a hidden platform in a swimming arena, using distal cues.

The test procedure involves placing the animal in a circular pool filled with water and evaluating the strategy it adopts to reach a target area, a platform positioned at a specific point in the pool. The subject must learn to use distal cues (i.e., visual landmarks arranged around the pool) to navigate a direct path to the platform, which is made invisible (submerged under the water's surface). The animal is given a series of daily trials, starting from different positions around the perimeter of the pool. With repeated trials, an improvement in the ability to quickly locate the platform is observed, demonstrating the acquisition of familiarity with the environment and spatial learning. At the end of the learning phase, a final trial (probe trial) is conducted, during which the platform is removed from the pool, and the animal's search in what was the target area is evaluated [7] [8].

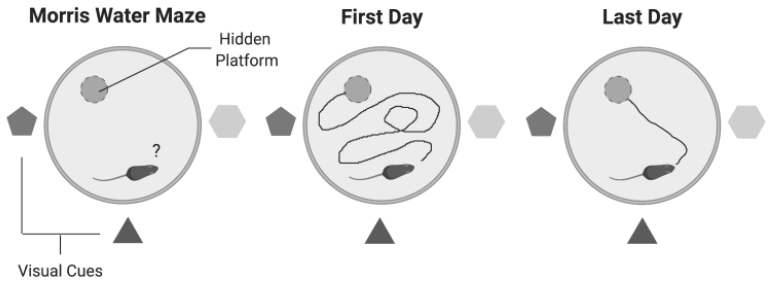


Figure 2.1: MWM test: setup and learning progression.

Subjects with cognitive impairments struggle to perform the task, as they fail to remember the platform's location and consequently exhibit inefficient navigation within the arena.

The main measurements of the MWM are: Path Length, Latency (swim time: the time from the start of the test to the identification of the platform), and Swim Speed.

The MWM method offers several advantages for assessing cognitive function in rodents: (1) It does not require pre-training and can be completed quickly with a small number of animals. (2) It allows analysis of learning and memory retrieval through "training" and "probe" trials. (3) It eliminates confounding olfactory cues. (4) It helps distinguish non-mnemonic behaviours and identify motor or motivational deficits. (5) It enables learning and relearning experiments, allowing multiple drug doses to be tested on the same group. (6) It avoids more aversive procedures like food deprivation or electric shock. (7) It is cost-effective, easy to set up, and simple to use [9].

However, the MWM test also has some limitations that could affect the analysis of the experiment: (1) It may cause stress or hypothermia in the animals, which can influence spatial learning. (2) The influence of non-cognitive factors on performance. (3) The impact of the swimming strategy adopted by the animal on performance. (4) It is less sensitive in evaluating working memory than other methods [10].

2.2.2 Contextual Fear Conditioning

The Contextual Fear Conditioning (CFC) is a test that evaluates fear conditioning, useful for analysing the mechanisms of learning and memory in animals and exploring the behavioural response to the perception of danger. The idea behind fear conditioning is that a fearful experience establishes an emotional memory that can result in long-term behavioural changes and, in some cases, these changes can become part of the permanent behavioural repertoire of the individual.

In this paradigm, mice are conditioned by pairing a tone (conditioned stimulus, CS) with foot shock (unconditioned aversive stimulus, US). The animals are later examined in two ways: one evaluates contextual fear and the other examines fear responses elicited by

the CS (cued fear conditioning). Fear in both tests is signified by immobility or freezing behaviour. This response is a direct reflection of the conditioned emotional response in animals [7].

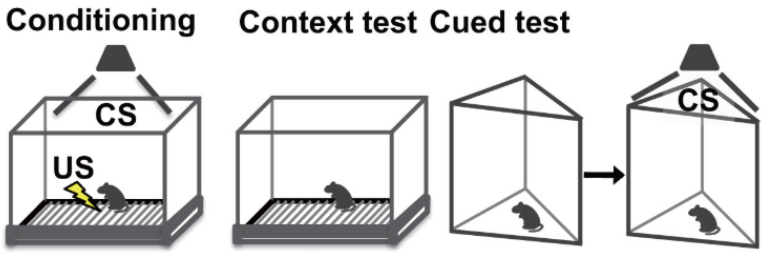


Figure 2.2: CFC test: schematic representation of the protocol.

The main parameter assessed in CFC is the % of Freezing (the duration of immobilization of the subject relative to the total duration of the test).

The strengths of the fear conditioning paradigm are multiple: (1) Conditioning requires only a single session. (2) The stimuli are under direct control of the investigator. (3) The behavioural responses have been operationally defined, validated, and are simple to measure [7].

During testing, specific precautions must be taken to ensure that the observed emotional responses are due to conditioning and not to other factors: (1) Since odours can influence rodent behaviour and induce freezing, the testing room must be properly cleaned between experiments using odourless cleaning products. (2) Animals should be housed in a group before testing, as isolation can cause abnormal behaviours and interfere with attention and learning. (3) Test subjects should be kept in a different room from the testing area to prevent them from hearing the stimulus before the test [7].

2.2.3 Y-Maze

The Y-Maze test (YM) is used to measure short-term spatial memory in murine models. It is based on rodents' intrinsic curiosity to explore new environments while there are no positive or negative stimuli in the maze.

This test uses a Y-shaped maze consisting of three arms of equal size and equidistant from each other, identified as A, B, and C. The animal is initially placed in arm A and allowed to freely explore the maze for a limited period (typically 8 minutes). Exploration is considered optimal when the subject visits all three arms in sequence without immediate repetitions, demonstrating the ability to remember and continuously update information about the last arm explored. In contrast, memory impairment disrupts this ability, leading to a lower percentage of spontaneous alternations. Therefore, a high percentage of alternation is indicative of good short-term spatial memory, where alternation is defined as a sequence of consecutive entries into three different arms [4] [11] [12].

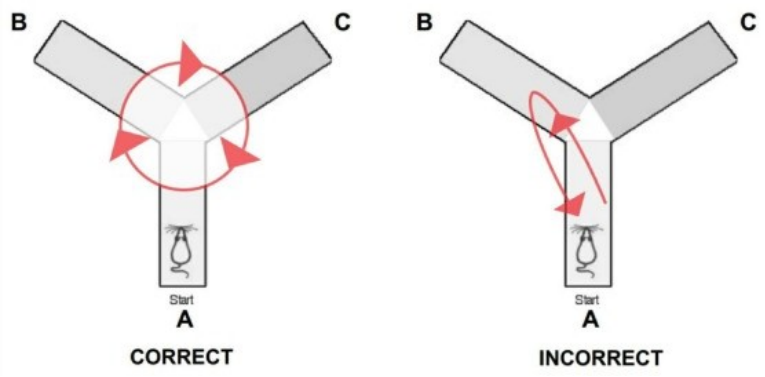


Figure 2.3: YM test: illustration of examples of a correct and an incorrect alternation.

The main parameters considered for YM analysis are:

- **N° Entries Tot** (the total number of entries into the arms)
- **Visited Arms** (the sequence of visited arms: A, B, C)
- **N° of Alternations** (the number of alternations)
  An alternation is defined as a sequence of consecutive entries into three different arms. For example, if the sequence of arm entries is ABCABC, the alternations are 4: ABC, BCA, CAB, ABC. In the Y-maze, there are six possible alternation sequences (clockwise: ABC, BCA, CAB; anti-clockwise: ACB, CBA, BAC).
- **% of Correct Alternations** (percentage of correct alternations)

The percentage of correct alternations is calculated as follows:

$$\% \text{ of Correct Alternations} = \left(\frac{\text{N}^{\text{o}} \text{ of Alternations}}{\text{N}^{\text{o}} \text{ Entries Tot} - 2} \right) \times 100 \tag{2.1}$$

One of the main advantages of the Y-Maze is its simplicity: it does not require rules learning, extensive animal handling, or food or water deprivation for prolonged periods [13].

Although valuable, the Y-maze task has several limitations. The interpretation of results can be complex in models exhibiting locomotor alterations (hypo- or hyper-locomotion), stereotypic behaviours, or anxiety-related novelty avoidance. Furthermore, high performance in the test may reflect rigid and repetitive behaviour rather than actual memory efficiency, making it challenging to distinguish between functional memory and behavioural perseveration [13].

2.3 Y-Maze test: a deeper insight

This paragraph provides a more detailed analysis of the Y-Maze test, as the historical dataset examined in this study consists of performance data from preclinical models that underwent this test.

First, performance in this test is evaluated based on the % of Correct Alternations parameter: the higher this value, the better the performance, while a lower value indicates a cognitive deficit.
It is important to note that in murine models, specifically in Tg2576 mice, this deficit (impairment in spontaneous alternations) typically appears around 9 months of age [6].

To date, most studies that have used this cognitive test have been restricted to a narrow range of parameters derived from the test, such as the total number of entries (N° Entries Tot) and the number of alternations (N° of Alternations).
Recent advances in video-tracking technology have enabled the analysis of various behavioural aspects in addition to arm entry and alternation rates. These technologies

offer contour-based tracking with millisecond resolution, enhancing the accuracy of locomotive behaviour analysis. Furthermore, applying deep learning techniques has greatly improved the accuracy of body point detection, enabling the tracking of multiple body points and the analysis of a rich behavioural repertoire [14].

Among the additional parameters available from the Y-Maze test tracking software, there are the following:

- Duration (total duration of the test)
- Distance (total distance covered by the animal)
- N° Entries A/B/C (number of entries into each arm)
- First Zone Entered (first arm visited by the animal)
- Mean Speed (mean speed performed by the animal)
- Max Speed (maximum speed reached by the animal during the test)
- Rotations (total number of full-body rotations performed)
- Clockwise Rotations (number of full-body rotations in the clockwise direction)
- Anti-clockwise Rotations (number of full-body rotations in the counterclockwise direction)
- Path Efficiency (index of the efficiency of the path taken by the animal to get from the first position in the test to the last position)

Furthermore, the software can provide a set of additional parameters specific to each arm (A, B, C) and to particular sequences (ABC, BCA, CAB, etc.).
For each arm, available statistics include:

- N° of Entries (number of entries in the specific arm)
- N° of Exits (number of exits from the specific arm)
- Time (total time spent in the arm during the test)

- Distance (total distance traveled in the arm)
- Mean Speed (mean speed performed in the arm)

Similarly, for each sequence, the software can track:

- Number (number of times the sequence was performed)
- Time (total time spent performing the sequence)
- Latency to 1st Start (time before the first occurrence of the sequence)
- Average/Minimum/Maximum Distance (average, shortest, and longest distances covered during the execution of the sequence)
- Average/Minimum/Maximum Duration (average, shortest, and longest durations of the sequence performances)

The parameters listed above represent only a portion of the data provided by the Y-Maze test tracking software. In addition to these, the system monitors a wide range of variables, allowing for a comprehensive analysis of locomotor patterns, exploration strategies, and behavioural dynamics.

The analysis of these additional parameters can be useful for a deeper understanding of cognitive function and potential deficits in murine models. They are a potential instrument to integrate and improve the results obtained from the analysis of traditional measures alone.

Although high spatio-temporal resolution tracking data are readily available, and facilitate the analysis of various behavioural factors, the influence of these parameters on the spontaneous alternation rate remains unexplored.

Chapter 3

Dataset Description

Real-world data are typically noisy, incomplete, and may originate from heterogeneous sources. A thorough comprehension of the dataset, including its structure, attributes, and inherent patterns, is essential to ensure its quality and reliability. Knowledge about data is useful for data preprocessing, the first major task of the data mining process.

3.1 Dataset overview

The dataset used in this study consists of data collected from the **Y-Maze** cognitive test performed on murine models. Data represent the performance of subjects in this test, capturing key behavioural metrics that reflect their cognitive abilities. By analysing these performance indicators, potential cognitive disorders can be assessed.

3.1.1 Data origin

The dataset is a historical collection derived from studies conducted over the past 15 years at the IRET Foundation laboratory (Bologna).

The data collection was constructed according to specific criteria. First, all studies conducted in the laboratory were screened to identify those that involved mice and focused on learning and memory tests, specifically Morris Water Maze, Contextual Fear Conditioning, and Y-Maze. From this subset, only studies involving the **Tg2576** murine

model of Alzheimer's disease were selected. To ensure data reliability, additional refinement steps were applied, excluding cohorts that did not meet specific validation criteria, such as those not supported by scientific publications.

For this research, the analysis focuses specifically on the results obtained from the Y-Maze cognitive test.

3.1.2 Data integration

The Y-Maze data were provided by the project partners in two distinct datasets, each containing different aspects of the data collected during the YM test.

The first collection (*Dataset 1: dataset_init.xlsx*) contains labelled data, including subject characteristics and a limited set of key parameters of the test (N° Entries Tot, Visited Arms, % of Correct Alternations). This set of data consists of 556 samples, gathered from nine different studies.

The second collection (*Dataset 2: dataset_ymaze_received.xlsx*) is unlabelled and includes a comprehensive set of 196 features extracted from the YM tracking software, alongside subject-related attributes. These additional parameters provide valuable insights that can enhance the understanding of the subjects' performance. This dataset comprises 462 samples, collected from seven distinct studies.

The *Final Dataset* (*dataset_def.xlsx*) used in this study was created by integrating these two independent collections.

The integration process was carried out on the basis of the common attributes shared by both datasets. Specifically, the additional parameters from *Dataset 2* were merged with the labelled data in *Dataset 1*, ensuring that each sample retained its original classification while incorporating the extended feature set. To ensure completeness and consistency, only the samples present in both collections were included, resulting in a final dataset where each record contains values for every feature.

To facilitate this integration, data harmonization steps were necessary to resolve inconsistencies between the two collections. These included standardising attribute names to ensure correspondence between equivalent features and addressing discrepancies in formatting and structure (details on this cleaning process in Chapter 4).

This integration resulted in a comprehensive dataset of 388 samples, containing a wide range of cognitive performance metrics (see Table 3.1). The final dataset is smaller than the initial collections of data because the two datasets were not entirely compatible. Only samples in both collections were retained for integration.

Dataset	**Samples**	**YM Features**	**Label**
Dataset 1	556	3	Yes
Dataset 2	462	196	No
Final Dataset	**388**	**196**	**Yes**

Table 3.1: Summary of dataset characteristics before and after integration.

By combining these sets of data, a more robust and diverse dataset was obtained, ensuring a more complete representation of cognitive performance and enabling a more accurate and generalizable analysis.

3.1.3 Dataset structure

The final dataset consists of multiple groups of attributes, each providing essential information about the subjects and their cognitive performance.

The attributes are structured as follows:

- **Subject Identifier** (2 attributes): Unique identifier of each subject, consisting of Animal ID and Study.
- **Subject-Related Data** (3 attributes): Biological characteristics of the subjects, including Strain, Gender, and Age (expressed in months).
- **YM Performance Metrics** (196 attributes): Quantitative measures extracted from the Y-Maze test, categorized into:
  - *General Parameters (19 attributes)*: General performance indicators.

- *Arm-Related Parameters (35 attributes x 3 arms)*: Features describing movements in a specific arm of the maze.
- *Sequence-Related Parameters (12 attributes x 6 sequences)*: Metrics analysing specific movement sequences.

- **Label** (3 attributes): Classification of cognitive performance for different levels of certainty, Label_80, Label_90, and Label_100.

The complete list of attributes of the dataset can be found in Appendix A.

The label assigned to each subject indicates the presence or absence of cognitive impairment based on their performance in the Y-Maze test. Specifically, the classification was determined using the **% of Correct Alternations** parameter, which serves as a critical measure of cognitive function. This classification was established according to predefined thresholds, which were derived from a detailed analysis of the % of Correct Alternations parameter.

Each subject was assigned one of three possible labels:

- **IMPAIRED**, if their performance fell below the lower threshold, indicating cognitive impairment.
- **UNIMPAIRED**, if their performance exceeded the upper threshold, suggesting intact cognitive function.
- **MID**, if their performance was within an intermediate range, where there was uncertainty about the presence of cognitive impairment.

The thresholds used to define these categories were established based on the 95% Confidence Interval (C.I.) of the % of Correct Alternations parameter.

To account for different levels of certainty in label assignment, three distinct probability ranges were established: 80-90%, 90-100%, and 100% probability.

As a result, each subject was assigned three distinct labels corresponding to these probability levels. The thresholds for the % of Correct Alternations parameter, which guided the labelling process, are presented in Table 3.2.

Probability	Range Unimpaired (%)	Range Impaired (%)
100%	> 82.50	< 15.50
$< 100\%$ and $> 90\%$	> 68.50	< 53.50
$< 90\%$ and $> 80\%$	> 65.50	< 58.50

Table 3.2: Impaired/Unimpaired range definition.

It can be seen that as the probability increases—indicating a higher certainty in label assignment—the classification thresholds become more restrictive. Specifically, the upper threshold for the Unimpaired category increases, requiring a higher percentage of correct alternations to be classified as unimpaired. Conversely, the lower threshold for the Impaired category decreases, meaning that a lower percentage of correct alternations is needed to be classified as impaired. Therefore, the intermediate range of values corresponding to the Mid category expands, reflecting a wider range of uncertainty in classification.

This labelling approach provides a classification of subjects into three categories of cognitive impairment, considering different levels of certainty in detection, and based on a statistical analysis of the dataset itself.

3.2 Exploratory data analysis

Before the data preprocessing step, it is essential to have an overview of the dataset in order to understand its structure and main characteristics. Knowing the meaning associated with each feature allows to understand its usefulness and impact on the analysis. Exploratory data analysis (EDA) is an approach to analyse and investigate datasets and summarise their main characteristics, often employing statistical graphics and data visualization methods. It helps identify potential errors, find patterns within the data, detect outliers, and reveal meaningful relationships between variables [15].

A detailed examination of the dataset was carried out to interpret the significance of each parameter. This step was crucial to understanding the data and ensuring its effective use.

The preliminary analysis focused on:

- **Feature types and values**: Understanding the different attribute categories in the dataset.
- **Basic statistical descriptions**: Examining key statistics such as mean, standard deviation, and value ranges, to gain insight into data variability and correlations.
- **Feature distributions**: Analysing the distribution of data with respect to certain features to detect patterns or inconsistencies.

This initial exploration provided valuable insights that guided subsequent preprocessing steps, ensuring that only meaningful and non-redundant information was retained for further analysis.

3.2.1 Feature types and values

The dataset consists mainly of numerical attributes, representing various measures obtained from the performance of the subjects in the cognitive test. These numerical features provide quantitative information on the subjects' behaviour during the test, reflecting characteristics of cognitive function that are essential for subsequent analysis.

Only a few features are nominal, which means that they represent categorical information rather than numerical values.

An important nominal attribute is Strain, which distinguishes between different subject groups: Tg2576 and WT. The Tg2576 genotype refers to the transgenic murine model of AD. These subjects are specifically used to study the onset of cognitive decline, as they are genetically predisposed to experience cognitive problems. On the other hand, the WT (Wild-Type) genotype represents the typical form of mice as it occurs in nature. These subjects serve as control models to study cognitive impairment under normal conditions. WT mice generally do not exhibit cognitive deficits, except for age-related changes that naturally occur over time.

Other nominal features are Study (code of the study from which the data originate), Gender and the labels.

3.2.2 Basic statistical descriptions

Basic statistical descriptions can be used to identify properties of the data and highlight which data values should be treated as noise or outliers. These include measures of central tendency, which measure the location of the middle or centre of a data distribution (mean, median, mode, and midrange), and measures of data dispersion (range, quartiles, interquartile range, variance, and standard deviation) [15].

A statistical analysis was performed to understand the characteristics of the data. In addition, a bivariate analysis of the distribution was conducted to identify the patterns and relationships associated with the three classes.

To visually represent these statistical measures, boxplots were used. A boxplot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables or across levels of a categorical variable. The boxplot is a graphical representation that summarizes the distribution of a dataset through five key statistics: the minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum [16].

In addition, scatterplots were used to explore the correlation between features. A scatter plot is one of the most effective graphical methods for determining if there appears to be a relationship, pattern, or trend between two numeric attributes. It shows the joint distribution of two variables using a cloud of points, where each point represents an observation in the dataset. This depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationship between them [17].

The results of the analysis on some of the most important features of the dataset are presented below.

Feature N° of Alternations

Figure 3.1 shows the boxplots for the N° of Alternations data with respect to the three different classes (considering Label_80).

The boxplots reveal that the distribution of values for the N° of Alternations feature is quite similar between the three classes, with overlapping value ranges. This suggests that there is no clear separation between the categories based on this parameter alone.

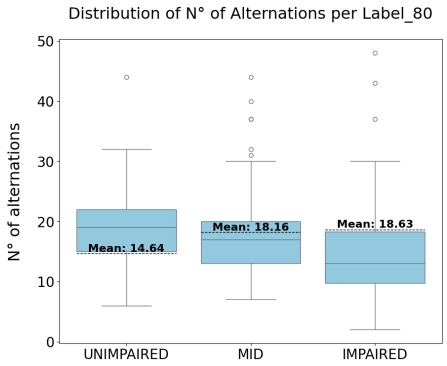
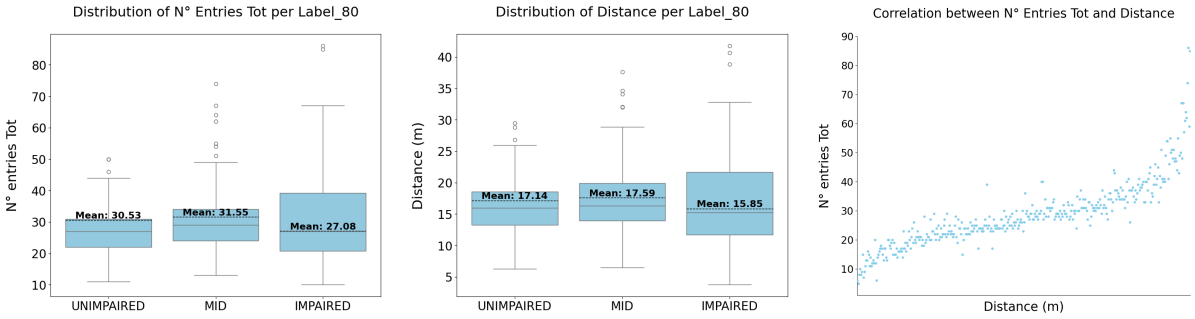


Figure 3.1: Distribution of N° of Alternations per Label_80.

The median values and interquartile ranges show only slight variations, with most data points falling within a range of 10 to 30 alternations. However, some subjects recorded a higher number of alternations, reaching up to 50. These values are marked as outliers in the boxplot, so it is important to check if they are potential errors or simply atypical but valid performances. To ensure data reliability, an analysis is done to check the number of alternations by examining the sequence of visited arms.

Features N° Entries Tot and Distance

Figure 3.2 shows the boxplots for the N° Entries Tot data and the Distance data with respect to the three different classes (considering Label_80).



(a) Boxplot for N° Entries Tot. (b) Boxplot for Distance. (c) Scatterplot for correlation.

Figure 3.2: Distribution of N° Entries Tot and Distance per Label_80.

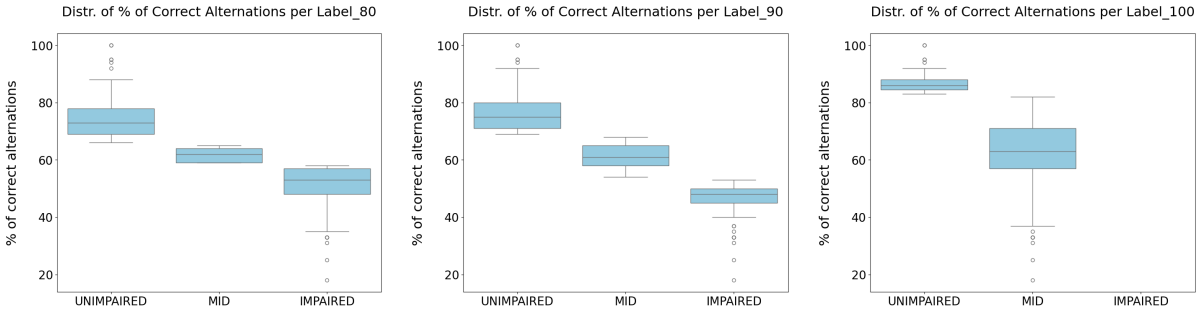
As observed previously, the value ranges are very similar across the three groups. The mean and median values present only little variations. However, both features exhibit a slightly lower range for the Unimpaired group, an intermediate range for the Mid group,

and a slightly higher and broader range for the Impaired group. This pattern suggests that the two features are correlated, which makes sense given their meaning. A higher number of entries into the arms of the maze naturally implies a greater total distance travelled. This positive correlation is also illustrated in the scatterplot (Figure 3.2c), which shows the relationship between the two variables.

In addition, the boxplots suggest the presence of some outliers, but their values remain relatively close to the interquartile range. This suggests that they are likely valid data points rather than errors, representing natural variations in subject performance.

Feature % of Correct Alternations

Figure 3.3 shows the boxplots for the % of Correct Alternations data with respect to the classes, for the three probability levels (considering Label_80, Label_90, Label_100).



(a) Boxplot for Label_80. (b) Boxplot for Label_90. (c) Boxplot for Label_100.

Figure 3.3: Distribution of % of Correct Alternations per different labels.

In this case, it is possible to note how the value ranges are completely non-overlapping across the three classes, indicating a clear separation between groups. As the probability level increases, the balance of IQRs across the three categories shifts, and in Label_100 the more restrictive thresholds result in the exclusion of the Impaired category from this dataset. Furthermore, for each of the three labels, the ranges assumed by the three groups of subjects reflect the thresholds set on the % of Correct Alternations parameter (reported in Table 3.2). These results highlight the consistency of the labelling process, as the observed distributions align with the expected trends derived from the classification thresholds. As expected, it can be concluded that the % of Correct Alternations parameter is highly correlated with the label.

3.2.3 Feature distributions

Data distribution by Label

Since supervised machine learning models will be trained using the dataset, it is essential to first analyse the distribution of subjects across the assigned labels. This analysis is important to understand whether the dataset is balanced.

Table 3.3 presents the number of subjects in each category for the three probability levels (80-90%, 90-100%, and 100%), highlighting how the classification changes as the confidence level increases.

Label	**IMPAIRED**	**MID**	**UNIMPAIRED**	**NaN**
Label_80	116	85	179	8
Label_90	61	180	139	8
Label_100	0	353	27	8

Table 3.3: Distribution of subjects by Label.

As the probability threshold increases from 80% to 100%, the distribution of subjects across the three categories changes significantly. At the 80% threshold, a relatively high number of subjects are classified as Impaired (116) and Unimpaired (179), while fewer fall into the Mid (85) category. However, as the threshold becomes more stringent (90% and 100%), the number of subjects in the Impaired and Unimpaired categories decreases, while the Mid category increases substantially (from 85 at 80% to 353 at 100%). This reflects greater uncertainty at higher confidence levels, as more subjects fail to meet the stricter criteria for clear classification as either Impaired or Unimpaired. Notably, at 100% confidence no subjects are classified as Impaired, reinforcing the challenge of achieving absolute certainty in impairment detection within this dataset.
Additionally, 8 subjects did not receive any label due to their N° Entries Tot being fewer than 10. This low number of entries suggests an unreliable test performance, making it impossible to determine their cognitive status. Consequently, these subjects will be excluded from the analysis.

Data distribution by Strain and Age

To better understand the dataset, an analysis was conducted to examine the distribution of data with respect to Strain and Age. Given their potential influence on the outcome, it is crucial to assess whether the dataset is well-balanced in terms of these characteristics. A balanced distribution ensures that the analysis captures a diverse range of subject profiles without being skewed toward specific age groups or genotypes.

Age	**Tg2576**	**WT**
3	44	54
4	41	56
5	11	21
6	19	28
7	6	14
9	2	28
12	-	20
18	-	20
24	12	12
tot	135	253

Table 3.4: Distribution of subjects by Age and Strain.

Table 3.4 highlights a significant imbalance in data distribution between age groups, likely due to the fact that only a small number of subjects reach older ages. This trend is particularly evident in the Tg2576 group, where no samples are available beyond 9 months, apart from a small subset of 12 subjects at 24 months. Notably, Tg2576 subjects typically begin to exhibit clear signs of cognitive decline around 9 months of age, making the scarcity of samples in this critical period a potential limitation. The lack of data on older Tg2576 mice could lead to biases in subsequent analyses, particularly when comparing age-related trends between Tg2576 and WT subjects. Furthermore, the substantial difference in total sample size between the two strains introduces an additional source of imbalance that could affect the robustness of statistical comparisons.

3.3 Data visualization

Data visualization aims to communicate data clearly and effectively through a graphical representation. There are many alternative techniques for graphically visualizing data, each suited for different types of analysis.

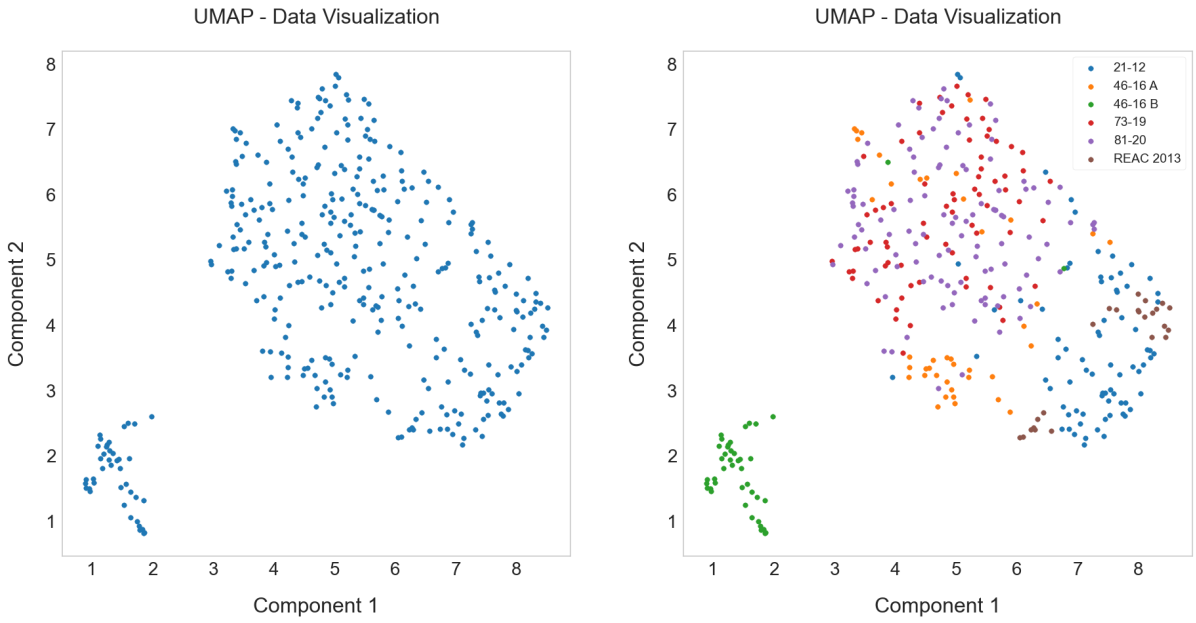
To obtain a 2D visualization of high-dimensional data, it is necessary to apply dimensionality reduction techniques. These techniques allow to represent data with fewer components, aiming to learn relationships between features and create a sparse latent structure. By reducing the dimensionality in a way that preserves as much of the data's structure as possible, it is possible to obtain a visualizable representation that reveals patterns and relationships within the data, providing an initial intuition about its underlying structure.

Uniform Manifold Approximation and Projection (UMAP) is an algorithm for dimension reduction based on manifold learning techniques and ideas from topological data analysis. It works by assuming that the data lie on a manifold, or a lower-dimensional space, embedded in a higher-dimensional space. UMAP first constructs a fuzzy topological representation of the data in its original high-dimensional space, capturing both the local and global structure. It then optimizes the layout of the data points in the lower-dimensional space to preserve as much of this structure as possible. This is done by minimizing a cost function that balances between local neighbourhood relationships and the broader structure of the data. The result is a lower dimensional representation of the data that retains the essential features, allowing for intuitive visual exploration and analysis [18].

To achieve a comprehensive 2D visualization of the dataset, the UMAP technique was applied, reducing the high-dimensional data into two dimensions. This transformation allows the data to be effectively visualized using a scatterplot, providing an overview of the dataset's structure.

As shown in Figure 3.4a, a small cluster of data points appears to be separated from the rest of the dataset. Understanding why the data points in the projection are grouped into distinct clusters is crucial, as it may reveal underlying patterns or substructures within the original high-dimensional space. The presence of well-defined clusters suggests

that certain observations share common characteristics that differentiate them from the rest of the dataset. To determine the factors responsible for this separation, a correlation analysis was performed to identify which features are most closely associated with cluster formation.



(a) UMAP projection.

(b) UMAP projection by Study.

Figure 3.4: Data visualization with UMAP.

Surprisingly, the feature Study was found to be the most influential variable distinguishing the separated cluster from the rest of the dataset in the UMAP projection. As illustrated in Figure 3.4b, where the data points are coloured according to the study of origin, the isolated cluster consists exclusively of samples from Study "46-16 B". This finding is unexpected, as the feature Study merely represents the name of the study from which the data were collected and should not directly influence the dataset's outcomes. Therefore, it is essential to examine why the data from this particular study differ from the others.

To investigate this issue, the distributions of several features with respect to the Study attribute were analysed using swarmplots. Swarmplots are particularly useful for visualizing data distributions, as they display individual data points clearly, ensuring that

they do not overlap and are spatially arranged according to their values [19]. Colouring the points of the swarmplot according to the Study allows a clear visual comparison of how the values of a feature are distributed within and between study groups.

The analysis revealed anomalies in certain features. As shown in Figure 3.5, data points from Study "46-16 B" (represented in blue) exhibit a different distribution from other data in six specific features (A/B/C: time getting closer to zone; A/B/C: time getting further from zone).

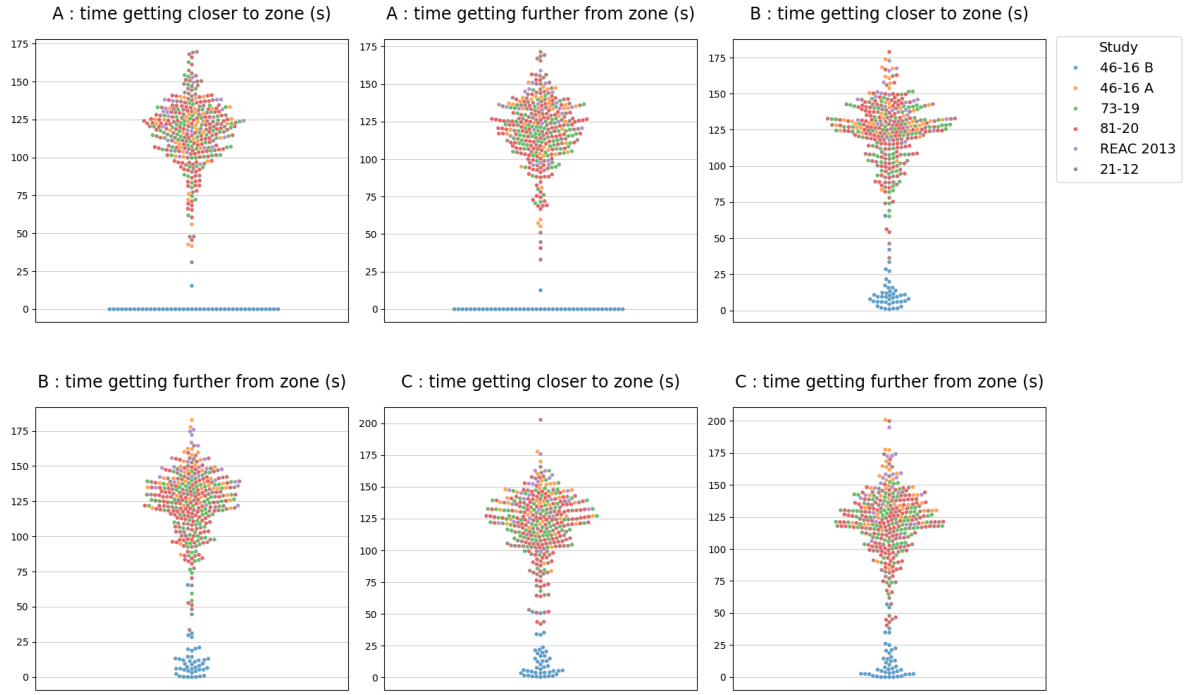


Figure 3.5: Distributions of selected features by Study.

This observation suggests that these specific features may have contributed to the separation seen in the 2D projection. To further validate this hypothesis, these anomalous features were removed from the dataset, and the UMAP reduction and data visualization were regenerated (see Figure 3.6). As anticipated, the previously isolated cluster disappeared, confirming that the separation was driven by these particular features.

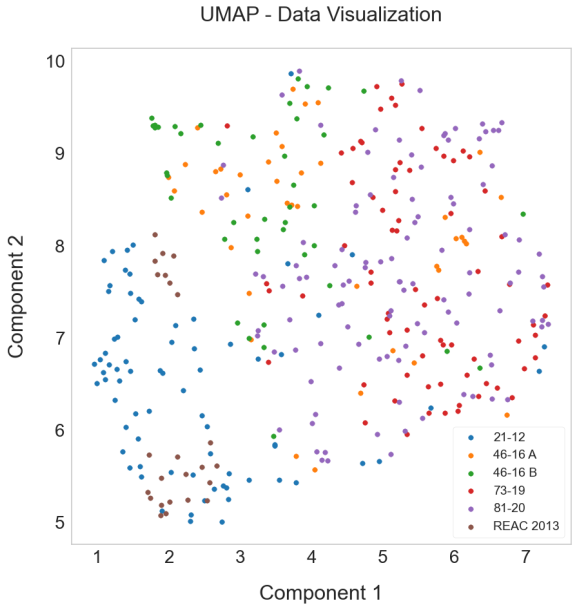


Figure 3.6: Refined data visualization with UMAP.

One possible explanation for the different distribution of the data from the Study "46-16 B" in those features is that the values were altered. Errors in the data can be caused by several factors and may result from the software that generated them or from unintentional mishandling of the data by operators. Furthermore, it is possible that the Study "46-16 B" was conducted under different experimental conditions than the other studies, that influenced these values.

This analysis, therefore, highlighted potential issues in the dataset and provided essential tips for the preprocessing phase. It is important to handle these anomalies to obtain a consistent and reliable dataset for further analysis.

Chapter 4

Data Preprocessing

Low-quality data will lead to low-quality mining results. Data preprocessing is essential to obtain accurate, consistent, and complete data. Data processing techniques, when applied before mining, can substantially improve the overall quality of the results obtained.

4.1 Data cleaning

Data are of high quality if they meet the requirements of their intended use. There are many factors comprising data quality, including accuracy, completeness, consistency, timeliness, believability, and interpretability.

Real-world data are typically inaccurate or noisy (containing errors or values that deviate from the expected), incomplete (lacking attribute values or certain attributes of interest), and inconsistent (containing discrepancies or different copies of the same data). There are many possible reasons for inaccurate data: the data collection instruments used may be faulty; there may have been human or computer errors occurring at data entry; users may purposely submit incorrect data values; errors in data transmission can occur; incorrect data may also result from inconsistencies in naming conventions or inconsistent formats for input fields. Data can be incomplete for different reasons: attributes of interest may not always be available; relevant data may not be recorded; data that were inconsistent with other recorded data may have been deleted. Inconsistent

data can occur for several problems: software errors may occur at data entry; data manipulation by humans may be incorrect; integration of data from different sources may be incompatible.

Data cleaning is the first major step involved in data preprocessing. This phase work to "clean" the data by filling in missing values, smoothing noisy data, identifying or removing outliers, and resolving inconsistencies [20].

4.1.1 Iterative cleaning of raw data

The raw data we received presented numerous quality problems, including inconsistencies, missing values, duplicates, and ambiguous coding. To address these issues, an iterative cleaning process was applied, involving error detection, error resolution, and discussion with domain experts. Interactions with domain experts were particularly important for understanding the meaning attributed to codes in the data, defining the acceptable values of each attribute, and ensuring the validity of the transformations or corrections performed.

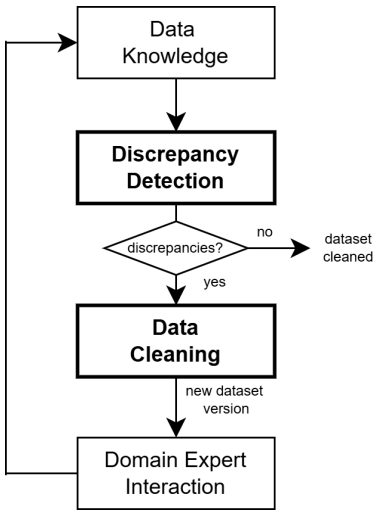


Figure 4.1: Data cleaning process.

Figure 4.1 illustrates the iterative cleaning process applied to obtain complete, accurate, and consistent data.

The cleaning process performed is characterised by four steps:

1. **Data Knowledge**: First, it is essential to obtain and analyse information about the data. This information is partly related to the meaning of the data (domain-specific knowledge) and partly derived from the preliminary exploratory data analysis (data types, value ranges, potential outliers, etc.).

2. **Discrepancy Detection**: Using the acquired knowledge on the data, identify inconsistencies and anomalies within the dataset.

3. **Data Cleaning**: Application of data cleaning techniques to correct the detected discrepancies, resulting in a new version of the dataset.

4. **Domain Expert Interaction**: Confrontation with domain experts, showing the changes made, to ensure that the transformations are correct. Based on the feedback received, repeat the process until no more anomalies are detected.

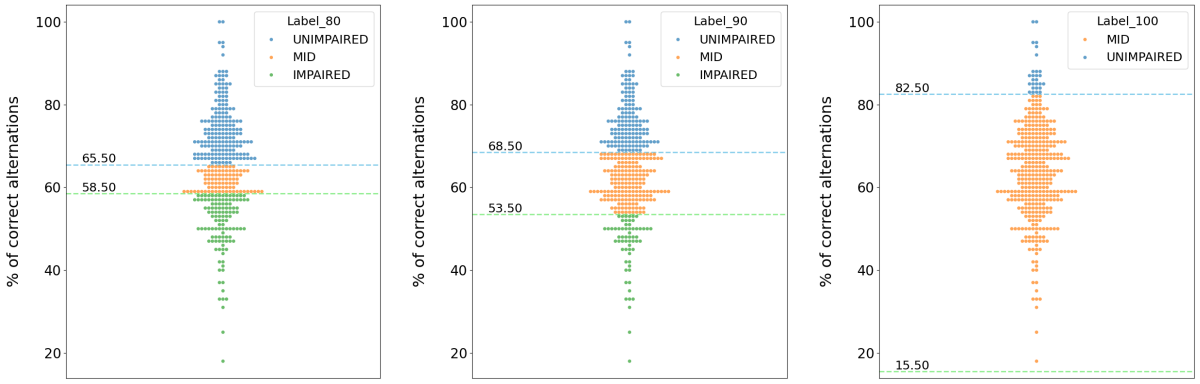
The raw dataset presented several quality issues that were addressed as follows:

- **Inconsistent labels in the Strain and Study features**: In the Strain column, the dataset contained multiple variants of labels of the same genotype, such as TG, TG2576, Tg2576, WT, and wt control. Since these labels actually referred to only two genotypes, they were standardized to Tg2576 and WT to ensure consistency. The same problem occurred in the Study column.

- **Noisy values in % of Correct Alternations**: The expected range of values for this feature is [0, 100]. However, some entries contained values outside this range. To obtain data integrity, rows with alternations exceeding 100% were removed.

- **Duplicated rows (entity identification problem)**: In addition to the duplications founded with automated techniques, duplicate records were identified manually. Specifically, some rows were identical except for the Study feature, indicating that the same subject's performance had been reported in multiple studies. These rows were confirmed to be duplications (equivalent real-world entities from multiple data sources) and removed accordingly.

- **Integration errors**: Some group of rows represented the same subject's performance within the same study but contained missing values in several attributes while sharing identical values in others. These rows were merged into a single comprehensive entry.
- **Labelling errors**: Discrepancies were found in the assigned labels (Label_80, Label_90, Label_100), which did not always match the defined thresholds on the % of Correct Alternations parameter. After applying the necessary corrections, the label assignments were made consistent with the established threshold criteria (Figure 4.2).



(a) Swarmplot for Label_80. (b) Swarmplot for Label_90. (c) Swarmplot for Label_100.

Figure 4.2: Label assignment after threshold-based correction.

In addition, during this cleaning process, anomalous features detected during the exploratory analysis were dropped and an analysis was performed to check whether potential outliers were valid or not. Through this mechanism of error detection and continuous interaction with domain experts, a correct and consistent dataset was obtained.

4.1.2 Integration process

As described in Section 3.1.2, the data were provided in two distinct collections, so an integration step was necessary to obtain a comprehensive dataset.

There are a number of issues to consider during data integration. Schema integration and object matching can be tricky. Some attributes representing a given concept may have different names in different datasets, causing inconsistencies and redundancies. In addition, when matching attributes from one dataset to another during integration,

special attention must be paid to the structure of the data. This is done to ensure that any attribute functional dependencies and referential constraints in the source system match those in the target system. Data integration also involves the detection and resolution of data value conflicts. For example, for the same real-world entity, attribute values from different sources may differ. This may be due to differences in representation, scaling, or encoding [20].

The main challenges faced during the integration phase concern with:

- **Attribute name mismatches**: The corresponding attributes in the two datasets were sometimes labelled differently. For example, the attribute for the subject identification is referred to as "Animal ID" in *Dataset 1* and "Animal" in *Dataset 2*. Also, in the first dataset there are the attributes "Visited arms", "N° Entries Tot", "% of Correct Alternations", while in the second dataset the same attributes are named "Visited zones", "Total Arm Entries", "% Alternations".
- **Label inconsistencies in Strain and Study features**: The same categories were represented with different labels in the two datasets.
- **Data format discrepancies**: The two dataset presented variations in data formats, such as numerical precision or units of measurement.

Redundancy is a significant issue in data integration that can be caused by different factors (not only from attribute name mismatch). For example, conceptual redundancies were identified in the dataset. Specifically, the First Zone Entered feature expresses the same information of A: was 1st zone, B: was 1st zone, and C: was 1st zone, which were consequently removed. Similar cases were also detected, leading to the removal of additional redundant features.

An attribute may be redundant if it can be "derived" from another attribute or set of attributes. These redundancies can be detected by correlation analysis. Given two attributes, such analysis can measure how strongly one attribute implies the other, based on the available data. For numeric attributes, it is possible to evaluate the (linear) correlation between two attributes, A and B, by computing the Pearson correlation coefficient. It is essentially a normalized measurement of the covariance, such that the

result always has a value between -1 and 1. If the result is greater than 0, then A and B are positively correlated, meaning that the values of A increase as the values of B increase. The higher the value, the stronger the correlation (i.e., the more each attribute implies the other). Hence, a higher value may indicate that A (or B) can be removed as redundancy. If the resulting value is equal to 0, then A and B are independent and there is no correlation between them. If the resulting value is less than 0, then A and B are negatively correlated, as the values of one attribute increase as the values of the other attribute decrease [20].

An analysis was conducted to identify any highly correlated attributes to be removed, computing the Pearson correlation coefficient between pairs of features. The results are reported in a heatmap, which is useful because it allows quick identification of attribute pairs with high correlation through an intuitive graphical representation [21].

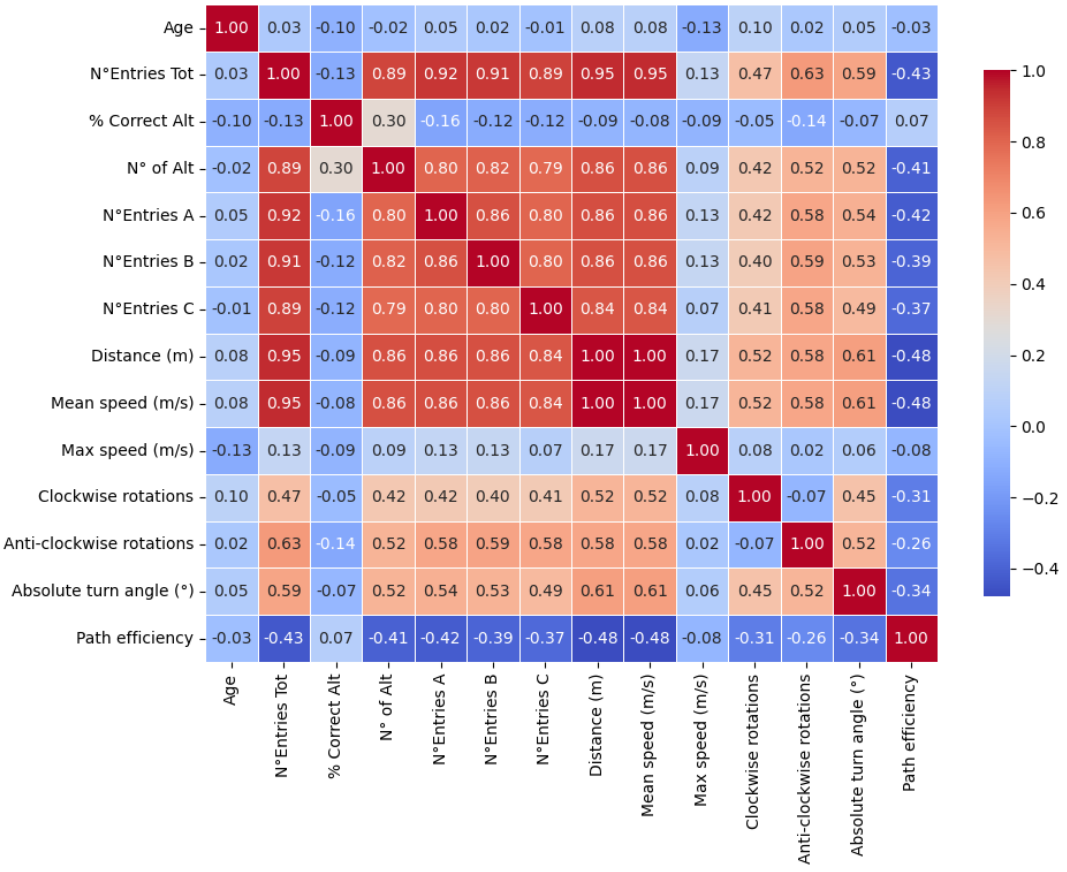


Figure 4.3: Correlation between features.

In Figure 4.3, the correlation between the main characteristics of the dataset is shown. First, it is possible to note that the Distance feature is completely positively correlated with the Mean Speed feature: this is because the Mean Speed is computed by dividing the Distance by the Duration of the test, which remains constant for each tuple. As a result, Mean Speed does not provide any additional information to the dataset and was removed to avoid redundancy. Additionally, it can be observed that N° Entries Tot is highly correlated with the features that represent the number of entries in each arm (N° Entries A, N° Entries B, N° Entries C). This correlation is not strange, as these features describe the same concept. However, N° Entries A/B/C express additional information beyond the aggregate feature. For this reason, it was initially decided not to remove these features and to evaluate which ones to maintain in the feature selection phase. Also, the N° Entries Tot feature is highly correlated with Distance, as demonstrated in Section 3.2.2. In general, the features relating to the number of entries, distance, and number of alternations exhibit strong correlations with each other, despite the different concepts expressed. Therefore, automated feature selection techniques will be used to identify the most relevant features.

Those reported in Figure 4.3 are the most critical features with regard to correlation analysis. The other features, i.e. those specific to individual arms and sequences, did not show potential redundancies.

4.1.3 Handling missing data

Often in data there are many tuples with no recorded value for several attributes. Managing missing values is important to obtain a complete dataset and to have data in a format that can be processed by algorithms.

There are different approaches to handle missing values: ignore the tuples, fill in the missing values manually, use a global constant to fill in the missing values, use a measure of central tendency for the attribute (e.g., the mean or median) to fill in the missing values, use the attribute mean or median for all samples belonging to the same class as the given tuple, use the most probable value to fill in the missing values.

Table 4.1 shows the count of NaN values in the dataset. The table summarizes the impact of different missing data handling strategies on the dataset. Initially, the dataset

contained 165 rows and 34 columns with missing values (NaN). Since the total number of samples is 388, these 165 rows represent approximately 43% of the data. Removing all of them would have significantly reduced the size of the dataset and its representativeness.

Scenario	**Rows with NaN**	**Col. with NaN**	**Col. removed**
Initial	165 ($\sim$ 43%)	34	-
10% threshold	84 ($\sim$ 22%)	14	20
5% threshold	**14** ($\sim$ 4%)	**4**	**30**

Table 4.1: Impact of missing data handling on rows and columns.

To minimize data loss, an approach was first applied to remove columns with the highest number of missing values before eliminating any remaining rows. By setting a maximum threshold of 5% of accepted missing values per column, 30 columns were removed, reducing the number of rows with missing values to 14 (only 4% of the total dataset). If the threshold had been increased to 10%, only 20 columns would have been removed instead of 30, increasing the remaining rows with NaN to 84 (that is more than the 20% of the dataset). Given that the removed columns mainly contained information related to the analysis of specific sequences, which had minimal impact on the determination of the label, they were considered secondary and less relevant. Therefore, column removal was preferred over row removal to preserve as many samples as possible. This approach ensured a balance between preserving valuable data and maintaining a clean, analysable dataset.

4.2 Feature engineering and transformation

After the cleaning phase, additional data preprocessing steps are applied to obtain high-quality data that can be processed by machine learning algorithms. Various transformations must be performed on the dataset before it can be processed effectively by ML models. First, since machine learning models can only handle numerical data, categorical attributes must be transformed into a proper configuration. In addition, lots of these

algorithms rely on distance-based analysis of samples. Since distances are very sensitive to the order of magnitude of the features, it is necessary to apply data normalization techniques before the training of models. Moreover, machine learning models perform well if the dataset is well-balanced in terms of the number of samples and attributes. A larger number of features requires a higher number of samples for proper representation. To address this issue, dimensionality reduction techniques or feature selection methods can be used to optimize model performance.

These preprocessing steps can dramatically affect the model training phase. If they are not performed properly, the results of the models will be unreliable. However, there is no standard rule for applying these techniques, as the best approach depends on both the dataset characteristics and the models being used. It is important to have an overview of the available techniques and to determine the most suitable ones for each case. The selection of these techniques is made by analysing and comparing different approaches to identify the most effective transformations for each model.

This section outlines the preprocessing techniques explored and compared during data preparation. In the following chapters, which focus on the modelling phase, the specific techniques chosen for each trained model will be detailed, and the results obtained will be presented.

4.2.1 Feature encoding

Encoding categorical features is essential to convert them to numerical features that can be used with machine learning estimators. One possibility is to convert categories into integer codes. However, such integer representation is not properly if the categorical feature is not ordinal because most models would interpret them as being ordered. Another most useful possibility to convert categorical features is to use a one-hot or dummy encoding. This type of encoding transforms each categorical feature with n_categories possible values into n_categories binary features, with one of them 1, and all others 0.

The **one-hot-encoder** [22] technique was selected to transform categorical features, such as Gender and Strain. Since there are only a few categorical features in the dataset, and these features have a small number of unique values, there were no problems with the

number of columns exploding. Table 4.2 shows how one-hot-encoding works for the Strain feature transformation. Since the feature Strain has only two possible values (Tg2576 and WT), it is converted into two binary columns (Strain_Tg2576 and Strain_WT). Each row in the transformed dataset has the value 1 in the column corresponding to its category and 0 in the other.

	Strain	**Strain_Tg2576**	**Strain_WT**
sample_1	Tg2576	1	0
sample_2	WT	0	1

Table 4.2: Feature encoding with one-hot-encoding.

In addition, the label column is a categorical feature. For this specific attribute, the **label encoder** [23] method was used. It encodes target labels with values between 0 and n_classes - 1. This transformation is specifically intended for the encoding of target features. Table 4.3 shows the encoding of the label (considering Label_80). It is possible to understand the mapping between the original classes and the assigned integer value.

	Label_80	**Label_80**
sample_1	IMPAIRED	0
sample_2	MID	1
sample_3	UNIMPAIRED	2

Table 4.3: Feature target encoding with label encoding.

4.2.2 Normalization

Normalization of datasets is a common requirement for many machine learning estimators. The value range of an attribute is directly related to the measurement unit used and can affect the data analysis. In general, attributes with initially large ranges tend to outweigh attributes with initially smaller ranges, meaning that they may have a greater effect on the analysis. To help avoid dependence on the choice of measurement units, the data should be normalized or standardized. This involves transforming the

data to fall within a smaller or common range such as [-1, 1] or [0, 1]. Data normalization attempts to give all attributes equal weight.

There are many methods for data normalization. Normalization techniques that were compared and applied for data preparation in the project are described below [20].

- **Z-score normalization**: Z-score normalization standardizes features by removing the mean and scaling to unit variance. Supposing to have an attribute A, a value, v_i, of A is normalized to v_i' by computing

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A} \tag{4.1}$$

  where $\bar{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A.

- **Min-max normalization**: Min-max normalization performs a linear transformation on the original data. It transforms numerical features by scaling each feature to a given range (e.g. [0.0, 1.0]), preserving the relationship among the original data values. Supposing that min_A and max_A are the minimum and maximum values of an attribute A, min-max scaling maps a value, v_i, of A to v_i' in the new range $[new\_min_A, new\_max_A]$ by computing

$$v_i' = \frac{v_i - min_A}{max_A - min_A} (new\_max_A - new\_min_A) + new\_min_A. \tag{4.2}$$

- **L1 / L2 normalization**: L1 and L2 normalization normalize samples individually to unit norm. Each sample is rescaled independently of other samples so that its norm (L1 or L2) equals one. L1 normalization, also known as Manhattan Norm, involves transforming the data such that the sum of the absolute values of the vector (a column in a dataset) is equal to 1. L2 normalization, or Euclidean Norm, transforms each sample so that the sum of the squares of its elements is equal to 1. Supposing that v_i is a value of an attribute A, mathematical formulae to compute L1 (4.3) and L2 (4.4) normalization are:

$$v_i' = \frac{v_i}{\sum_j |v_j|} \tag{4.3}$$ and $$v_i' = \frac{v_i}{\sqrt{\sum_j {v_j}^2}}. \tag{4.4}$$

It is not possible to determine in advance the best normalization technique to apply. The most effective approach is to test different methods and identify the one that yields the highest performance for each model.

4.2.3 Dimensionality reduction

The "curse of dimensionality" refers to the problems associated with multivariate data analysis, as dimensionality increases. Dimensionality reduction is the process of reducing the number of variables under consideration by applying a transformation to the dataset to obtain a reduced or "compressed" representation of the original data. Unlike feature selection, which reduces the attribute set size by retaining a subset of the initial set of attributes, these approaches combine the essence of attributes by creating an alternative, smaller set of variables.

Different dimensionality reduction techniques were analysed. In particular, Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) were tested.

- **Principal Component Analysis (PCA)**: PCA reduces the number of dimensions in large datasets to "principal components" that retain most of the original information, by transforming potentially correlated variables into a smaller set of variables. These components are linear combinations of the original attributes that have the maximum variance compared to other linear combinations. This technique transforms the original dataset into a new coordinate system that is structured by the principal components (axes of the new space). The original data are thus projected onto a much smaller space, resulting in dimensionality reduction [24].

- **Linear Discriminant Analysis (LDA)**: LDA is a supervised method that separates multiple classes with multiple features through data dimensionality reduction. It works by identifying a linear combination of features that separates or characterizes two or more classes of objects. This maximizes the between-class variance and minimizes the within-class variance. LDA does this by projecting data with two or more dimensions into one dimension so that it can be more easily classified [25].

These techniques may be useful to improve model performance, but they affect the ability to analyse the impact of original features on results.

4.2.4 Feature selection

Datasets may contain hundreds of attributes, many of which may be irrelevant to the mining task or redundant. Leaving out relevant attributes or keeping irrelevant attributes may be detrimental, causing confusion for the mining algorithm used. The goal of feature selection is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Since determining which attributes are useful can be a difficult and time-consuming task, automated feature selection techniques can be employed.

The feature selection algorithms that were analysed are described below.

- **Recursive Feature Elimination (RFE)**: Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of RFE is to select features by recursively considering smaller and smaller sets of features. First, the estimator (e.g., Logistic Regression) is trained on the initial set of features, and the importance of each feature is obtained. Then, the least important features are pruned from current set of features. That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached [26].

- **Recursive Feature Elimination with Cross-Validation (RFE-CV)**: RFECV performs RFE in a cross-validation loop to find the optimal number of features. The number of features selected is automatically tuned by fitting an RFE selector on the different cross-validation splits. The performance of the RFE selector is evaluated using a scorer for different numbers of selected features and aggregated together. Finally, the scores are averaged across folds and the number of features selected is set to the number of features that maximize the cross-validation score [27].

- **SelectKBest**: SelectKBest is a feature selection method that selects the top K features based on statistical tests. It evaluates each feature individually by measuring its relevance to the target variable using a scoring function, such as ANOVA F-score. The method ranks the features according to their scores and retains only the K highest-scoring ones [28].

- **SelectFromModel**: SelectFromModel is a meta-transformer that can be used alongside any estimator that assigns importance to each feature. Characteristics are considered irrelevant and removed if the corresponding importance is lower than the threshold parameter provided. [29].

These different techniques were compared in order to analyse which one provided the best feature selection, thereby improving model performance.

4.3 Additional steps for supervised learning

In supervised machine learning, data splitting and class balancing are crucial preprocessing steps that directly impact model performance and fairness. Specifically, the goal of a classifier is to predict the class of previously unseen instances. To achieve this, the model is trained on a given subset of the dataset, the training set, which includes both feature values and corresponding class labels. Once trained, the classifier is applied to unseen data, the test set, to predict class labels. The performance of the estimator is evaluated by comparing the predicted labels with the true labels of the test instances. Achieving high performance is not solely dependent on the quantity of training data, but also on its quality and representativeness. In particular, ensuring that all classes are adequately represented in the training set is essential for building a model that performs fairly and accurately across different categories.

4.3.1 Data splitting

To build and evaluate a supervised model is necessary the data splitting preprocessing step, which splits the dataset into two sets, the training and test sets. These two sets should be independent of each other to ensure that the induced model can accurately predict the class labels of instances it has never seen before. Typically, 80% of the initial dataset is assigned to the training set, while the remaining 20% is used as the test set. This proportion is chosen because machine learning models generally require a large amount of data to learn meaningful patterns and generalize well to new data. A larger training set helps the model capture complex relationships within the data, reducing the

risk of underfitting. At the same time, a sufficiently large test set is needed to provide a reliable evaluation of the model's performance on unseen data. In addition, the dataset should be split while maintaining the class distribution across both sets. A stratified split ensures that both the training and test sets contain a representative proportion of each class.

Data splitting, with attention to the distribution of classes, was applied to obtain training and test sets for the classification task of the project. This step was performed on the dataset after data cleaning, using the **train_test_split** [30] function from scikit-learn, with the stratify parameter to ensure that the class distribution is preserved in both the training and test sets.

	IMP.	**MID**	**UNIMP.**	**Tot**	**% of Tot**
Dataset cleaned	113	82	174	369	100%
Training Set	90	66	139	295	80%
Test Set	23	16	35	74	20%

Table 4.4: Dataset split into training and test sets.

Table 4.4 shows the split of the dataset into training and test sets, considering Label_80 as target label. It is possible to note that the distribution of classes in the two sets is consistent with the distribution of classes in the original cleaned dataset.

4.3.2 Class balancing

Handling class imbalance is not only relevant during the data splitting phase. Data balancing represents an additional preprocessing step that must be applied to the training set before the modelling. A training set is imbalanced when the distribution of instances among the classes is significantly uneven, meaning that some classes have lots of samples while others are underrepresented. This can lead to biased models that perform well on the majority class but poorly on the minority class. The training set should be balanced to ensure that the model learns to correctly predict all classes.

There are different techniques available for data balancing, each with different advantages and weaknesses:

- **Upsampling**: Upsampling, or oversampling, increases the number of instances in minority classes by duplicating existing samples until all classes are of equal size. The main advantages of this technique are the absence of information loss and the ability to increase the size of the dataset at low cost. Upsampling also has some disadvantages, including a higher risk of overfitting, the potential introduction of noise, and increased computational complexity during model training [31].

- **Downsampling**: Downsampling, or undersampling, decreases the number of instances in majority class by removing data such that it matches the size of the minority class. Downsampling offers several advantages, including lower storage requirements, faster training times, and a reduced risk of overfitting. It also has some disadvantages, such as the potential loss of information and the introduction of bias in the training data [32].

- **SMOTE**: The Synthetic Minority Oversampling Technique, or SMOTE, is an upsampling technique that synthesizes new data points from the existing points in the minority class, by interpolating between existing instances and their nearest neighbours in the feature space. SMOTE counters the problem of overfitting in random oversampling by adding previously unseen new data to the dataset rather than simply duplicating pre-existing data. However, SMOTE's artificial data point generation adds extra noise to the dataset, potentially making the classifier more unstable [33].

- **Class weighting**: Class weighting is a different technique of data balancing, it addresses class imbalance without altering the size of the dataset. When using class weighting, the algorithm is informed to give more importance to certain classes during training by adjusting the loss function. This is done by assigning a higher weight to underrepresented classes, so that errors on these classes have a greater impact on the optimization process of the model [34].

As shown in Table 4.4, the training set is imbalanced, with the Unimpaired class

that is overrepresented with respect to the other classes. For this reason, data balancing methods were compared during the preprocessing phase, and the best balancing technique for each model was selected and applied to the training set.

The results of the different methods were quite similar, with SMOTE and class weighting being the most effective. Undersampling was excluded because the dataset already has a limited number of samples, and oversampling could lead to overfitting due to the significant imbalance. In the end, class weighting was selected as the best balancing method, as it does not change the dataset by adding or removing samples. This technique was then applied to all models.

4.4 Preprocessing pipeline

All the preprocessing steps described are essential to prepare the dataset for processing by machine learning algorithms. Data cleaning is performed first to ensure that the subsequent steps can be applied effectively. After this initial step, the transformation phases can be structured within a pipeline. Pipelines are particularly useful because data processing often follows a fixed sequence of steps.

In the project, pipelines were used to manage preprocessing, for both the classification task and the unsupervised exploratory analysis.

4.4.1 Preprocessing technique selection via cross-validation

For each transformation step (described in Section 4.2), different techniques were evaluated to determine the most effective for each model. For example, various normalization techniques, such as z-score normalization and min-max normalization, were tested to identify the best approach for each model.

To select the optimal technique for each preprocessing step, cross-validation (CV) was applied, ensuring a robust evaluation of different methods. Cross-validation is particularly important because, when comparing different techniques for a specific model, it is essential to partition the dataset into distinct parts. One part, the training set, is used to train the model, while the other part, the validation set, is used to assess its performance. However, splitting the available training data into two sets can significantly reduce

the number of samples that can be used to learn the model and can lead to results that depend on a particular random split of the dataset. Using cross-validation, the validation set is no longer needed. In k-fold CV, the training set is split into k smaller sets (also called "folds"). The model is trained on k-1 folds and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the validation set once. The final performance metric is obtained by averaging the results across all iterations, providing a more reliable estimate of the performance of the model [35]. In this project, **stratified k-fold cross-validation** was applied, ensuring that each fold contained approximately the same percentage of samples of each target class as the complete set.

Using cross-validation, different transformation techniques were compared, ensuring that the most effective approach was chosen. Once the best technique for each transformation step had been identified, it was incorporated into the pipeline.

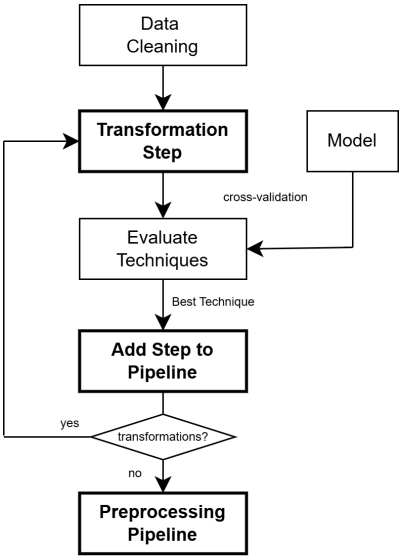


Figure 4.4: Transformation technique selection via cross-validation.

4.4.2 Pipeline for the classification task

The primary objective of the project is to develop a classifier capable of detecting cognitive impairment. To this end, different supervised machine learning models have been compared in order to identify the most accurate estimator.

This section describes the pipeline implemented for the classification task, which

manages the preprocessing steps as well as the final model selection. The pipeline integrates all necessary preprocessing steps to prepare the dataset and the application of a final estimator.

Before applying the pipeline, the dataset is first cleaned and then partitioned into training and test sets. The pipeline is subsequently fitted on the training set during model selection and applied to the test set during performance evaluation. The preprocessing pipeline was implemented using scikit-learn's **Pipeline** [36] and **ColumnTransformer** [37] classes and is composed of the following main steps:

1. **Encoding** and **Normalization**: Categorical features are encoded using OneHotEncoder. Numerical features are standardized using the normalization technique selected (e.g., StandardScaler or MinMaxScaler).

2a. **Dimensionality Reduction** *(if enabled)*: A dimensionality reduction method (e.g., PCA or LDA) can be applied to reduce feature space complexity while retaining the most informative components.

2b. **Feature Selection** *(if specified)*: Alternatively, a feature selection strategy (e.g., RFE or RFECV) may be used instead of dimensionality reduction.

3. **Model**: The final step of the pipeline includes the supervised model (e.g., Logistic Regression) for classification. The model includes built-in handling of class imbalance via *class_weight="balanced"*.

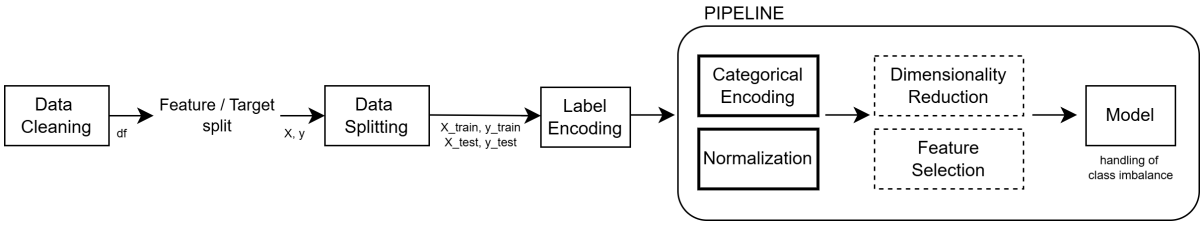


Figure 4.5: Data preprocessing workflow for the classification task.

The implementation of the described pipeline can be found in Appendix B. Figure 4.5 illustrates the complete preprocessing workflow implemented for the classification task,

including all mandatory (categorical feature encoding and numerical feature normalization) and optional (dimensionality reduction and feature selection) preprocessing steps in the pipeline.

4.4.3 Pipeline for the unsupervised analysis

The unsupervised exploratory analysis of the project aims to identify patterns in the data and explore potential predictive factors of cognitive impairment, by applying clustering techniques. Since the goal is to analyse the characteristics of the clusters and interpret their meaning, there is no need to split the dataset into training and test sets. Labels are not used, and no performance evaluation is conducted for these unsupervised models.

For this reason, the preprocessing for the unsupervised analysis is managed through a ColumnTransformer, rather than a full pipeline. The ColumnTransformer is used to handle the encoding of categorical features and the normalization of numerical features, which are applied to the entire dataset. Initially, dimensionality reduction and feature selection are excluded from the pipeline to ensure a comprehensive exploration of the data. Details on their potential application are provided in Chapter 7.

Figure 4.6 shows the complete preprocessing workflow implemented for the unsupervised analysis.

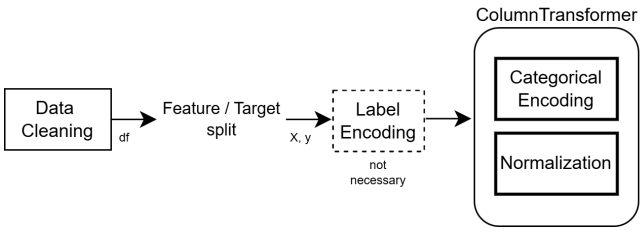


Figure 4.6: Data preprocessing workflow for the unsupervised analysis.

Once the transformer is applied, the dataset is fully preprocessed and ready to be passed to the unsupervised models.

Chapter 5

Overview on Machine Learning Models

Machine learning models can automatically learn to recognise complex patterns and make intelligent decisions based on data. Their potential lies in their adaptability and scalability, allowing them to be applied to address several tasks across a wide range of domains.

5.1 Supervised learning

Supervised learning is a machine learning technique that uses labelled datasets to train models to identify the underlying patterns and relationships between input features and class labels. Specifically, a supervised model is created using a given set of data, the training set, which contains attribute values as well as class labels for each instance. Once trained, the model is applied to new data, the test set, to predict class labels. The performance of the estimator is evaluated by comparing the predicted labels with the true labels of the test instances. The goal of the learning process is to create a model that can predict correct outputs on previously unseen data. Therefore, supervised learning is basically a synonym for classification.

In this study, supervised learning algorithms are employed to build a model, or classifier, capable of detecting cognitive impairment.

The following sections present different classification algorithms applied to the classification task of the project. Each machine learning model is based on different concepts and mechanisms, which may make it more or less suitable for the specific use case.

5.1.1 Logistic Regression

Logistic Regression (LR) [38] is a supervised machine learning algorithm widely used for classification tasks, such as diagnosing diseases by assessing the presence or absence of specific conditions based on test results from subjects. It estimates the probability that a given input belongs to a particular category.

In its standard form, Logistic Regression is designed for binary classification, where the dependent variable has only two possible outcomes. The LR model uses the **logistic (or sigmoid) function** to transform a linear combination of input features into a probability value in the range [0, 1]. This probability represents the likelihood that a given input corresponds to one of the two predefined classes. The logistic function is defined as:

$$P(x) = \frac{1}{1 + e^{-t}} \tag{5.1}$$

where t is a linear combination of the input variables:

$$t = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_k x_k. \tag{5.2}$$

Each coefficient β_i represents the influence of the corresponding feature x_i on the log-odds of the outcome. Higher positive or negative values indicate a stronger impact on the classification decision. The model parameters (coefficients) are estimated by maximizing the likelihood of observing the training data, which is equivalent to minimizing the negative log-likelihood loss function:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right] \tag{5.3}$$

where N is the total number of samples in the dataset, y_i is the true label for sample i, and $\hat{y}_i$ is the predicted probability that sample i belongs to class 1.

The Binary Logistic Regression model can be extended to handle multiclass classification problems, where the target variable has more than two categories. This extension is known as Multinomial Logistic Regression. In the multinomial case with K classes, the model estimates K probability scores (one for each class), using the **softmax function** to ensure that all predicted probabilities are in the range [0, 1] and sum to 1:

$$P(y = k \mid x) = \frac{e^{t_k}}{\sum_{j=1}^{K} e^{t_j}}, \quad \text{for } k = 1, \ldots, K \tag{5.4}$$

where each $t_k = \beta_{k0} + \beta_{k1}x_1 + \beta_{k2}x_2 + \ldots + \beta_{kn}x_n$ is a linear combination of the input features for class k.

5.1.2 Support Vector Machine

Support Vector Machine (SVM) [39] is a supervised algorithm commonly used in classification problems. SVM classifies data by finding an optimal line or hyperplane that maximizes the distance between each class in an n-dimensional space.

The number of features in the input data determines whether the hyperplane is a line in a 2-D space or a plane in an n-dimensional space. Since multiple hyperplanes can be found to differentiate classes, maximizing the margin between points enables the algorithm to find the best "decision boundary" between classes (i.e., the **maximum marginal hyperplane**). Lines adjacent to the optimal hyperplane are known as "support vectors", as these vectors run through the data points that determine the maximal margin. Mathematically, this separating hyperplane can be represented as:

$$wx + b = 0 \tag{5.5}$$

where w is the weight vector, x is the input vector, and b is the bias term. Thus, the weights can be adjusted so that the support vectors defining the "sides" of the margin satisfy the formula:

$$(wx_i + b)y_i \geq a. \tag{5.6}$$

Support Vector Machine can handle both linear and non-linear classification tasks. When data are not linearly separable, **kernel functions** are used to transform the data

into a higher-dimensional space to allow linear separation. The choice of kernel function, such as linear kernels, polynomial kernels, radial basis function (RBF) kernels, or sigmoid kernels, depends on the data characteristics and the specific use case.

5.1.3 Random Forest

Random Forest (RF) [40] is a flexible machine learning model commonly used for classification tasks that combines the outputs of multiple decision trees to obtain a single result. The term "forest" refers to the ensemble of **uncorrelated decision trees**, which are merged to reduce variance and increase accuracy.

Each decision tree in the forest is a hierarchical model composed of a root node, internal decision nodes, branches, and leaf nodes. At each decision node, the input data is split based on the value of a selected feature. The split aims to separate the data in a way that maximizes a chosen criterion (e.g., information gain or Gini impurity). The process is repeated until a stopping condition is met, and the leaf nodes represent the final class predictions. Although decision trees are easy to interpret and train, they tend to suffer from high variance and can easily overfit the training data. Random Forest addresses these limitations by building an ensemble of trees and aggregating their predictions to obtain more stable and accurate results.

Random Forests can be built using different **ensemble methods**, such as bagging and boosting. In the first method, bagging, multiple classifiers are trained in parallel on different random subsets of the training data, sampled with replacement. On the other hand, boosting builds classifiers sequentially, where each new model focuses on the errors made by the previous ones.

The Random Forest algorithm is essentially an extension of the bagging method, as it uses both bagging and feature randomness to create an uncorrelated forest of decision trees. During the training of each tree, not only are the training data sampled with replacement, but at each split a random subset of features is considered instead of evaluating all available features. This feature randomness ensures that the individual trees are more diverse and less correlated, further improving the ensemble's ability to generalize.

5.1.4 K-Nearest Neighbors

K-Nearest Neighbors (KNN) [41] algorithm is a non-parametric, supervised learning method commonly used for classification tasks. It makes predictions based on the proximity of data points, assigning a label to an unknown instance by analysing the labels of its closest neighbors in the training set.

KNN operates on the principle of **learning by analogy**: given a test instance, the algorithm compares it with previously observed instances (training tuples) to determine its class. Each training sample is represented as a point in an n-dimensional feature space, where n is the number of input attributes. All training data are stored in this multidimensional space without any model being explicitly trained. When given a new, unlabelled instance, the classifier searches the pattern space for the k training tuples that are closest to the unknown sample. These k points are the "nearest neighbors" of the unknown tuple. The predicted class is then typically determined by majority vote among the neighbors.

The choice of k, the number of nearest neighbors, is crucial for the model's performance: a small k may lead to overfitting and sensitivity to noise, while a large k can cause underfitting. Typically, k is selected through cross-validation to balance bias and variance.

The concept of "closeness" in K-Nearest Neighbors is defined in terms of a **distance metric**, which quantifies how similar or dissimilar two data points are in the feature space. The most commonly used distance functions include the Euclidean, Manhattan, and Minkowski distances.

Euclidean distance represents the straight-line distance between two points in an n-dimensional space. Given two points $X_1 = (x_{11}, x_{12}, ..., x_{1n})$ and $X_2 = (x_{21}, x_{22}, ..., x_{2n})$, it is computed as:

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}. \tag{5.7}$$

Manhattan distance measures the total absolute difference between the dimensions. It is commonly displayed with a grid, illustrating how one might navigate from one point

to another through the grid streets:

$$dist(X_1, X_2) = \left(\sum_{i=1}^{n} |x_{1i} - x_{2i}| \right). \tag{5.8}$$

Minkowski distance is the generalized form of the Euclidean and Manhattan distances through a parameter p, which controls the distance metric's behaviour. When $p = 2$, it reduces to the Euclidean distance; when $p = 1$, it becomes Manhattan distance:

$$dist(X_1, X_2) = \left(\sum_{i=1}^{n} |x_{1i} - x_{2i}| \right)^{1/p}. \tag{5.9}$$

5.2 Unsupervised learning

Unsupervised learning is a machine learning technique that analyses unlabelled data. Unlike in classification, the class label of each sample is not used. These algorithms discover similarities and differences in data, finding hidden patterns or data groupings without the need for human intervention. Unsupervised learning is the ideal solution for exploratory data analysis.

A common unsupervised learning approach is clustering. Clustering is the process of partitioning a set of data objects into subsets. Each subset is a cluster, such that objects in a cluster are similar to one another, yet dissimilar to objects in other clusters. Dissimilarities and similarities are assessed on the basis of the attribute values describing the objects and often involve distance measures.

In this project, clustering techniques are employed to discover previously unknown groups within the data and to uncover novel patterns, investigating possible predictive factors of cognitive impairment.

Different clustering methods may generate different clusterings on the same dataset, depending on their underlying algorithmic assumptions and strategies. To ensure a thorough exploratory data analysis, multiple clustering techniques have been applied and compared.

5.2.1 K-Means

K-Means clustering [42] is a **partitioning algorithm** that split the dataset X into k distinct clusters, $C_1, ..., C_k$, such that $C_i \subset X$ and $C_i \cap C_j = \emptyset$ for $(1 \leq i, j \leq k)$. Each data point is assigned to exactly one cluster, with the goal of minimizing the **within-cluster variation**.

K-Means is a centroid-based technique, meaning that each cluster is represented by its **centroid**, which corresponds to the mean μ_j of the data points assigned to that cluster. The algorithm attempts to form clusters of approximately equal variance by minimizing the inertia, or within-cluster sum-of-squares, which measures how internally coherent clusters are:

$$\sum_{i=0}^{n} \min_{\mu_j \in C} (\|x_i - \mu_j\|^2). \tag{5.10}$$

This objective function tries to make the resulting k clusters as compact and as separate as possible.

The algorithm can basically be described as three steps. First, it randomly selects k objects in X, each of which initially represents a cluster centre. For each of the remaining objects, an object is assigned to the cluster to which it is most similar, based on the Euclidean distance between the object and the cluster centre. Then, for each cluster, it computes the new mean using the objects assigned to the cluster. All the objects are then reassigned using the updated means as new centroids. These steps are repeated until convergence—i.e., when the cluster assignments no longer change between iterations.

A key limitation of K-Means is that the number of clusters k must be specified in advance. To address this, several analytical techniques, such as the elbow method or the silhouette score, can be employed to estimate an optimal value for k.

5.2.2 HDBSCAN

HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) [43] is an advanced **density-based clustering algorithm** that models clusters as dense regions of objects in the data space, separated by sparse regions. Unlike partitioning

methods such as K-Means, which assume spherical clusters and require the number of clusters to be predefined, HDBSCAN can discover clusters of arbitrary shapes and varying densities, and automatically determine the number of clusters.

HDBSCAN is an extension of the DBSCAN algorithm. In DBSCAN, clusters are formed around **core points**, defined as samples that have at least a minimum number of neighbors (specified by the parameter *min_samples*) within a given radius (parameter *eps*). This density threshold (*min_samples*) determines whether a point lies in a dense region of the data space. A cluster is therefore a set of core points, each close to each other, and a set of **border points** that are close to a core sample. Border points are non-core samples, but lie within the *eps*-neighborhood of a core point. Points that do not meet either condition—meaning they are not dense enough and are not reachable from any core point—are labelled as **outliers**.

The algorithm works by examining each point in the dataset and counting how many neighbors it has within its *eps*-radius, and consequently labelling it. This approach allows DBSCAN to identify clusters based on density, without requiring the number of clusters to be specified in advance.

HDBSCAN improves DBSCAN by eliminating the need to choose a specific *eps* value. Instead, it performs DBSCAN clustering over a range of *eps* values, creating a hierarchy of clusterings, and integrates the result to find a clustering that gives the best stability over *eps*. This allows HDBSCAN to find clusters of varying densities and be more robust to parameter selection.

5.2.3 Gaussian Mixture Model

Gaussian Mixture Model (GMM) [44] is a **probability-based clustering method** that models the dataset as a combination (or mixture) of multiple **Gaussian distributions**, each representing a different cluster. Unlike hard clustering methods such as K-Means, which assign each data point to a single cluster, GMM performs soft clustering, meaning that each point is assigned a probability of belonging to each cluster.

A Gaussian Mixture Model assumes that all data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters (mean and standard deviation), which are estimated during the clustering process. Given a dataset

X and the desired number of clusters k, the goal is to find the set of k probabilistic components that most likely explains the observed data. Formally, for each object x_i, the probability that it was generated from the *j-th* distribution is:

$$P(x_i) = \sum_{j=1}^{k} \pi_j \cdot P(x_i \mid \theta_j) \tag{5.11}$$

where π_j is the mixing coefficient, $\theta_j = (\mu_j, \sigma_j)$ are the parameters of the *j-th* Gaussian distribution, and $P(x_i \mid \theta_j)$ is the probability density of x_i under the Gaussian with parameters θ_j.

The overall task is therefore to estimate the parameters $\theta_1, ..., \theta_k$ of the Gaussians such that the likelihood of the data $P(X)$ is maximized.

To achieve this, the **Expectation-Maximization (EM)** algorithm is used. It starts by assigning random values to the parameters and then iteratively applies the following two steps until convergence. The Expectation (E) step computes the probability that each data point belongs to each Gaussian component based on the current parameter estimates:

$$P(\theta_j \mid x_i, \theta) = \frac{P(x_i \mid \theta_j)}{\sum_{l=1}^{k} P(x_i \mid \theta_l)} \tag{5.12}$$

The Maximization (M) step updates the parameters so that the expected log-likelihood $P(X)$ is maximized:

$$\mu_j = \frac{1}{k} \frac{\sum_{i=1}^{n} x_i P(\theta_j | x_i, \theta)}{\sum_{i=1}^{n} P(\theta_j | x_i, \theta)}, \tag{5.13}$$

$$\sigma_j = \sqrt{\frac{\sum_{i=1}^{n} P(\theta_j | x_i, \theta)(x_i - u_j)^2}{\sum_{i=1}^{n} P(\theta_j | x_i, \theta)}}. \tag{5.14}$$

Using probabilistic assignments, the Gaussian Mixture Model is capable of capturing elliptical cluster shapes and overlapping clusters, offering a more flexible modelling of the data compared to simpler clustering techniques.

Chapter 6

Cognitive Impairment Detection with Supervised Learning

Supervised machine learning techniques were used to predict cognitive impairment in murine models. The modelling phase, along with a rigorous evaluation of performance metrics, is crucial to ensure the development of generic models that perform well on unseen data.

6.1 Classification task definition

The main objective of the project is the development of a classifier capable of detecting cognitive impairment in murine subjects. Using the validated historical dataset derived from the Y-Maze test, the goal is to train a supervised learning model that can accurately assess cognitive decline based on behavioural features.

To this end, different machine learning algorithms were compared in order to identify the most reliable and accurate estimator. The primary purpose of the model is to correctly predict the class of new, unseen instances. Furthermore, another important aspect is to understand the contribution of each feature to the classification task. A classifier is basically an abstract representation of the relationship between the input features and the class label. Feature importance analysis plays a key role in this context, as it helps identify which variables are most relevant to distinguish between classes and offers in-

sights into the decision-making process of the model. This aspect is particularly crucial in a medical setting, where achieving high predictive performance must be accompanied by a clear understanding of the underlying reasons for each prediction, in order to support transparency, interpretability, and clinical decision-making.

This task is a multiclass classification problem, as the target feature can assume one of three categories—IMPAIRED, UNIMPAIRED, or MID—representing varying levels of impairment severity. In the analysis, the focus is specifically on the use of Label_80 as the target feature for model training and evaluation. This label represents the primary classification objective. The additional labels, Label_90 and Label_100, will be used at a later stage exclusively for validation purposes, in order to assess the generalizability and robustness of the classifiers trained through similar but distinct labelling thresholds.

As described in Chapter 3, the set of input attributes includes characteristics of the subject and behavioural performance metrics from the Y-Maze test. Since the class label was determined based on thresholds set on the % of Correct Alternations parameter, this attribute was excluded from the dataset because it is extremely correlated with the target feature.

6.2 Model training

The training phase is the systematic approach to learning a classification model given a training set. This process is known as induction and is also often described as "learning a model" or "building a model". In this study, several supervised learning algorithms were trained and compared: Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), and K-Nearest Neighbors (KNN). These models were selected for their interpretability, robustness, and popularity in classification tasks.

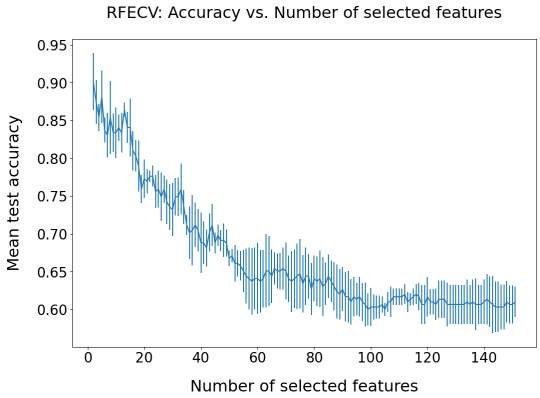
The preprocessing steps heavily affect the modelling phase. As previously delineated in Chapter 4, the training pipeline incorporates all necessary transformations prior to the application of the learning models. For each preprocessing step, multiple techniques were tested, and the most effective ones were selected based on the performance achieved by each algorithm. Specifically, feature encoding was consistently performed using one-hot-encoding for categorical variables across all models. For the normalization step, the

techniques discussed in Section 4.2.2 were evaluated. Moreover, feature selection was preferred over dimensionality reduction, as it allows a more interpretable understanding of the contribution of each feature to the classification process. Table 6.1 summarizes the optimal combination of preprocessing operations adopted for each algorithm.

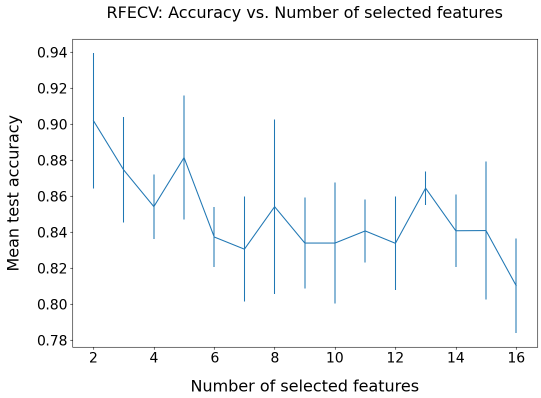
Model	**Normalization**	**Feature Selection**
LR	Z-score	RFE-CV (min 5)
SVM	Z-score	RFE-CV (min 5)
RF	Z-score	RFE-CV (min 2)
KNN	Z-score	RFE-CV (min 2)

Table 6.1: Preprocessing transformations selected for each model.

It can be seen that the same techniques were selected for each model, except for small variations in the initial settings. Z-score normalization was implemented through the scikit-learn's **StandardScaler** function [45]. For feature selection, **RFECV** [27] proved to be the most effective technique. This method was implemented using a *LogisticRegression* model as the supervised learning estimator. To optimize performance, different values of the *min_features_to_select* parameter were tested. This parameter defines the minimum number of features to be retained during the recursive elimination process and plays a crucial role in balancing model complexity and predictive performance.



(a) All features evaluated.



(b) Zoomed selection performance.

Figure 6.1: RFECV: impact of the number of selected features on accuracy.

As shown in the plot produced by the RFECV process (Figure 6.1), setting the *min_features_to_select* parameter to 2 leads to the selection of exactly two features, whereas setting it to 5 results in five selected features. The graph illustrates that increasing the number of input features tends to reduce the overall accuracy. Despite this general trend, model-specific performance varies: Logistic Regression and Support Vector Machine achieve better results when the minimum number of selected features is set to 5, suggesting a benefit from slightly richer input representations. In contrast, Random Forest and K-Nearest Neighbors perform better with just 2 features, likely due to their sensitivity to irrelevant or redundant inputs. More specifically, when *min_features_to_select* is set to 2, the selected features are **N° Entries Tot** and **N° of Alternations**. When the parameter is set to 5, the selected features are **N° Entries Tot**, **N° of Alternations**, and the exit counts for arms A, B, and C (**A/B/C: N° Exits**).

After defining the model-specific preprocessing pipeline, the next step involved **hyperparameter tuning**, a crucial phase to optimize the performance of the model. Hyperparameter tuning refers to the process of identifying and selecting the optimal configuration of hyperparameters of a machine learning model—parameters that are not learned directly during the training process but significantly influence the final performance of a model. To ensure optimal classification accuracy and model generalization, a grid search optimization was performed for each algorithm. This method exhaustively explores all possible combinations within the predefined hyperparameter space, evaluating each configuration through cross-validation. The tuning procedure was implemented through a custom function that supports both grid and randomized search strategies, although in this analysis the grid search approach was preferred for its exhaustive nature. Finally, the best-performing parameter set, as identified by the search, was used to train the final model instance.

The code for the hyperparameter tuning process can be found in Appendix B. The appendix also includes an example of how the preprocessing and tuning functions were applied in practice, specifically for the Support Vector Machine model.

At the end of this process, each algorithm yielded a trained model based on the optimal preprocessing steps and hyperparameter configuration.

6.3 Model evaluation and selection

The process of applying a classification model on unseen test instances to predict their class labels is known as deduction. The performance of a classifier can be evaluated by comparing the predicted labels with the true labels of the test instances. A model that is able to accurately predict the class labels of instances it has never encountered before has good generalization performance.

There are several metrics for assessing how good a classifier is at predicting class labels [46]. A commonly used measure is the **accuracy**, which represents the percentage of test set tuples correctly classified by the model. In the binary case, it is defined as:

$$\text{Acc} = \frac{TP + TN}{P + N}, \tag{6.1}$$

where TP (true positives) and TN (true negatives) refer to correctly classified positive and negative samples, respectively, and P and N are the total number of positive and negative instances.

Accuracy is also referred to as the overall recognition rate, indicating the model's general ability to assign the correct label across all instances. However, in the presence of imbalanced class distributions, accuracy alone may provide a misleading assessment of model performance. In such scenarios, a classifier may achieve high accuracy by simply predicting the majority class, while failing to correctly identify instances of the minority class. To address this issue, the **balanced accuracy** metric is used. It computes the average of recall values obtained for each class, thus giving equal weight to all classes regardless of their frequency. In the binary setting, it is defined as the arithmetic mean of sensitivity (true positive rate) and specificity (true negative rate):

$$\text{B-Acc} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right), \tag{6.2}$$

where FN and FP are false negatives and false positives, respectively.

An alternative measure to evaluate the model is the **F1 score**. The F1 score combines precision (the ability to avoid false positives) and recall (the ability to detect all true positives) into a single metric.

The precision and recall measures are defined as follows:

$$\text{precision} = \frac{TP}{TP + FP}, \qquad (6.3) \qquad \text{recall} = \frac{TP}{TP + FN} = \frac{TP}{P}. \qquad (6.4)$$

The F1 score is defined as the harmonic mean of precision and recall:

$$\text{F1} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}}. \qquad (6.5)$$

This metric is especially useful when the goal is to balance precision and recall, rather than prioritizing one over the other. It is well-suited for tasks where false negatives and false positives are equally undesirable, as in many medical or diagnostic settings.

In this study, accuracy, balanced accuracy, and F1 score were used as primary evaluation metrics to compare the predictive performance of the trained models. These metrics provide complementary perspectives and are particularly suitable for the multi-class, unbalanced classification task. Table 6.2 reports the evaluation metrics obtained by each trained model on the test set. It can be seen that the SVM achieved the best performance in all metrics, with scores of 1.0, indicating a perfect classification on the test set. The LR model also performed very well, with all three metrics close to 0.96. On the other hand, RF and KNN exhibited slightly lower performance, especially in terms of balanced accuracy, which is more sensitive to class imbalance.

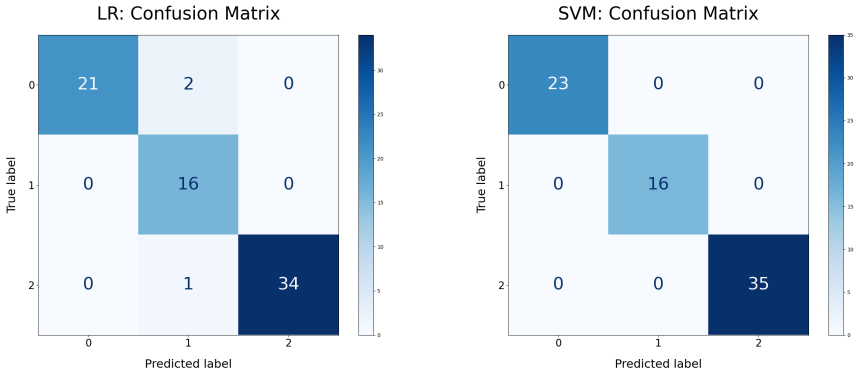
Model	**Acc**	**B-Acc**	**F1**
LR	0.96	0.96	0.96
SVM	1.0	1.0	1.0
RF	0.88	0.85	0.88
KNN	0.91	0.88	0.90

Table 6.2: Performance of trained models on test set.

In addition to metric-based evaluation, a **confusion matrix** was generated for each model to gain deeper insight into the classification behaviour. The confusion matrix

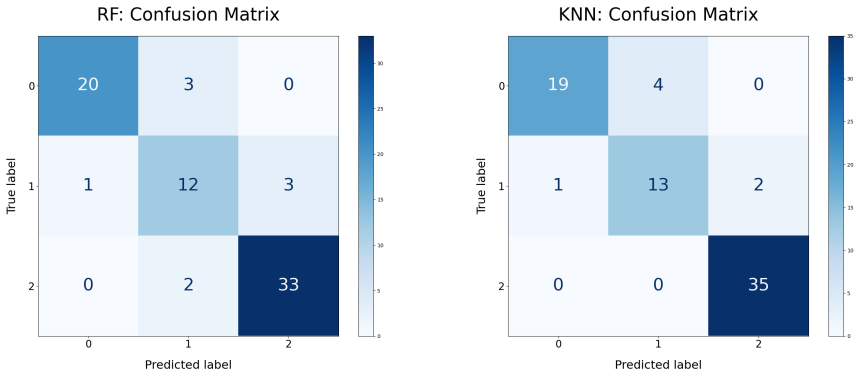
provides a detailed overview of the correct and incorrect predictions, showing how the predicted labels deviate from the true labels for each class. This representation is particularly useful to understand the types of errors the model makes and can reveal potential weaknesses that are not immediately apparent from summary metrics alone.

The confusion matrices of the trained models are shown in Figure 6.2. These matrices must be interpreted considering the label encoding transformation applied to the target variable, as summarised in Table 4.3, where the classes are encoded as follows: IMPAIRED = 0, MID = 1, and UNIMPAIRED = 2.



(a) Logistic Regression.

(b) Support Vector Machine.



(c) Random Forest.

(d) K-Nearest Neighbors.

Figure 6.2: Confusion matrices of trained models.

As expected, the matrix corresponding to the SVM model reveals perfect classification performance: all predictions fall on the main diagonal, indicating that every instance was

correctly assigned to its true class. This result is consistent with the previously reported metrics for the SVM model, which all reached the maximum value of 1.0.

Regarding the LR model, the confusion matrix highlights a total of three misclassifications. Specifically, the model incorrectly predicts the class MID for two instances that actually belong to the IMPAIRED class, and for one instance that truly belongs to the UNIMPAIRED class. These are probably samples with characteristics of both impaired and unimpaired subjects, which the model struggles to classify with certainty, opting instead for the intermediate MID label. Overall, the number of errors is very limited, and the model does not make critical errors, confirming its solid and reliable performance.

In contrast, the RF model makes a higher number of classification errors. Misclassifications involve the prediction of the MID class for instances that are actually IMPAIRED or UNIMPAIRED, as well as incorrect assignments of IMPAIRED or UNIMPAIRED to samples that truly belong to the MID class. These results suggest that the model has difficulty in accurately distinguishing borderline cases, often oscillating between the MID and the two extreme classes. However, the model never confuses IMPAIRED samples with UNIMPAIRED ones and vice versa, a desirable behaviour in medical contexts, where such misclassification could be critical.

A similar pattern can be observed in the KNN model, which also avoids directly confusing the IMPAIRED and UNIMPAIRED classes. Although the distribution of errors differs slightly, the majority of misclassifications still concern samples that are close to the decision boundaries, leading to a tendency to mislabel them as MID.

Based on the results obtained through metric evaluation and confusion matrix analysis, the models selected as the most effective are Support Vector Machine and Logistic Regression. The SVM model achieved perfect scores across all metrics and did not make classification errors, demonstrating exceptional performance and reliability. Similarly, the LR model showed high accuracy and balanced performance, with only a few minor misclassifications and no critical errors. These results indicate that both models are well-suited for the classification task, with SVM slightly outperforming LR. Therefore, these two models were selected as the best and further analysis and specific evaluations were subsequently carried out on them.

6.3.1 Decision threshold tuning

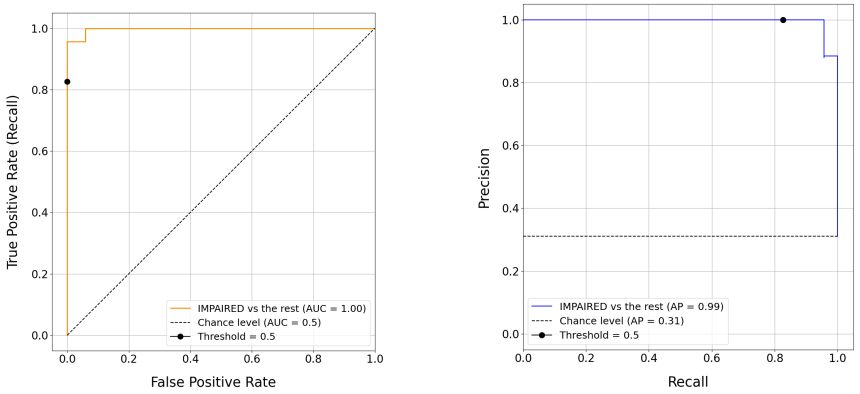
In the specific context of the project, the objective of the classifier is the detection of cognitive impairment. Therefore, it is particularly important to correctly detect subjects belonging to the IMPAIRED class. This consists in minimising the number of IMPAIRED subjects who are not correctly identified by the model, that is, reducing the number of false negatives (considering IMPAIRED as the positive class). In this sense, the **recall** for the IMPAIRED class becomes a key metric to consider during model evaluation. An additional analysis was carried out to explore the possibility of further improving the previously selected Logistic Regression model. The SVM model, having already achieved perfect performance, was excluded from this step.

This analysis was conducted by examining two fundamental graphs for this type of evaluation: the ROC curve and the Precision-Recall curve.

The **ROC curve** (Receiver Operating Characteristic curve) displays the trade-off between the True Positive Rate (TPR), also known as sensitivity or recall, and the False Positive Rate (FPR) across different decision thresholds. Specifically, TPR is the proportion of positive samples that are correctly labelled by the model, FPR is the proportion of negative tuples that are mislabelled as positive. A higher TPR and a lower FPR indicate better model performance and the area under the ROC curve is a measure of the accuracy of the model [46].

The **Precision-Recall curve** visualizes the trade-off between precision (the proportion of correctly identified positive subjects among all those predicted as positive) and recall (the proportion of correctly identified positive subjects among all actual positive cases) [47].

Since this analysis involves a multiclass classification task, the ROC curve and the PR curve were computed with the One-Vs-Rest scheme, specifically comparing the IMPAIRED class (the positive class of interest) against the other two classes (MID and UNIMPAIRED combined). Figure 6.3 shows the two generated curves, expressing the performance of the Logistic Regression model in distinguishing the IMPAIRED class from the rest.



(a) ROC curve.

(b) Precision-Recall curve.

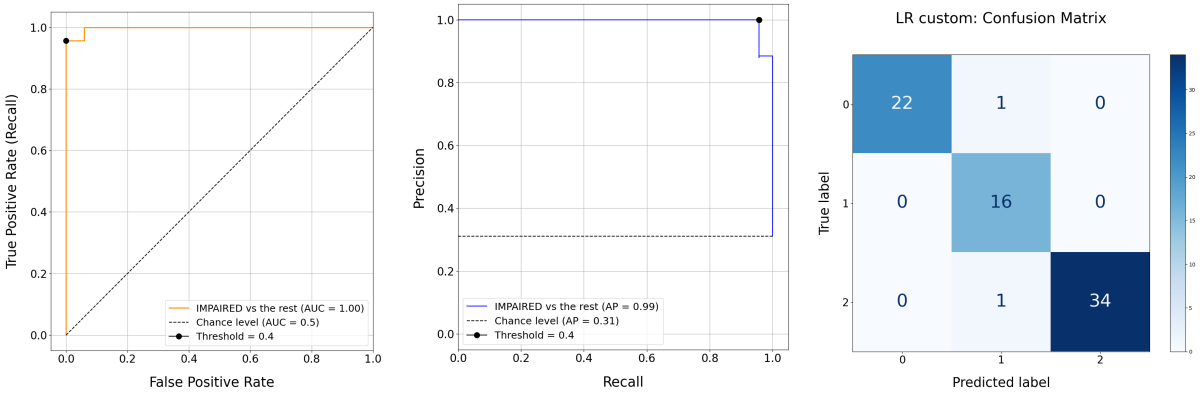
Figure 6.3: Discriminative performance of Logistic Regression.

The ROC curve bows towards the top-left corner of the plot, with an AUC (Area Under the Curve) of 1.00, indicating that the LR model performs very well as a classifier. Similarly, the Precision-Recall curve shows an average precision (AP) of 0.99, confirming that the model maintains high precision at various recall levels and significantly outperforms the chance level (AP = 0.31).

Both plots reflect strong model performance in identifying the IMPAIRED class, but also suggest that the recall could still be improved without compromising other performance metrics. This could be achieved by adjusting the decision threshold used for classification. In fact, both plots include a marker at the default threshold (0.5), visually highlighting the current trade-off between precision, recall, and false positive rate. These threshold indicators serve as valuable references for understanding how performance may vary with different decision boundaries, offering insights into how to further optimise the model's behaviour depending on specific application needs.

To address this, the decision threshold was subsequently tuned with the specific goal of maximising the recall metric. The selected threshold of 0.4 allowed the model to increase the recall without compromising performance in other metrics (see Figure 6.4). Reducing the threshold further would have totally eliminated the misclassification of IMPAIRED subjects as MID, but at the cost of introducing errors by incorrectly labelling MID samples as IMPAIRED, more than those avoided by adjusting the threshold. Therefore, a threshold of 0.4 was chosen as the best trade-off. It is important to note that

the few remaining errors involve misclassifying individuals as MID, a class that reflects clinical uncertainty and suggests the potential presence of cognitive decline. As such, these cases are still identified as critical, ensuring that the model remains effective in its primary objective of supporting early diagnosis.



(a) ROC curve. (b) Precision-Recall curve. (c) Confusion matrix.

Figure 6.4: Discriminative performance of customized Logistic Regression.

With this threshold adjustment, the Logistic Regression model achieved improved overall performance. Specifically, it reached an accuracy and an F1-score of 0.97 and a balanced accuracy of 0.98.

6.4 Model interpretation and comparative evaluation

Model interpretation consists in understanding the model decision-making process. This step is crucial to enable fairness, accountability, and transparency of a predictive estimator, giving humans enough confidence to use that model in real cases. However, there exists a typical trade-off between model performance and interpretability: simpler models, which are more intuitive and interpretable, tend to reach lower performance, especially in complex problems.

In this study, Logistic Regression and Support Vector Machine (with a linear kernel) were identified as the best-performing models for the classification task. Since they

are both linear algorithms that model the associations between features and the target linearly, their interpretation is fairly straightforward.

To further support the interpretability of the trained models, a feature importance analysis was carried out.

6.4.1 Feature importance assessment

Feature importance analysis is essential to understand the contribution of each input feature to the classification task, identifying the most relevant variables in class prediction.

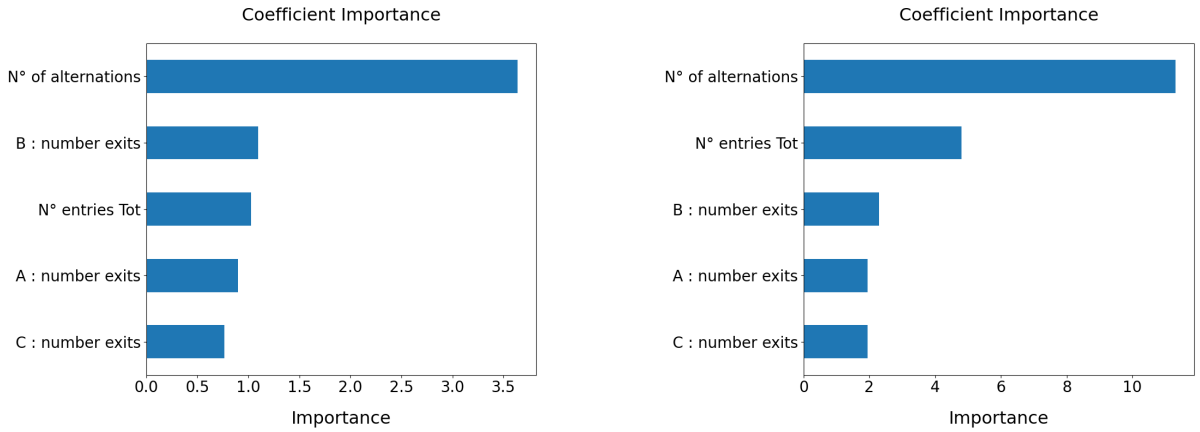
In linear models, one way to evaluate the importance of features in the predictive task consists in the analysis of the **coefficients** in the decision function. The coefficients represent the influence of each input feature on the model's output, assuming that all other variables remain constant. This method provides clear insights in linear models, but it is less applicable to non-linear models.

Another method to assess the feature importance in machine learning models is the **permutation importance** technique. The intuition is to measure the decrease in model performance when the values of a single feature are randomly shuffled, or permuted, across the dataset. The more the model's performance degrades after shuffling a feature, the more important that feature is considered to be. One key advantage of the permutation feature importance is that it is model-agnostic, i.e. it can be applied to any fitted estimator. Moreover, it can be calculated multiple times with different permutations of the feature, further providing a measure of the variance in the estimated feature importances for the specific trained model [48].

Both of these methods were used to assess the feature importance for the Logistic Regression and Support Vector Machine models.

Figure 6.5 shows the influence attributed to the input features by the estimators based on the coefficient analysis. For both models, the most influential feature is N° of Alternations, with substantially higher relative importance compared to all other input variables. This suggests that both the LR and the SVM rely heavily on this feature when

making classification decisions. Following this, the remaining features exhibit lower and relatively similar levels of importance. In particular, the SVM model attributes more importance to the N° Entries Tot feature, while the other three variables show almost identical and minimal influence. Conversely, the LR model appears to rely more on the feature B: N° Exits (the number of exits from arm B performed), highlighting a slight difference in how the two models weigh the supporting features in their predictions.

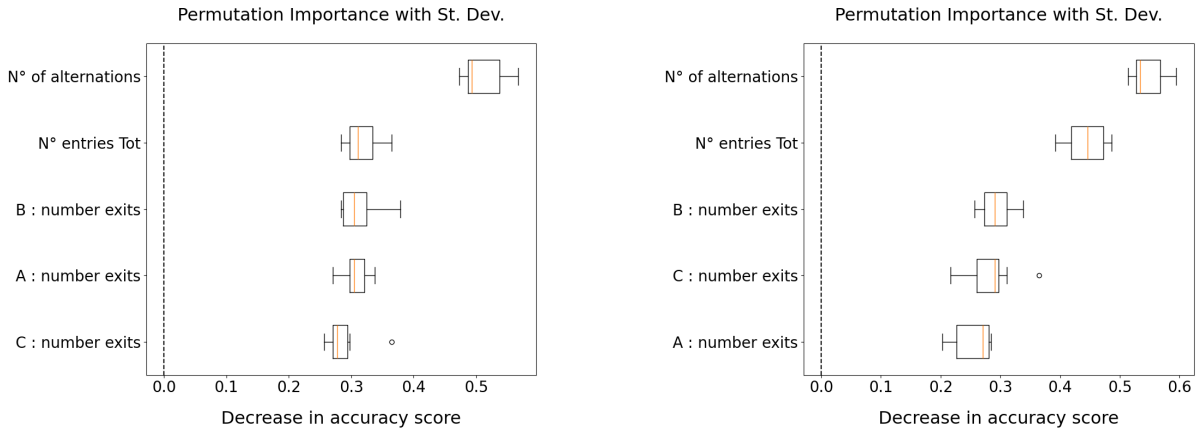


(a) Logistic Regression.

(b) Support Vector Machine.

Figure 6.5: Feature importance assessment through coefficient analysis.

Figure 6.6 illustrates the importance attributed to each feature using the permutation importance technique. For every feature, the plot reports an importance value along with the corresponding standard deviation, obtained through repeated permutations. This method confirms that N° of Alternations remains the most influential feature for both the SVM and LR models, followed by N° Entries Tot. The remaining three features are less relevant, with lower and similar importance values and overlapping ranges of variance. It is possible to note that, for the LR model, the two most important features differ between the two methods. This discrepancy can be attributed to the fact that coefficient-based analysis reflects the direct, linear relationship between features and the output, while permutation importance evaluates the actual impact of each feature on model performance, capturing interactions, correlations, or effects of regularization.



(a) Logistic Regression.

(b) Support Vector Machine.

Figure 6.6: Feature importance assessment through permutation analysis.

These findings highlight the consistency of **N° of Alternations** as the dominant predictive feature in both models and evaluation methods, confirming its central role in distinguishing between classes. This feature, combined with variables expressing the total number of entries performed and the number of entries in each arm of the maze, provides a consistent instrument for the detection of cognitive impairment.

The observed alignment between LR and SVM suggests that the classification task is being solved consistently. Overall, these analyses offer valuable insights into how the models operate and which variables drive their decisions, supporting model transparency and potentially guiding future feature engineering.

6.4.2 Label-based performance comparison

The modelling phase was initially carried out using Label_80 as the target feature, guiding the selection and tuning of the Logistic Regression and Support Vector Machine models. The additional labels present in the dataset, Label_90 and Label_100, were subsequently used. New models were trained following the same procedures and hyperparameter optimization strategies previously specified, using the different target variables. This analysis aims to assess whether the LR and SVM models, developed under the original labelling scheme, maintain high performance when trained with different label definitions. This evaluation provides insights into the stability of the selected

modelling approach across different interpretations of the classification task.

The performance of the Logistic Regression and Support Vector Machine models trained with the different target labels is reported in Table 6.3.

	Acc	B-Acc	F1
LR	0.97	0.98	0.97
SVM	1.00	1.00	1.00

(a) Label_80

	Acc	B-Acc	F1
LR	0.93	0.95	0.93
SVM	0.95	0.96	0.95

(b) Label_90

	Acc	B-Acc	F1
LR	0.91	0.95	0.92
SVM	0.85	0.92	0.88

(c) Label_100

Table 6.3: Performance comparison of LR and SVM across different label thresholds.

It is evident that model performance tends to decrease as the probability threshold used to define the target labels increases (from Label_80 to Label_100). A gradual decline in performance is observed for both models, more prominently for SVM. Specifically, for Label_90 both classifiers maintain strong predictive power, while with Label_100 the SVM model shows a more substantial drop, and the LR model maintains relatively stable performance.

This drop in performance may be partially attributed to the fact that both preprocessing strategies and modelling techniques were specifically optimized for the classification task using Label_80. However, the main limiting factor lies in the increasing class imbalance introduced by Label_90 and, even more so, by Label_100 (as shown in Table 3.3). As these alternative labels adopt stricter thresholds in defining the target class, the distribution of samples becomes increasingly skewed, resulting in a much smaller proportion of IMPAIRED class instances (total absent for Label_100). This imbalance makes it more difficult for models to effectively learn the distinguishing features of the minority class, even when balancing techniques such as class weighting are applied. Consequently, the classifiers tend to exhibit reduced performance under these conditions.

Despite the observed decrease in performance with these other target labels, the Logistic Regression and Support Vector Machine models still achieve consistently high results even when trained on Label_90 and Label_100. This suggests that the overall modelling pipeline—including encoding, normalization, feature selection, and hyper-

parameter tuning—is robust and well-suited to the task. The ability of both models to maintain strong performance across different label configurations indicates that the modelling strategies adopted are generalizable and effective, even under more challenging conditions such as increased class imbalance or stricter labelling criteria.

6.5 Robustness evaluation of model performance

Model evaluation is the process of assessing how well a machine learning model performs its task on previously unseen data. Typically, the original dataset is split into two subsets: a training set, used to learn the model, and a test set, used to evaluate its performance. However, relying on a single train/test split may lead to biased or overly optimistic performance estimates, especially if the split favours certain patterns in the data. To address this, it is crucial to assess the robustness of the model by evaluating its performance across multiple data partitions. This analysis aims to evaluate how consistently the model performs when trained and tested on different subsets of the data. Robustness evaluation provides insights into the model's ability to generalize beyond the specific conditions of a single experimental setup, reducing the risk of overestimating its effectiveness due to favourable data sampling.

To perform this robustness analysis on the selected classifiers, Logistic Regression and Support Vector Machine models, the **cross-validation** technique was used. The process repeatedly split the original dataset into multiple independent training and test sets. Specifically, the **Stratified Shuffle Split** method was employed to generate a specified number of random splits (10 splits) while preserving the original class distribution within each split. For each split, the models are trained on the training set and evaluated on the test set, which is a predefined fraction (20%) of the entire dataset. This process yields a **distribution** of accuracy scores, balanced accuracy scores, and f1 scores, reflecting the variability of the performance of each of the two models across different data partitions.

Based on this approach, the **mean value** for each metric across the different splits was calculated to provide a comprehensive summary of the models' overall performance. In addition, a **95% confidence interval** was computed around each mean value to quantify the uncertainty associated with these performance estimates.

The confidence interval represents a range of values within which the true model's performance is expected to be with a certain level of confidence (commonly 95%). This means that if the cross-validation procedure is repeated multiple times on different samples, the true accuracy would fall within this interval in approximately 95% of the cases. Calculating the confidence interval is crucial because it gives insight into the stability and reliability of the model's performance. A narrow confidence interval indicates consistent performance across different data splits, suggesting that the model generalizes well and its accuracy estimate is robust. In contrast, a wide confidence interval suggests a high variability of performance, which may indicate that the model is sensitive to the specific data used for training and testing, and therefore less reliable.

The results of the cross-validation analysis for the Logistic Regression model indicate a mean **accuracy** score of **0.89**, with a 95% confidence interval of [**0.86**, **0.92**]. In terms of **balanced accuracy**, the model achieved a mean score of **0.87**, with a corresponding confidence interval of [**0.83**, **0.91**]. The mean **F1** score was **0.89**, with a 95% confidence interval of [**0.86**, **0.92**].

These values were notably lower than the performance observed in the initial single train/test split, where the accuracy, balanced accuracy, and F1 score were **0.97**, **0.98**, and **0.97**, respectively. This suggests that the initial results may have overestimated the model's true generalization performance.

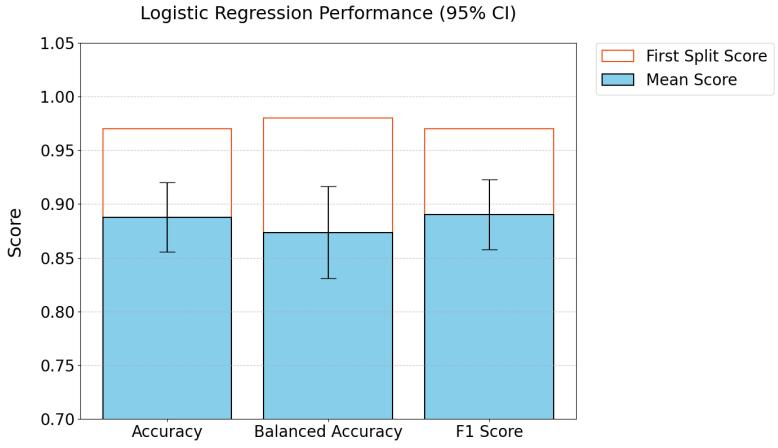


Figure 6.7: LR performance with 95% confidence interval.

The Support Vector Machine model consistently outperformed Logistic Regression across all metrics. The mean **accuracy** obtained through cross-validation was **0.95**, with a 95% confidence interval of [**0.93**, **0.97**]. The mean **balanced accuracy** reached **0.95** with a confidence interval of [**0.93**, **0.97**], and the mean **F1** score was also **0.95**, with the same confidence bounds.

As the Logistic Regression model, these scores were slightly lower than those obtained from the initial single split evaluation, where accuracy, balanced accuracy, and F1 score were all equal to **1.0**. This again reflects the tendency of single-split evaluations to produce overly optimistic performance estimates.

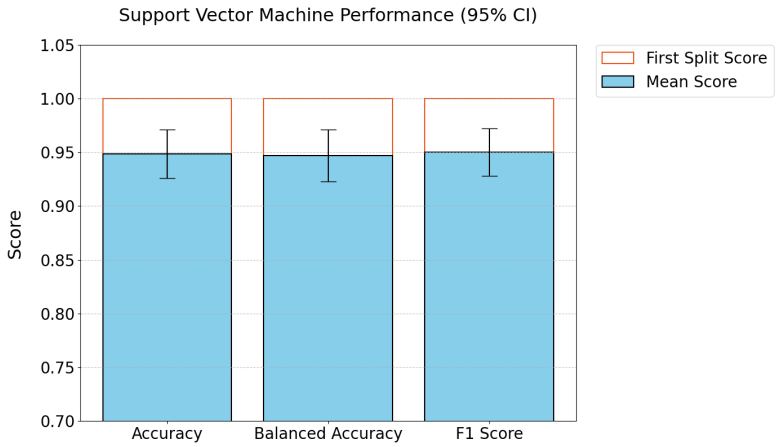


Figure 6.8: SVM performance with 95% confidence interval.

The results of the cross-validation analysis differ notably from the initial scores obtained from a single train/test split, for both the two models. The initial scores were likely overly optimistic, as they reflected performance on a single, specific partition of the data that may have favoured the models. Another possible reason for the higher initial performance values is that the hyperparameter tuning for each model was performed specifically on the initial training set used in the single train/test split. In contrast, the cross-validation approach provides a more realistic and robust estimate by averaging performance across multiple splits.

By comparing the two models, the Support Vector Machine demonstrated higher and more stable performance across all evaluation metrics. Although both models showed a

drop in performance when evaluated using cross-validation rather than a single split, the drop was less pronounced for the SVM, suggesting better generalizability.

Moreover, the confidence intervals across all three metrics—accuracy, balanced accuracy, and F1 score—reveal the variability in model performance. For the Support Vector Machine, the consistently narrower intervals indicate greater stability and less variability across different data partitions. In contrast, the Logistic Regression model shows wider confidence intervals, reflecting more fluctuation in its performance. This suggests that SVM not only achieves higher average scores, but also generalizes more reliably across different subsets of data for all evaluation metrics considered.

Overall, these results reinforce the importance of evaluating model performance through multiple splits and considering confidence intervals, rather than relying on a single train/test split, to better understand the true predictive capability and robustness of machine learning models.

Despite the lower metrics estimates obtained through cross-validation, both models continue to demonstrate excellent performance in different data partitions, confirming their reliability and suitability as effective predictive tools for cognitive impairment.

Chapter 7

Exploratory Analysis with Unsupervised Techniques

Cluster analysis was performed to explore the partitioning of the dataset into groups of similar data. The characteristics of the clusters were compared with those of the predefined target categories, trying to uncover novel patterns and identify alternative factors of cognitive impairment.

7.1 Unsupervised analysis objectives

In this project, the goal of the unsupervised analysis is to uncover predictive factors of cognitive impairment without relying on the predefined labelling approach. The current labelling method focuses on detecting cognitive decline in murine models through the parameter % of Correct Alternations of the Y-Maze test. However, the dataset contains a wide range of other parameters that have not yet been used in the classification of subjects. This unsupervised analysis aims to explore these additional features, potentially uncovering new factors or relationships that could improve the understanding of cognitive impairment.

By employing clustering methods, the aim is to identify groups of data that share similar characteristics and are dissimilar to those of other groups. Using these automated algorithms, partitioning can lead to the discovery of previously unknown groups within

the data and even reveal hidden structures that extend beyond traditional cognitive impairment detection techniques. The use of an unsupervised approach provides the advantage of exploring relationships in the data without being constrained by predefined labels, allowing for an exploration of the data's inherent structure.

Specifically, the analysis focuses on comparing the clusters identified by the unsupervised models with the predefined target categories, aiming to understand the distinctive characteristics of each group. By examining the correlation between clusters and predefined classes, the study seeks to discover meaningful patterns and novel predictive factors of cognitive impairment.

As this is an unsupervised task, the target labels (Label_80, Label_90, and Label_100) were excluded from the analysis, allowing the clustering methods to operate solely on the available input features. All other parameters within the dataset were retained without any selection.

7.2 Implementation of clustering methods

Cluster analysis is the process of partitioning a set of data objects into subsets, called clusters, such that data objects within the same cluster are more similar to each other than to those in different clusters. Since clustering algorithms adopt different strategies and assumptions to group data, they may produce varying results even when applied to the same dataset. For this reason, multiple clustering techniques were employed in this study: K-Means, HDBSCAN, and Gaussian Mixture Model (GMM). The results of these methods were compared to evaluate the consistency and interpretability of the resulting clusters.

Some preprocessing steps are essential to ensure meaningful and reliable clustering results. As described in Chapter 4, the preprocessing pipeline for the unsupervised analysis includes the encoding of categorical features and the normalization of numerical variables. These transformations are fundamental for allowing clustering algorithms to operate effectively and to prevent the results from being biased by differences in the value ranges of features. Similarly to the approach used in the supervised analysis, categorical variables were encoded using one-hot-encoding, while numerical features were

standardized using scikit-learn's StandardScaler, ensuring that all variables contribute equally to distance-based computations.

Moreover, each clustering algorithm requires a specific modelling approach, with careful tuning of key parameters that can significantly influence the quality and structure of the resulting clusters. For this reason, a dedicated analysis was conducted for each method to define the most suitable parameter settings, taking into account both the nature of the data and the algorithm-specific assumptions. These preliminary steps were essential to ensure that each model could produce meaningful and interpretable groupings.

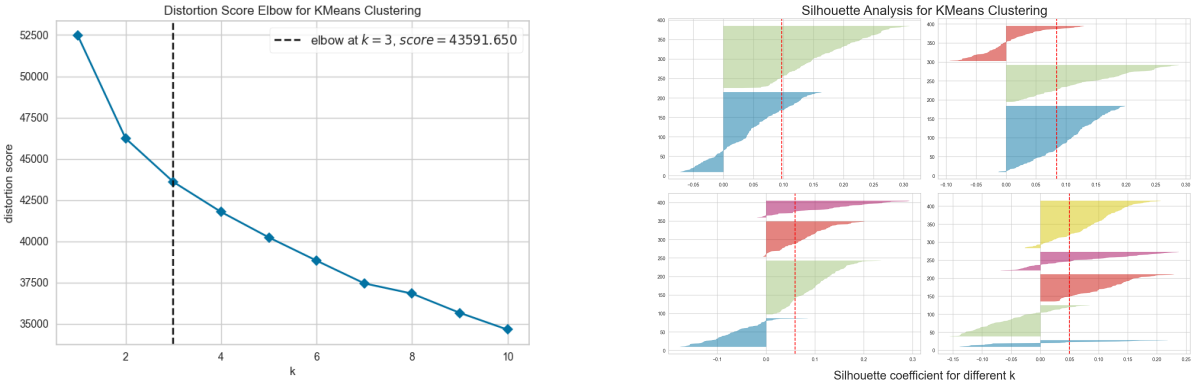
Among the clustering techniques considered, **K-Means** was the first to be applied. As described in Section 5.2.1, it is a centroid-based technique that assigns an object to a cluster based on the Euclidean distance between the object itself and the centroid of the cluster. Since the clustering results may depend on the initial random selection of cluster centres, it is good practice to run the algorithm multiple times with different initializations. To address this aspect, the *init* hyperparameter of the scikit-learn's KMeans class [42] was set to "k-means++", which selects initial cluster centroids improving convergence, and the *n_init* hyperparameter was set to 10, ensuring that the algorithm runs with different centroid seeds.

A central aspect of the K-Means algorithm is that the user has to specify the number of clusters k in advance. Determining the "right" number of clusters in the dataset is essential, but it is a difficult task. To identify the optimal value of k, two well-established techniques were employed: the Elbow Method and the Silhouette Analysis.

The **Elbow Method** involves plotting the inertia (i.e., the within-cluster variance) for a range of k values and identifying the point where the rate of decrease slows down considerably. This turning point in the curve, which looks like an "elbow" in the plot, is considered a good estimate of the optimal number of clusters [49].

The **Silhouette Analysis** evaluates how well each data point fits within its assigned cluster compared to other clusters. It provides a silhouette coefficient ranging from -1 to 1, with higher values indicating better-defined and more distinct clusters. A good value of k have all clusters with scores above the dataset's average, and their sizes are balanced [50].

Figure 7.1 displays the results obtained from both the Elbow Method and the Silhouette Analysis. The Elbow Method suggests k=3 as a potential optimal value. However, the absence of a clear inflection point in the plot limits the strength of this indication. In contrast, the Silhouette Analysis shows a progressive decrease in the average Silhouette score as the number of clusters increases. The most balanced configuration appears to be at k=2, where both the average Silhouette score and the consistency across clusters are more favourable. At k=3, although cluster sizes remain relatively balanced, one of the clusters exhibits a high number of samples with negative Silhouette scores, suggesting poor assignment. Further increasing the number of clusters results in highly imbalanced groupings and a general drop in clustering quality, as reflected by lower Silhouette scores across many samples.



(a) Elbow Method.

(b) Silhouette Analysis.

Figure 7.1: Methods to identify the optimal value of k in K-Means clustering.

Thus, both k=2 and k=3 were tested in K-Means Clustering. The results obtained are presented in the following section.

Another clustering technique used is **HDBSCAN**. Unlike K-Means, HDBSCAN is a density-based method that automatically determines the number of clusters, discovering clusters of arbitrary shapes and varying densities. However, also this algorithm has some hyperparameters that need to be properly tuned. Specifically, it is important to focus on the *min_samples* hyperparameter, which refers to the number of samples in a neighbourhood for a point to be considered as a core point, and on the *min_cluster_size*

hyperparameter, that is the minimum number of samples in a group for that group to be considered a cluster. The tuning of these hyperparameters must be performed with the aim of minimizing the number of outliers, obtaining a coherent number of clusters with a high probability of sample assignment to clusters, and avoiding unbalanced clusters. The results obtained from the tuning are reported in Table 7.1.

min_cluster_size	*min_samples*	**Clusters**	**Outliers**	**Cluster sizes**	**Min prob.**
3	default	4	264	81, 15, 3, 3	0.93
5	default	2	265	88, 13	0.85
6	default	2	286	74, 6	0.87
7	default	0	366	-	-
5	2	4	327	96, 23, 5, 5	0.91
5	3	2	243	108, 15	0.82
5	4	2	277	80, 9	0.86
5	15	0	366	-	-

Table 7.1: HDBSCAN hyperparameter tuning results.

The results highlight some limitations in the direct application of HDBSCAN to the original dataset. Although the algorithm shows a high minimum probability of assigning samples to clusters, indicating good confidence in its predictions, most samples are classified as outliers. Additionally, the identified clusters tend to be heavily unbalanced in size, with a large cluster and several very small ones.

While HDBSCAN did a great job on the data it could cluster, it did a poor job of actually managing to cluster the data. The problem is that, as a density-based clustering algorithm, HDBSCAN tends to suffer from the curse of dimensionality: high-dimensional data require more observed samples to produce much density.

To address this, a dimensionality reduction step was introduced prior to clustering. While PCA is a common technique, its linear nature makes it less effective in preserving the complex, non-linear relationships that may exist in the data. Instead, UMAP was

used to reduce the dimensionality while preserving both global and local structures.

The application of HDBSCAN after reducing the dimensionality with UMAP led to a significant improvement in clustering performance. In particular, the number of detected outliers decreased notably in several configurations, and the silhouette scores increased, indicating better-defined cluster boundaries. However, despite these improvements, the clustering results still suffered from a high number of small and fragmented clusters, and the cluster assignments often lacked interpretability and coherence from a domain-specific perspective.

Given these results, a further preprocessing step was introduced: manual feature selection, retaining only the "general" features of the Y-maze test. The complete preprocessing pipeline thus included: manual feature selection, feature encoding, normalization, and UMAP-based dimensionality reduction. This refined pipeline aimed to improve both the stability and interpretability of the final clustering outcomes.

The best parameter configuration identified is presented in Table 7.2. While the clusters are slightly unbalanced, they are nonetheless well-separated, as indicated by the silhouette score (0.52), and the minimum assignment probability confirms a generally confident assignment of samples to clusters.

min_cluster_size	*min_samples*	**Clusters**	**Outliers**	**Cluster sizes**	**Min prob.**
10	15	3	48	172, 140, 20	0.77

Table 7.2: HDBSCAN final configuration.

The interpretation of the clusters identified by HDBSCAN is provided in the following section, as for the analysis of the results of the other clustering methods.

The last clustering technique employed is **Gaussian Mixture Model**. To implement this probability-based method, a grid search was performed to identify the optimal configuration of the hyperparameters. Specifically, the number of mixture components *n_components* and the covariance type *covariance_type* were tuned. The Bayesian Information Criterion (BIC) was employed as the scoring metric, aiming to minimize model complexity while ensuring a good fit.

The best configuration identified for the Gaussian Mixture Model through grid search optimization has *n_components*=3 and *covariance_type*="spherical". The choice of three components suggests that the underlying structure of the data can be effectively described by three distinct probabilistic distributions, which means that the model attempts to identify three clusters. The use of spherical covariance indicates that the clusters are assumed to be isotropic, i.e., having the same variance in all directions, while still allowing each cluster to have its own overall variance.

7.3 Clustering results and interpretation

The analysis of the clustering results focused on identifying the distinctive characteristics of the groups found, with the aim of attributing meaningful interpretations to each cluster.

In the context of the Y-Maze test data, each data point represents a mouse's behavioural profile across multiple features, and the clustering process groups together mice with similar patterns of performance. Here, similarity is defined in terms of distance in the multidimensional feature space: mice whose feature vectors are closer together are considered behaviourally more similar, while greater distances indicate more divergent behavioural profiles.

To investigate the factors driving this separation, the distributions of individual features across clusters were examined using both visual and statistical tools. Feature distribution plots provided an initial insight into which variables differed the most between clusters, while the statistical *t-test* was used to identify features whose mean values differed significantly between groups. This combination allowed for a rigorous interpretation of the dimensions that contribute to the clustering structure.

Furthermore, the results of the clustering methods were compared with the predefined target categories to assess the alignment between data-driven groupings and established classification criteria (the original labelling method). This comparison involved analysing whether the distinctive features of the clusters corresponded to those characterizing the predefined classes, as well as visually comparing the spatial distribution of clusters and original labels through graphical representations. For this purpose, cluster visualization

was carried out by applying UMAP for dimensionality reduction, projecting the data into a two-dimensional space suitable for plotting.

The analysis starts with the exploration of **K-Means** results. Initially, K-Means clustering was performed setting the number of clusters to k=2.

As can be found in Figure 7.2, the two clusters identified by the algorithm differ primarily in the following features: N° of Alternations, N° Entries Tot, Distance, and Mean Speed. Cluster 0 includes data points with lower values for these features, while Cluster 1 shows consistently higher values. It is almost possible to find a threshold value for these characteristics, which distinguishes the two clusters.

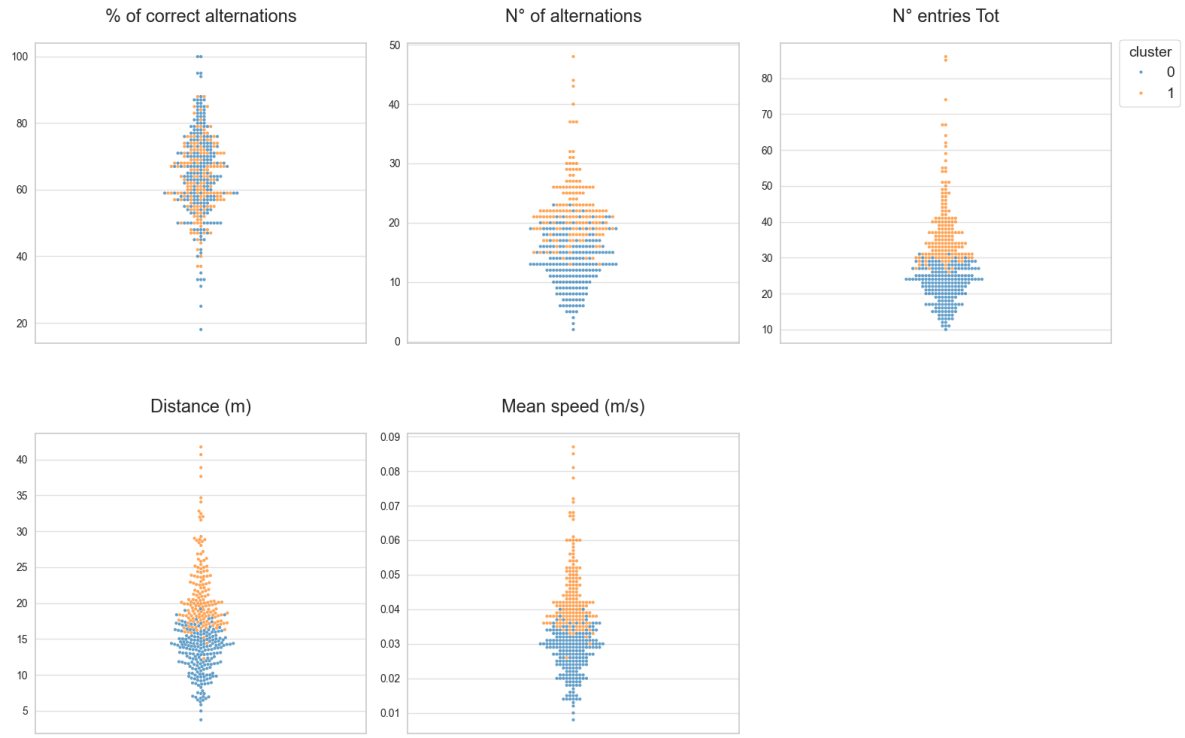


Figure 7.2: Distributions of selected features by K-Means clusters.

These features are commonly associated with general locomotor activity and exploratory behaviour. Mice in Cluster 1, exhibiting higher movement and entry rates, could be interpreted as displaying increased exploratory drive or hyperactivity, while those in Cluster 0 might reflect reduced activity levels, potentially due to apathy, motor impairments, or cognitive decline-related disengagement.

In addition to the primary features, additional variables—such as those related to entries into individual arms or specific movement sequences—also showed significant differences between clusters, as highlighted by the results of the *t-test* analysis. However, the clusters do not show significant separation with respect to the parameter % of Correct Alternations, the main indicator used in the literature to assess the presence of cognitive impairment.

In Figure 7.3, the distribution of the key features is shown with respect to the original Label_80 classification. The most distinctive feature that separates these predefined classes is the % of Correct Alternations. In contrast, features such as the number of alternations, total entries, distance travelled, and mean speed—that mainly define the K-Means clusters—do not show any notable differentiation between the original labels.

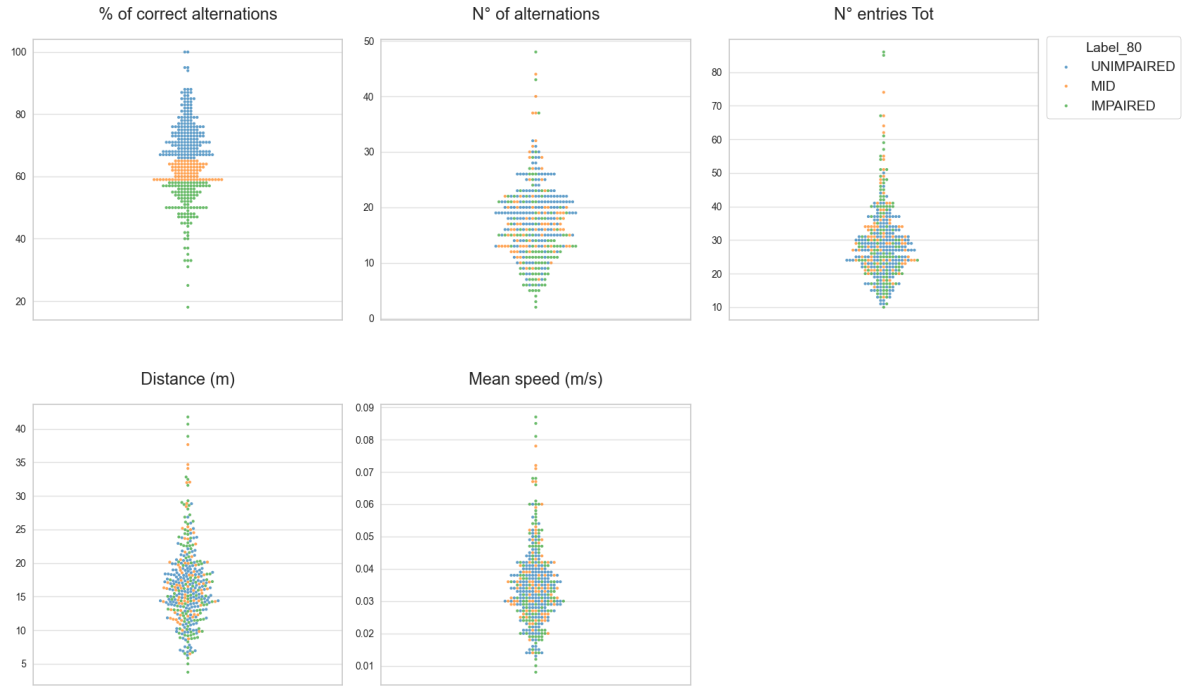
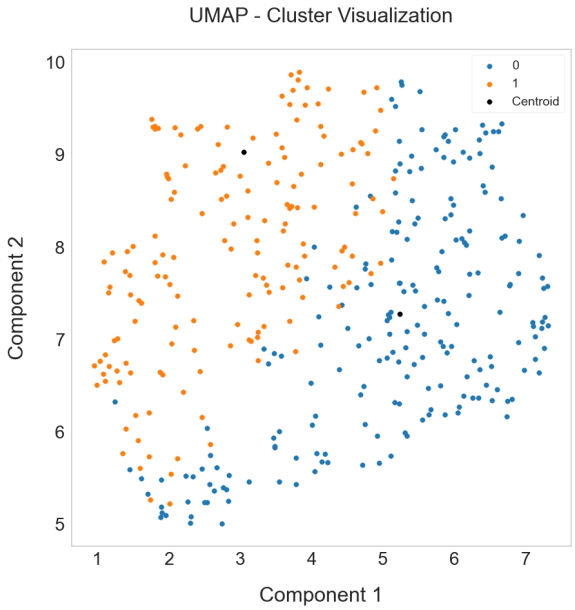


Figure 7.3: Distributions of selected features by predefined classes.

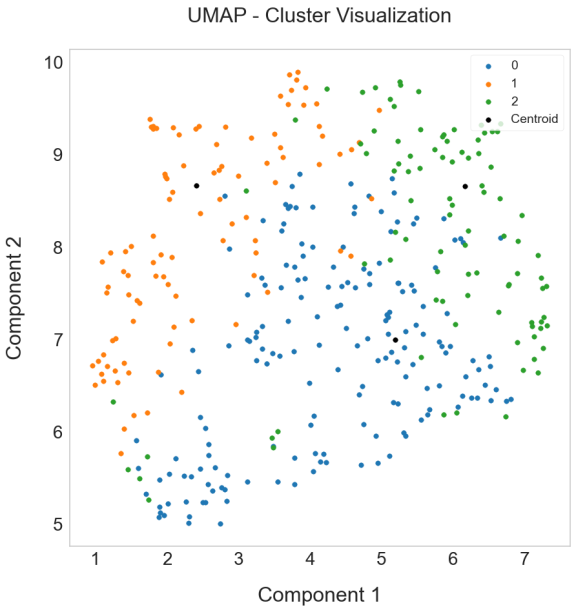
This discrepancy reveals a fundamental difference in what the two approaches (unsupervised clustering vs. predefined labelling) are capturing. The clusters found reflect behavioural patterns related to locomotor activity and exploration, but do not differ in cognitive performance as typically measured by alternation accuracy. This first result

highlights a discrepancy between the discovered clusters and the original class labels.

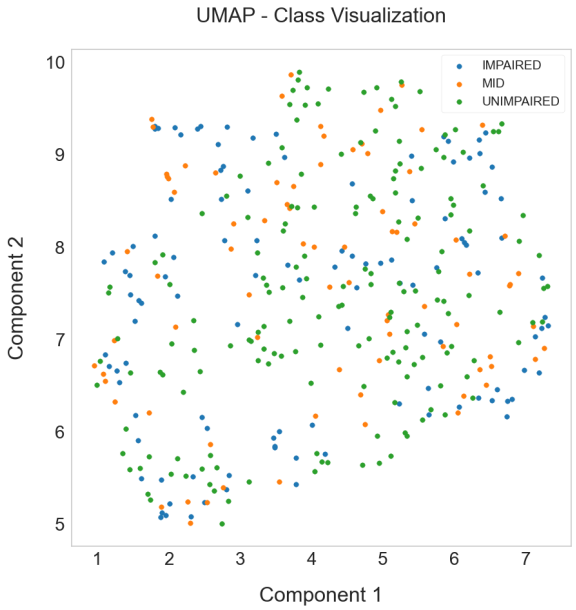
Subsequently, the spatial distribution of the clusters was compared to that of the original classes.



(a) K-Means clustering (k=2).



(b) K-Means clustering (k=3).



(c) Predefined classes (Label_80).

Figure 7.4: UMAP data visualization: K-Means clustering.

As shown in Figure 7.4a, the two clusters identified by K-Means appear well separated in space, indicating a successful partition into distinct groups. In contrast, the points associated with the original classes (Figure 7.4c) are more scattered and show considerable overlap.

This visual evidence reinforces the notion that the clusters extracted through unsupervised learning do not correspond to the predefined categories.

K-Means clustering was tested also setting k=3. The features that most differentiate the clusters remain the same as before, with the clustering identifying a group with high average values, one with low values, and one with intermediate values. However, the % of Correct Alternations continues to be irrelevant in the separation of the clusters.

Furthermore, when comparing the plot of the clusters (Figure 7.4b) with those of the original labels (Figure 7.4c), it becomes evident that the cluster distribution is organized differently compared to the distribution of the predefined categories.

This further highlights a mismatch between the natural grouping suggested by K-Means and the original class labels.

The clustering exploration continues with the analysis of the **HDBSCAN** results, obtained from the selected configuration of the algorithm (as reported in Table 7.2).

Figure 7.5 shows the distribution of the analysed features with respect to the clusters identified by HDBSCAN (Cluster -1 refers to the category of outliers).

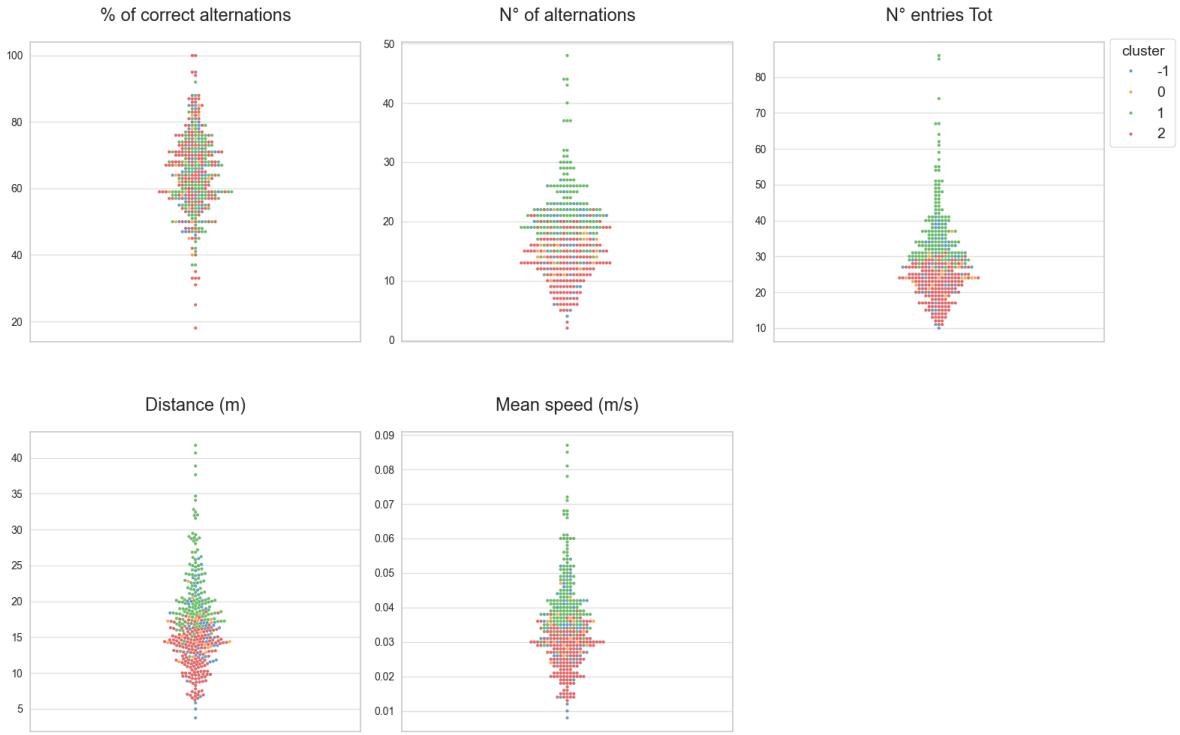
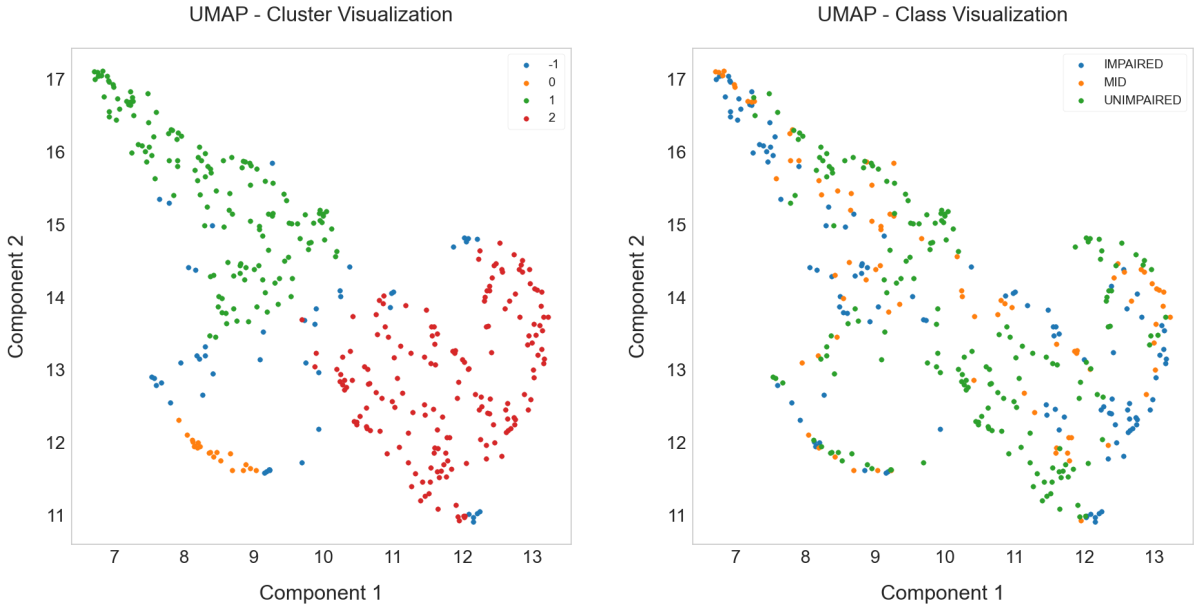


Figure 7.5: Distributions of selected features by HDBSCAN clusters.

Consistent with the findings from K-Means, HDBSCAN reveals a separation primarily driven by activity-related features, such as the number of alternations, total entries, distance travelled, and mean speed. Once again, the % of Correct Alternations does not emerge as a distinguishing variable between the clusters, suggesting that the unsupervised partitioning remains focused on behavioural rather than cognitive differentiation.

In Figure 7.6, the spatial distribution of the HDBSCAN clusters obtained and that of the original categories can be seen. This 2D visualization shows a different spatial projection of the data points compared to Figure 7.4, due to the additional feature selection step performed during the HDBSCAN clustering process.



(a) HDBSCAN clustering.

(b) Predefined classes (Label_80).

Figure 7.6: UMAP data visualization: HDBSCAN clustering.

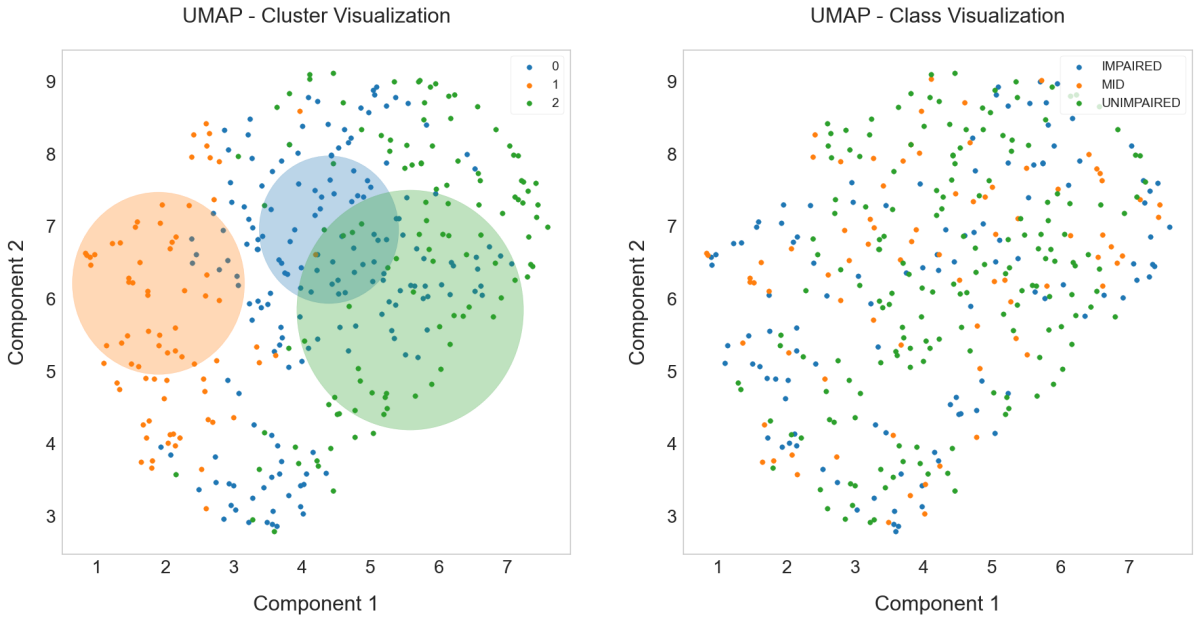
Once again, no clear correspondence can be observed between the clusters and the original predefined classes. The clusters are composed of data points that are spatially close to each other, forming compact and well-separated groupings in the low-dimensional projection space. In contrast, the original classes appear more dispersed and overlapping, lacking the internal cohesion that characterises the clusters.

This visual and structural discrepancy reinforces the idea that the clustering algorithms are capturing patterns in the data that are not aligned with the predefined classification scheme.

Lastly, **GMM** clustering results are reported. The outcomes are consistent with those observed using the other clustering methods, emphasizing consistent differences between the same core features. As with the previous methods, the % of Correct Alternations continues to show limited discriminative power, reinforcing the observation that spontaneous clustering is not associated with traditional measures of cognitive performance in the Y-Maze.

Figure 7.7 shows the spatial representation of the clusters obtained using the Gaussian Mixture Model clustering algorithm. In the plot, the ellipses represent the probability distributions estimated by the GMM: each ellipse is centred at the mean of a cluster and shaped according to its covariance matrix, which reflects the spread and orientation of the data points around the centre. This graphical representation provides an intuitive visualization of the structure and uncertainty within each cluster.

Notably, the figure also highlights a clear visual distinction between the GMM clusters and the original predefined classes.



(a) GMM clustering.

(b) Predefined classes (Label_80).

Figure 7.7: UMAP data visualization: GMM clustering.

The results obtained from the different clustering techniques (K-Means, HDBSCAN, and GMM) show a high degree of consistency, which supports the robustness and reliability of the identified groupings. This convergence across methods reinforces the idea that the clusters reflect genuine structure within the data, rather than artefacts of a particular algorithm.

However, the unsupervised clusters do not correspond to the original predefined classes, which are based on cognitive performance measures such as the % of Correct Alternations. This discrepancy suggests that clustering methods are capturing behavioural variability—particularly in locomotor activity and exploratory patterns—rather than cognitive impairment as traditionally defined.

Therefore, it is not possible to use these clustering approaches to explore alternative patterns or factors of cognitive impairment that align with the subject classification method currently considered correct.

Chapter 8

Conclusions

The cognitive impairment classification models obtained are the first outcome of the AI4ChemoBrain project. Their high performance makes them useful tools in the subsequent stages of the research. From the specific historical dataset used, no alternative factors of cognitive impairment emerged compared to those defined in the literature.

8.1 Discussion of results

This research is part of the AI4ChemoBrain project that aims to generate a tool for clinical use to predict the onset of chemobrain, namely the cognitive impairment experienced by patients treated with systemically administered anticancer therapies. As in any study of human pathological conditions, it is necessary to validate potential diagnostic or therapeutic tools using animal models.

Therefore, this study focused on the development of a predictive model of cognitive impairment in murine models, based on a validated historical dataset containing behavioural measures from the Y-Maze cognitive test. This classifier constitutes a first useful tool in the overall context of the project.

As described in the previous chapters, different machine learning models were trained for this task, with Logistic Regression and Support Vector Machine classifiers ultimately selected as the best predictors. These models achieved high predictive performance, highlighting the effectiveness of the preprocessing steps and the modelling process ap-

plied. Specifically, both models exceeded 95% accuracy when trained and evaluated using Label_80, which was adopted as the primary target label for model development. Furthermore, they consistently achieved strong results even when applied to alternative labelling strategies. Since model performance varies slightly depending on the label used, both classifiers were retained as valid outcomes, allowing the most appropriate model to be selected according to the specific labelling strategy or classification goal.

Both selected models are linear, which aligns with the relatively simple nature of the data and the classification task. The models base their predictions on a limited set of key features: Nº of Alternations, Nº Entries Tot, and A/B/C: Nº Exits. Specifically, subjects exhibiting a high proportion of alternations relative to the total number of entries are considered UNIMPAIRED, while a lower proportion indicates IMPAIRED cognitive performance. It is important to note that these variables must be evaluated in combination, as none of them alone provides a meaningful indicator of cognitive status. The number of exits from each individual arm adds further information, providing insight into the distribution of spatial exploration.

In the context of the supervised cognitive impairment detection task—defined according to the predefined labels—additional more specific parameters available in the dataset proved to be irrelevant for improving classification performance.

Basically, the classifiers developed are machine learning models that reproduce the labelling strategy adopted by domain experts, using a combination of original features obtained from the Y-Maze test. This alignment further supports the reliability and interpretability of predictive systems.

In addition, the study has an exploratory component aimed at investigating whether a larger set of behavioural features derived from the Y-Maze test could serve as a potential indicator of cognitive impairment. The dataset includes 196 attributes—quantitative measures extracted from the Y-Maze test—which remain largely unexplored in the existing literature.

To this end, unsupervised learning algorithms were applied to evaluate the intrinsic structure of the data and to assess whether meaningful groupings could emerge independently of predefined labels.

Although clustering techniques were able to identify groups of subjects with similar

behavioural profiles, these clusters did not align with the established cognitive impairment categories. This discrepancy suggests that the additional features may not directly correlate with the expert-defined labelling criteria. In fact, many of these parameters represent highly specific details of the animals' movement patterns during the test—such as fine-grained trajectory metrics—which may be too granular to provide additional discriminative power. Moreover, these detailed attributes reflect behavioural patterns already captured by the more general and informative parameters traditionally used in cognitive assessment, such as alternation rates and total entries.

Consequently, clustering approaches do not actually provide an effective means of identifying alternative factors or latent dimensions of cognitive impairment consistent with the current classification framework.

8.2 Statistical significance of classifier performance

The cognitive impairment classifiers are the main outcome of the study. To ensure the reliability and validity of these models, it is essential to demonstrate that their performance does not result from random chance, but rather reflects a meaningful and generalizable relationship between input features and class labels.

To assess this, a statistical hypothesis testing framework was employed [51]. Hypothesis testing is a statistical procedure used to evaluate whether the results of a study provide sufficient evidence to reject a default hypothesis, also called the null hypothesis, in favour of an alternative hypothesis.

In the specific context of validating the cognitive impairment classifiers, the goal of the statistical test is to determine whether the performance of the classifiers on the test set provides sufficient evidence to reject the null hypothesis—that the observed accuracy could be explained by random chance alone—in favour of the alternative hypothesis, which suggests that the classifiers capture a true, generalizable pattern in the data.

Formally, the **null hypothesis** (H_0) is: *"The classifier is not able to learn a generalizable relationship between the input features and the class labels from the given training set."* The **alternative hypothesis** (H_1) is: *"The classifier learns a generalizable relationship between the input features and the class labels from the training set."*

The statistical testing process follows a well-defined series of steps:

1. **Selection of the test statistic**: To evaluate model performance on the test set, a suitable test statistic is selected—in this case, the classification accuracy.

2. **Estimation of the null distribution**: To approximate the distribution of the test statistic under the null hypothesis, a permutation-based approach is employed. This involves generating synthetic training datasets by randomly shuffling the class labels multiple times. A classifier is trained on each permuted dataset, and its accuracy on the original test set is computed. This process yields a null distribution of accuracies, representing the expected performance of the classifier under the assumption that no true relationship exists between the input features and the class labels.

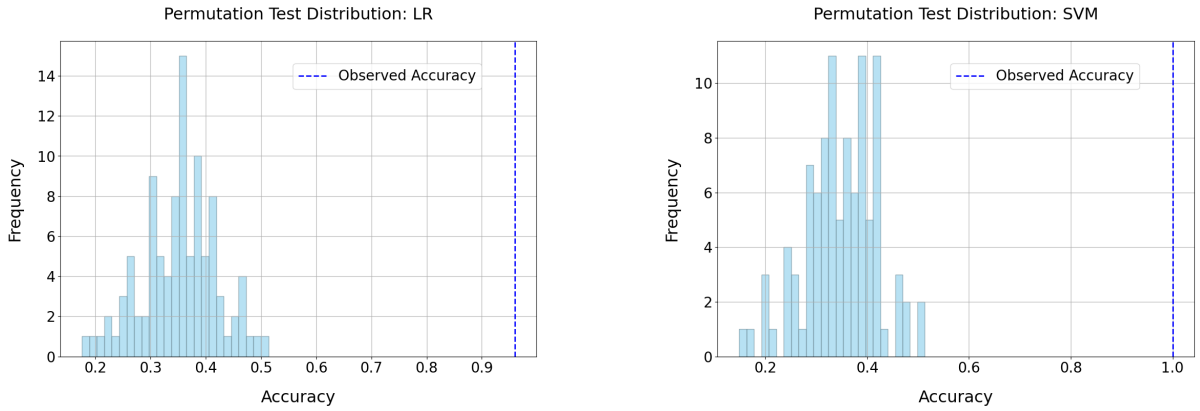
3. **Assessment of statistical significance**: The accuracy of the original model (trained on true labels) is compared to the null distribution of accuracies obtained from the permuted datasets. Statistical significance is assessed by computing a p-value, defined as the proportion of models trained on permuted labels that achieve an accuracy equal to or greater than that of the original model. A low p-value suggests that such high performance is unlikely to occur by chance, thus providing evidence against the null hypothesis. Statistical significance is evaluated with respect to a predefined threshold, typically $\alpha = 0.05$, which represents a 5% probability of observing such a result under the null hypothesis. A p-value lower than α indicates that the result is statistically significant, implying that there is sufficient evidence to reject the null hypothesis in favour of the alternative.

As previously reported, the Support Vector Machine (SVM) classifier achieved an accuracy of **100%**, while the Logistic Regression (LR) classifier achieved an accuracy of **97%** on the test set. To determine whether these performances were significantly better than what could be expected by chance, the described permutation-based statistical test was conducted.

For each classifier, a null distribution of test accuracies was constructed by repeatedly training models on versions of the training set where class labels were randomly permuted, thereby simulating the case where no true relationship exists between input features and target labels.

For the LR model, the mean accuracy under the null hypothesis was **0.34**, and the observed accuracy of **0.97** yielded a ***p*-value of 0.0099**. For the SVM model, the null distribution had a mean of **0.35**, and the perfect accuracy of **1.0** produced a ***p*-value of 0.0099**.

In both cases, the p-value is below the predefined significance threshold of $\alpha = 0.05$, leading to the rejection of the null hypothesis. This is also evident in the histograms in Figure 8.1, where the observed accuracies of both models lie far to the right of their respective null distributions, indicating significantly better performance than would be expected by random chance.



(a) LR: Accuracy vs. Null Distribution.

(b) SVM: Accuracy vs. Null Distribution.

Figure 8.1: Permutation test for classifier performance.

These results provide strong evidence that both classifiers capture generalizable patterns in the data, and that their predictive performance is not due to random chance. The rejection of the null hypothesis, supported by statistically significant p-values and large effect sizes (i.e., high observed accuracies compared to the null distributions), confirms the validity of the models' outcomes. Consequently, it can be concluded that the classifiers are not merely overfitting the training data, but rather learning meaningful relationships that generalize well to unseen data. This finding is particularly important in the context of this study, whose primary goal is to develop reliable tools for the early detection of cognitive impairment in murine models.

8.3 Limitations of the study

High-quality data are essential to correctly represent a specific phenomenon and to ensure the development of reliable and generalizable predictive models. This concept encompasses several dimensions, including accuracy, reliability, consistency, and completeness, all of which directly impact the effectiveness of machine learning models. In the context of biomedical research, where subtle variations in the data can reflect meaningful biological differences, data quality becomes even more critical.

In this study, extensive data cleaning was carried out supported by continuous interactions with domain experts, aiming to resolve errors, remove inconsistencies, and ensure reliability of the dataset. This process revealed several issues, including data entry errors, duplicates, and internal inconsistencies, which led to the exclusion of a substantial number of samples. Therefore, the final dataset comprises a relatively small number of valid samples, which limits its representativeness of the real-world environment and affects the learning capacity of machine learning models.

Furthermore, despite the data cleaning process, some limitations related to the intrinsic nature of the data may still persist, potentially compromising the results.

As discussed during the exploratory data analysis (Section 3.2.3), a particularly critical issue is the under-representation of older subjects, especially those belonging to the Tg2576 genotype, which typically exhibits clear symptoms of cognitive impairment around nine months of age.

This data gap may partially result from a higher incidence of noise and inconsistencies in records related to older animals, which may have led to their exclusion during the data cleaning phase. However, it is also likely influenced by factors inherent to the experimental context. For instance, older mice may be less frequently available for testing due to natural age-related mortality, or they may have undergone the Y-Maze test but produced performance results considered invalid for the analysis, resulting in further data loss.

The resulting lack of representation in this critical subgroup—likely to exhibit impaired cognitive behaviour—compromises the model's capacity to learn discriminative

patterns associated with cognitive decline and highlights the importance of collecting more diverse and representative data in future studies.

Despite these limitations, the models developed in this study achieved very high performance and demonstrated their validity as predictive tools within the specific experimental context considered. These results support the feasibility of using behavioural test data for the early detection of cognitive impairment and constitute the first step for further refinement and validation.

8.4 Future research directions

The cognitive impairment classifiers developed in this study represent the first outcome of the AI4ChemoBrain project, based on the analysis of the historical dataset collected over the last 15 years from preclinical models of cognitive decline. These preliminary models serve as a foundational step toward the broader objective of developing reliable predictive tools for chemobrain detection.

Future research will focus on testing the models using a new dataset specifically generated within the course of the project, derived from an experimental preclinical chemobrain model. This step is crucial to evaluate the models' ability to generalize to data that more directly reflect the targeted pathology.

The project will then validate the predictive models to ensure their robustness, effectiveness, and translatability in the clinical environment. This phase will use a dataset derived from a second preclinical model of chemobrain in subjects with cognitive decline (internal cohort) and a dataset derived from an external cohort.

The combination of the datasets will enable to study the contribution of each descriptor with respect to the models' predictive ability. Data augmentation, transfer learning, and fine-tuning techniques may be used to overcome any problems related to the availability of large datasets in the context of the chemobrain in order to improve the predictive capabilities of the ML/AI models.

The ultimate goal is the integration of these predictive systems into the clinical

setting, where they could play a role in the early identification of cognitive side effects in cancer patients. Such tools could support the design of targeted adjuvant therapies, contribute to personalised treatment strategies, and ultimately empower patients by enhancing their quality of life. Additionally, by enabling earlier interventions, these models may help reduce the long-term cognitive burden and mitigate the economic and social costs associated with chemotherapy-induced cognitive impairment.

Appendix A

Dataset Features

	Name	Type
Identifier	Animal ID	int
	Study	String
Subject-rel data	Strain	String
	Age	int
	Gender	String
YM general data	N° Entries Tot	int
	N° of Alternations	int
	% of Correct Alternations	int
	Visited Arms	String
	N° Entries A	int
	N° Entries B	int
	N° Entries C	int
	First Zone Entered	String
	Duration (s)	int
	Distance (m)	float
	Mean Speed (m/s)	float
	Max Speed (m/s)	float
	Rotations	int
	Clockwise Rotations	int
	Anti-clockwise Rotations	int
	Absolute Turn Angle (°)	int
	Path Efficiency	float
	Line Crossings	int
	Num Centre Positions	int
YM arm-rel data: A, B, C	N° Entries	int
	N° Exits	int
	Was 1st Zone	Bool
	Time (s)	float

	Name	Type
YM arm-rel data: A, B, C	Distance (m)	float
	Distance to First Entry (m)	float
	Latency to First Entry (s)	float
	Latency to First Exit (s)	float
	Latency to Last Entry (s)	float
	Average Speed (m/s)	float
	Max Speed (m/s)	float
	Mean Visit (s)	float
	Max Visit (s)	float
	Min Visit (s)	float
	Visit Duration List	String
	Initial Distance (m)	float
	Mean Distance from (m)	float
	Max Distance from (m)	float
	Min Distance from (m)	float
	Cumulative Distance (m*s)	float
	Mean Distance to Border (m)	float
	Max Distance to Border (m)	float
	Min Distance to Border (m)	float
	Time Getting Closer to Zone (s)	float
	Time Getting Further from Zone (s)	float
	Initial Heading Error (°)	int
	Signed Initial Heading Error (°)	int
	Average Absolute Heading Error (°)	int
	Time Moving Towards (s)	float
	Time Moving Away from (s)	float
	In Zone Oriented Towards Centre (s)	float
	Absolute Turn Angle (°)	int
	Path Efficiency to Entry	float
	CIPL (m*s)	float
	Line Crossings	int
YM sequence-rel data: Abc, Bca, Cab, Acb, Bac, Cba	Number	int
	Total Time (s)	float
	Latency to 1st Start (s)	float
	Latency to 1st End (s)	float
	Mean Duration (s)	float
	Max Duration (s)	float
	Min Duration (s)	float
	Total Distance (m)	float
	Mean Distance (m)	float
	Max Distance (m)	float
	Min Distance (m)	float
	Mean Speed (m/s)	float
Label	Label_80	String
	Label_90	String
	Label_100	String

Appendix B

Code Implementations

In this section, selected code snippets relevant to the implementation of the classification models are presented. The complete code developed for this project is available upon request.

The code had been implemented in **Python**, using libraries for data analysis and machine learning. In particular, the following packages had been used: **pandas**, **numpy**, **matplotlib**, **seaborn** and **scikit-learn**.

Listing B.1: Pipeline for the classification task.

```
def preprocessing_pipeline(categorical_features, numerical_features, best_normalization,
     best_dim_red, apply_dim_red, feature_selection_method, min_features_to_select,
    model, seed=42):

    # Preprocessing Step: Feature Encoding and Normalization
    preprocessor = ColumnTransformer(
        transformers=[
            # Encoding for Categorical Features
            ("cat", OneHotEncoder(), categorical_features),
            # Normalization for Numerical Features
            ("num", best_normalization, numerical_features)
        ],
    )
    # Pipeline Definition: base
    pipeline = Pipeline(steps=[
        ("preprocessor", preprocessor), # encoding and normalization
        ("model", model) # modeling
    ])
```

```
    # Preprocessing Step: Dimensionality Reduction
    if apply_dim_red:
        pipeline = Pipeline(steps=[
            ("preprocessor", preprocessor),
            ("reduction", best_dim_red), # dimensionality reduction
            ("model", model)
        ])
        # Return the pipeline
        return pipeline

    # Preprocessing Step: Feature Selection
    selector = ""
    match feature_selection_method:
        case "RFE":
            estimator = LogisticRegression(random_state=seed, max_iter=1000,
                class_weight="balanced")
            selector = RFE(estimator)
        case "RFE-CV":
            estimator = LogisticRegression(random_state=seed, max_iter=1000,
                class_weight="balanced")
            selector = RFECV(estimator, step=1, cv=StratifiedKFold(3, random_state=seed,
                 shuffle=True), scoring="accuracy", min_features_to_select=
                min_features_to_select, n_jobs=2)
        case "SelectKBest":
            selector = SelectKBest(score_func=f_classif, k=5)
        case "SelectFromModel":
            estimator = LogisticRegression(random_state=seed, max_iter=1000,
                class_weight="balanced")
            selector = SelectFromModel(estimator, threshold="mean")
        case _:
            # Return the pipeline
            return pipeline

    pipeline = Pipeline(steps=[
        ("preprocessor", preprocessor),
        ("selector", selector), # feature selection
        ("model", model)
    ])
    # Return the pipeline
    return pipeline
```

Listing B.2: Hyperparameter tuning for supervised models.

```
def hyperparameter_tuning(X_train, y_train, pipeline, param_grid, search_type="grid",
    n_iter=50, scoring="f1_weighted", seed=42):

    # Stratified K Fold cross-validation
    skf = StratifiedKFold(n_splits=3, shuffle=True, random_state=seed)
```

```
    # Define the search type
    if search_type == "grid":
        search = GridSearchCV(pipeline, param_grid=param_grid, cv=skf, n_jobs=-1,
            verbose=2, scoring=scoring)
    elif search_type == "random":
        search = RandomizedSearchCV(pipeline, param_distributions=param_grid, n_iter=
            n_iter, cv=skf, random_state=seed, n_jobs=-1, verbose=2, scoring=scoring)
    else:
        raise ValueError("search_type must be 'grid' or 'random'")

    # Fit on train set
    search.fit(X_train, y_train)

    best_model = search.best_estimator_
    print("Best hyperparameters:", search.best_params_)
    print("Best cross-validated accuracy:", search.best_score_)

    # Return the model
    return best_model
```

Listing B.3: Application of the preprocessing and tuning functions to the SVM model.

```
# hyperparameters
param_grid_svm = [
    {"model__kernel": ["linear"], "model__C": [0.1, 1, 10]},
    {"model__kernel": ["rbf"], "model__C": [0.1, 1, 10], "model__gamma": [0.1, 1]},
    {"model__kernel": ["poly"], "model__C": [0.1, 1, 10], "model__gamma": [0.1, 1],
                                "model__degree": [2, 3, 4]}
]

# Define the Model
model = SVC(random_state=seed, class_weight="balanced")

# Preprocessing Pipeline
pipeline = preprocessing_pipeline(categorical_features, numerical_features,
    best_normalization, best_dim_red_svm, apply_dim_red, feature_selection_method,
    min_features_to_select, model)

# Hyperparameter Tuning
svm = hyperparameter_tuning(X_train, y_train, pipeline, param_grid_svm)

# the model automatically applies the pipeline defined in the tuning
print("Performance on Train Set:", svm.score(X_train, y_train))
print("Performance on Test Set:", svm.score(X_test, y_test))
```

References

[1] B. Fleming, P. Edison, and L. Kenny, "Cognitive impairment after cancer treatment: mechanisms, clinical characterization, and management," *BMJ*, vol. 380, 2023. [Online]. Available: https://doi.org/10.1136/bmj-2022-071726

[2] IRET Foundation, "AI4ChemoBrain." [Online]. Available: https://ai4chemobrain.it/en/home/

[3] T. Bayne, D. Brainard, R. W. Byrne, L. Chittka, N. Clayton, C. Heyes, J. Mather, B. Ölveczky, M. Shadlen, T. Suddendorf, and B. Webb, "What is cognition?" *Current Biology*, vol. 29, no. 13, 2019. [Online]. Available: https://doi.org/10.1016/j.cub.2019.05.044

[4] M. Ghafarimoghadam, R. Mashayekh, M. Gholami, P. Fereydani, J. Shelley-Tremblay, N. Kandezi, E. Sabouri, and M. Motaghinejad, "A review of behavioral methods for the evaluation of cognitive performance in animal models: Current techniques and links to human cognition," *Physiology & Behavior*, vol. 244, 2022. [Online]. Available: https://doi.org/10.1016/j.physbeh.2021.113652

[5] H. Eichenbaum, "Hippocampus: cognitive processes and neural representations that underlie declarative memory," *Neuron*, vol. 44, no. 1, 2004. [Online]. Available: https://doi.org/10.1016/j.neuron.2004.08.028

[6] D. Puzzo, W. Gulisano, A. Palmeri, and O. Arancio, "Rodent models for Alzheimer's disease drug discovery," *Expert opinion on drug discovery*, vol. 10, no. 7, 2015. [Online]. Available: https://doi.org/10.1517/17460441.2015.1041913

[7] R. M. Rodriguiz and W. C. Wetsel, *Assessments of Cognitive Deficits in Mutant Mice.* Levin ED, Buccafusco JJ, editors. Animal Models of Cognitive Impairment. Boca Raton (FL): CRC Press/Taylor & Francis, 2006. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK2527/

[8] C. V. Vorhees and M. T. Williams, "Morris water maze: procedures for assessing spatial and related forms of learning and memory," *Nature protocols*, vol. 1, no. 2, 2006. [Online]. Available: https://doi.org/10.1038/nprot.2006.116

[9] A. V. J. Terry, *Spatial Navigation (Water Maze) Tasks.* Buccafusco JJ, editor. Methods of Behavior Analysis in Neuroscience. 2nd edition. Boca Raton (FL): CRC Press/Taylor & Francis, 2009. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK5217/

[10] M. Z. Othman, Z. Hassan, and A. T. C. Has, "Morris water maze: a versatile and pertinent tool for assessing spatial learning and memory," *Experimental animals*, vol. 71, no. 3, 2022. [Online]. Available: https://doi.org/10.1538/expanim.21-0120

[11] A.-K. Kraeuter, P. C. Guest, and Z. Sarnyai, "The Y-Maze for Assessment of Spatial Working and Reference Memory in Mice," *Methods in molecular biology (Clifton, N.J.)*, vol. 1916, 2019. [Online]. Available: https://doi.org/10.1007/978-1-4939-8994-2_10

[12] J. Bak, H.-I. Pyeon, J.-I. Seok, and Y.-S. Choi, "Effect of rotation preference on spontaneous alternation behavior on Y maze and introduction of a new analytical method, entropy of spontaneous alternation," *Behavioural Brain Research*, vol. 320, 2017. [Online]. Available: https://doi.org/10.1016/j.bbr.2016.12.011

[13] M. Cleal, B. D. Fontana, D. C. Ranson, S. D. McBride, J. D. Swinny, E. S. Redhead, and M. O. Parker, "The Free-movement pattern Y-maze: A cross-species measure of working memory and executive function," *Behavior research methods*, vol. 53, no. 2, 2021. [Online]. Available: https://doi.org/10.3758/s13428-020-01452-x

[14] J. Kim, H. Kang, Y.-B. Lee, B. Lee, and D. Lee, "A quantitative analysis of spontaneous alternation behaviors on a Y-maze reveals adverse effects of acute

social isolation on spatial working memory," *Scientific Reports*, vol. 13, no. 14722, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-41996-4

[15] J. Han, M. Kamber, and J. Pei, "2 - Getting to Know Your Data," in *Data Mining (Third Edition)*. Morgan Kaufmann, 2012. [Online]. Available: https://doi.org/10.1016/B978-0-12-381479-1.00002-2

[16] M. L. Waskom, "seaborn.boxplot." [Online]. Available: https://seaborn.pydata.org/generated/seaborn.boxplot.html

[17] M. L. Waskom, "seaborn.scatterplot." [Online]. Available: https://seaborn.pydata.org/generated/seaborn.scatterplot.html

[18] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction," *ArXiv e-prints 1802.03426*, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1802.03426

[19] M. L. Waskom, "seaborn.swarmplot." [Online]. Available: https://seaborn.pydata.org/generated/seaborn.swarmplot.html

[20] J. Han, M. Kamber, and J. Pei, "3 - Data Preprocessing," in *Data Mining (Third Edition)*. Morgan Kaufmann, 2012. [Online]. Available: https://doi.org/10.1016/B978-0-12-381479-1.00003-4

[21] M. L. Waskom, "seaborn.heatmap." [Online]. Available: https://seaborn.pydata.org/generated/seaborn.heatmap.html

[22] scikit-learn developers, "OneHotEncoder - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

[23] scikit-learn developers, "LabelEncoder - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[24] IBM, "What is principal component analysis (PCA)?" [Online]. Available: https://www.ibm.com/think/topics/principal-component-analysis

[25] IBM, "What is linear discriminant analysis (LDA)?" [Online]. Available: https://www.ibm.com/think/topics/linear-discriminant-analysis

[26] scikit-learn developers, "RFE - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[27] scikit-learn developers, "RFECV - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html

[28] scikit-learn developers, "SelectKBest - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

[29] scikit-learn developers, "SelectFromModel - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html

[30] scikit-learn developers, "train_test_split - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[31] IBM, "What is upsampling?" [Online]. Available: https://www.ibm.com/think/topics/upsampling

[32] IBM, "What is downsampling?" [Online]. Available: https://www.ibm.com/think/topics/downsampling

[33] IBM, "What is upsampling? - Upsampling techniques." [Online]. Available: https://www.ibm.com/think/topics/upsampling#Upsampling+techniques

[34] Towards Data Science, "Why Weight? The Importance of Training on Balanced Datasets." [Online]. Available: https://towardsdatascience.com/why-weight-the-importance-of-training-on-balanced-datasets-f1e54688e7df/

[35] scikit-learn developers, "cross_validation - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[36] scikit-learn developers, "Pipeline - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline

[37] scikit-learn developers, "ColumnTransformer - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.compose.ColumnTransformer.html#sklearn.compose.ColumnTransformer

[38] scikit-learn developers, "Linear Models, Logistic Regression - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression

[39] scikit-learn developers, "Support Vector Machines - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/svm.html

[40] scikit-learn developers, "Random Forests - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/ensemble.html#forest

[41] scikit-learn developers, "Nearest Neighbors - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/neighbors.html

[42] scikit-learn developers, "Clustering, K-Means - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#k-means

[43] scikit-learn developers, "Clustering, HDBSCAN - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#hdbscan

[44] scikit-learn developers, "Gaussian mixture models - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html#mixture

[45] scikit-learn developers, "StandardScaler - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[46] J. Han, M. Kamber, and J. Pei, "8 - Classification: Basic Concepts," in *Data Mining (Third Edition)*. Morgan Kaufmann, 2012. [Online]. Available: https://doi.org/10.1016/B978-0-12-381479-1.00008-3

[47] scikit-learn developers, "Precision-Recall - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

[48] scikit-learn developers, "Permutation feature importance - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/modules/permutation_importance.html

[49] J. Han, M. Kamber, and J. Pei, "10 - Cluster Analysis: Basic Concepts and Methods," in *Data Mining (Third Edition)*. Morgan Kaufmann, 2012. [Online]. Available: https://doi.org/10.1016/B978-0-12-381479-1.00010-1

[50] scikit-learn developers, "Selecting the number of clusters with silhouette analysis on KMeans clustering - scikit-learn." [Online]. Available: https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

[51] P.-N. Tan, M. Steinbach, A. Karpatne, and V. Kumar, "10 - Avoiding False Discoveries," in *Introduction to Data Mining (Second Edition)*. Pearson, 2018.