Exploiting machine learning techniques to predict Alzheimer's Disease Progression

João Paulo Goulart Pedrosa

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Pedro Filipe Zeferino Tomás
Prof. Helena Isabel Aidos Lopes Tomás

Examination Committee

Chairperson: Prof. Francisco António Chaves Saraiva de Melo
Supervisor: Prof. Pedro Filipe Zeferino Tomás
Member of the Committee: Prof. Sara Alexandra Cordeiro Madeira

January 2021

Abstract

Affecting 30% of the population above 85 years of age, Alzheimer's Disease (AD) is a chronic neurodegenerative disease responsible for about two-thirds of dementia cases worldwide. With the ageing in the global population, the number of AD patients is expected to rise significantly in the coming years. As most of the tests that detect AD are either too expensive or invasive, we turn our heads to neuropsychological tests and machine learning to help solve this issue. These tests assess the cognitive abilities of patients and can be taken in less than one hour with little expense. In search for the optimal solution, we look into state-of-the-art technologies for classification, missing value imputation (MVI) and other steps in the data mining process. From this, we manage to build a working classification pipeline capable of analyzing the test data and predicting a patient's future conversion to AD as well as its time frame. This prediction is performed for a certain time window, and with a certain degree of confidence. Our solution to improve upon this work is to implement state-of-the-art algorithms and test different configurations until an ideal setup is determined.

Keywords

Alzheimer's Disease; Mild Cognitive Impairment; Machine Learning; Deep Learning; Classification; Missing Value Imputation; Class Imbalance; Feature Selection.

Resumo

Afetando 30% da população acima de 85 anos de idade, a Doença de Alzheimer (AD) é uma doença crónica neurodegenerativa responsável por cerca de dois terços dos casos mundiais de demência. Com o envelhecimento da população global é esperado que o número de pacientes com AD suba significativamente nos próximos anos. Visto que a maioria dos testes que detetam AD são demasiado caros ou invasivos, viramo-nos para os testes neuropsicológicos e machine learning para ajudar a solucionar o problema. Estes testes avaliam as habilidades cognitivas dos pacientes e podem ser realizados em menos de uma hora com pouca despesa. Em busca da solução ideal, olhamos para tecnologias do estado da arte nas áreas de classificação, imputação de valores em falta (MVI) e outros passos no processo de data mining. A partir disso, construímos uma pipeline de classificação capaz de analisar dados de teste e prever a futura conversão de um paciente para AD, bem como o horizonte temporal desta conversão. Esta previsão é efetuada para uma certa janela temporal e com um certo grau de confiança. A nossa solução para reforçar este trabalho passa por implementar algoritmos do estado da arte e testar diferentes configurações até que uma configuração ideal seja determinada.

Palavras Chave

Doença de Alzheimer; Défice Cognitivo Ligeiro; Aprendizagem de Máquina; Aprendizagem Profunda; Classificação; Imputação de valores; Balanceamento de Classes; Seleção de Features.

Contents

List of Figures

List of Tables

List of Equations

Acronyms

AD Alzheimer’s Disease

ADAS-Cog Alzheimer’s Disease Assessment Scale - Cognitive Subscale

ADASYN Adaptive Synthetic

AE Autoencoder

AUC Area Under the Curve

BC Breast Cancer Wisconsin Dataset

BCE Binary Crossentropy

BLAD Lisbon Battery for Dementia Evaluation

CBIS Cluster-Based Instance Selection

CCC Cognitive Complaints Cohort

cMCI Converter MCI

CB Class Balancing

CS Chi-Squared

CT Computed Tomography

CV Cross Validation

CSF Cerebrospinal Fluid

DAE Denoising Autoencoder

DNA Deoxyribonucleic Acid

DT Decision Tree

EM Expectation-Maximization

FMUL Faculty of Medicine of the University of Lisbon

FS Feature Selection

GAN Generative Adversarial Network

JMI	Joint Mutual Information
KC	House Sales in King County Dataset
KNN	K-Nearest-Neighbour
KNNI	K-Nearest Neighbours Impute
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAR	Missing At Random
MCAR	Missing Completely At Random
MCI	Mild Cognitive Impairment
MF	MissForest
MI	Mutual Information
MICE	Multiple Imputation by Chained Equations
MIM	Mutual Information Maximization
MLE	Maximum Likelihood Estimation
MMSE	Mini Mental State Exam
MNAR	Missing Not At Random
MRI	Magnetic Resonance Imaging
MRMR	Maximum Relevance Minimum Redundancy
MSE	Mean Squared Error
MVI	Missing Value Imputation
NB	Naive Bayes
NN	Neural Network
NPT	Neuropsychological Test
OSM	Overall Sample Mean
PET	Positron Emission Tomography
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
RFE	Recursive Feature Elimination

RMSE	Root Mean Square Error
RNA	Ribonucleic Acid
RNN	Recurrent Neural Network
RO	Random Oversampling
ROC	Receiver Operating Characteristic
RU	Random Undersampling
RW	Red Wine Quality Dataset
sMCI	Stable MCI
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support-Vector Machine
TOMEK	Tomek Links
UF	UFC-Fight Historical Data Dataset
VAE	Variational Autoencoder

1 Introduction

Contents

1.1 Motivation and Problem Formulation

Alzheimer's Disease (AD) is a complex chronic neurodegenerative disease responsible for about two thirds of dementia cases worldwide. Its causes are poorly understood and it remains untreatable. As of 2019 it is estimated that AD affects 3% of people aged 65 or younger, although the number rises to above 30% for people aged 85 and older [2].

At the earliest stages of the disease a full manifestation of the symptoms does not occur. Usually, a patient is first diagnosed with Mild Cognitive Impairment (MCI) when some cognitive declines begin to appear (such as memory lapses and motor difficulties). However, diagnosing early stage AD is not a straightforward process, as it is necessary to understand the differences between MCI caused declines and declines generally caused by ageing. Furthermore, neurodegenerative diseases can take years to manifest, all the while draining at MCI patients' cognitive abilities. By the time a patient is diagnosed with dementia, their brain would have already suffered irreparable damage, severely impacting autonomy and cognition. The World Alzheimer Report 2015 [3] estimated that, in 2015, 818 billion dollars were spent worldwide in the care of AD patients, a figure which would amount to the world's 18th largest economy. And with the ever ageing of the global population, it is expected that the number of people living with dementia will nearly triple to 131 million by 2050.

Being able to predict the progress from patients with MCI to AD is of great importance. It can help with the appropriate selection of therapeutic interventions for each unique patient, as well as improving their quality of life for the following years by slowing the decline in cognitive skills. It can also have a great social-economic impact, as it would reduce unnecessary tests and procedures on millions of patients worldwide. Lastly it could have a large impact on the families and patients themselves, by providing some predictability to a disease ridden with uncertainty.

This work addresses the problem of predicting the evolution from MCI to AD in any given patient, as well as predicting the time of such evolution. To achieve this, several datasets with neuropsychological data will be used. The datasets contain the scores of multiple neuropsychological tests performed by each patient to evaluate their present cognitive capabilities. The different sets are organized into yearly intervals, so as to help determine how long the disease takes to evolve in each patient.

We will use state of the art machine learning techniques to help determine the time until conversion to AD, with a particular focus on the field of missing value imputation. Our aim is to innovate and explore new and unique approaches to solve the problems at hand.

1.2 Objectives

In this work we will handle a constantly updating dataset, upon which several pieces of research have previously been performed [4] [5]. Our general goal is to build upon previous work, applying state-of-the-art deep learning and data processing techniques in order to obtain prediction models with increased accuracy.

By analyzing the work previously performed with the considered dataset, we identified missing value imputation as a specific area where we could improve and innovate in order to reach our goals. We will carefully study recent deep learning techniques to address this problem, and as such, a significant part of this work will be devoted to this topic.

Nonetheless, we decided to further research and experiment with other steps of the data mining process, to ensure that state-of-the-art approaches are considered for each step. As such, we aim to explore the best methods available for each step along the data mining pipeline, and if possible develop our own methods tailored to our specific problem.

By the end, we should be able to improve upon previous works, achieving more accurate results.

1.3 Contributions

This work tackles the problem of predicting the conversion from MCI to AD, recurring to state-of-the-art machine learning techniques. Regarding these fields, our contributions are:

1. Direct comparison between different models utilized for missing value imputation. We studied the performance results of these models utilizing diverse sets of real-life data.

2. Development of a modular data mining pipeline, capable of swapping between different methods for each stage of the process: (i) Missing Value Imputation; (ii) Feature Selection; (iii) Class Balancing; (iv) Classification; (v) Result evaluation.

3. Application of the developed pipeline to the problem of predicting the conversion from MCI to AD.

1.4 Outline

The remaining of this document is organized as follows:

- **Chapter 2** initially describes the main processing pipeline for the classification process, from receiving the data to returning a prediction result. It then overviews the state-of-the-art methods that constitute the aforementioned pipeline, as well as other aspects relevant to machine learning. The chapter finishes with an overview of previous work developed in the areas of machine learning and AD.

- **Chapter 3** further explores the topic of Missing Value Imputation. In this chapter several state-of-the-art methods will be implemented, tested and compared.

- **Chapter 4** explores the topic of Alzheimer’s Disease, including our approach to address the problem. It concludes with our proposed final setup for the data mining pipeline.
- **Chapter 5** contains the bulk of the experimental work performed. Here the proposed data mining pipeline will be tested. Furthermore we will present and discuss all the experiments performed in order to reach the pipeline setup capable of returning the most accurate classification results.
- **Chapter 6** ends this paper with a brief discussion of the results obtained as well as describing steps that can be taken in future work.

2 Background on Machine Learning

Contents

The dataset being used in this work contains a heterogeneous set of data (more on this in section 4.2). This includes both categorical and numerical features, as well as features with different proportions of missing data. Furthermore, each feature has a different or unknown correlation to MCI and AD. In order to solve problems like these with such datasets there is the need to set up a classification pipeline. Ideally, any step in this pipeline should be adjustable without requiring any changes to the remaining steps. This structure allows us to experiment with different methods and different configurations in order to achieve the best possible results. In problems like these, the pipeline is generally composed of the following steps:

1. **Data Cleaning** - This step consists in scanning the database to eliminate any errors or inconsistent data. Some examples can include the presence of decimal values in categorical features, or values that have no explanation other than human error. Features with an extremely high amount of missing data should also be eliminated. As for the datasets being used, this step has already been performed by reporting all errors to the doctors responsible for its maintenance.

2. **Feature Selection** - In the Feature Selection step, a subset of features are selected from the original dataset, based on how relevant or unique they are. Several different methods can be used for this process, with each of them measuring and comparing different characteristics of the features. Some state of the art methods will be explored to address this problem.

3. **Missing Value Imputation** - In this step, algorithms are applied to the dataset in order to generate values to replace any missing data. The existence of missing data within a dataset can heavily affect the results of some classifiers, making this an important step in the classification process. This step is of significant importance in the work performed on this thesis.

4. **Handling Class Imbalance** - A class imbalance exists when the final classifications for all samples in the dataset are not proportionally distributed. Most classifiers excel with a balanced dataset. Due to this we will explore several approaches to balance the distribution of points in each class.

5. **Classification** - This is the process that takes our modified datasets and attempts to extract patterns from them. The recognition of patterns will then allow the classifier to predict the classes for new data samples. The best state of the art classifiers will be explored and used to achieve accurate results.

6. **Evaluation** - In order to determine the best technologies to use and the overall best configuration, all results need to be compared and evaluated. To do this we will determine several evaluation metrics and processes to compare the results achieved.

The following subsections briefly describe the state-of-the-art approaches for each of the steps in the classification pipeline.

2.1 Feature Selection

Feature Selection is the process of restricting the number of features to a subset of the original dataset. This process can be of great importance to the outcome of the classification for several reasons:

- Reducing overfitting - The presence of redundant features within a dataset can hinder the classification process, as it can deceive the classifiers into focusing on coincidences within the dataset. Eliminating such features can in turn reduce the chance of the classifier overfitting on the training data.
- Decreasing training time - Training a classifier on a dataset of smaller dimensionality can lead to a decrease in execution time, which can be of great value depending on the context.
- Ease of understanding - Reducing the entire dataset to its most important features can lead to a better understanding of the dataset, and the problem being addressed. It can allow the user to understand why the selected features are deemed as more important and positively influence future collection of data.

Feature selection algorithms can be divided into supervised and unsupervised [6]. This distinction is based on whether the algorithms consider the target class for their selection. Unsupervised algorithms typically focus on measuring correlation between features and then eliminating redundant ones. Supervised algorithms, on the other hand, take the target value into consideration and use it to determine and eliminate irrelevant features.

Feature Selection algorithms can also be divided into Wrapper, Filter and Embedded [7]. Filter methods, which are typically supervised, utilize statistical models to evaluate the relationships between input vectors and target classes. Wrapper methods create models that are trained on different subsets (containing different sets of features) of the original dataset. After training, an internal metric is used to assert which of the subsets results in the best performance, and the selected subset is returned. Embedded methods are an umbrella term for techniques that perform feature selection as part as the model construction process.

There are several methods for performing Feature Selection, some of them are Chi-Squared, Recursive Feature Elimination and several Mutual Information methods, as described in the next subsections.

2.1.1 Mutual Information methods

Mutual Information (MI), also known as Information Gain, is a measure that can be calculated between a pair of vectors X and Y, that quantifies the amount of information obtained about one variable through the other. Simply put, it represents the amount of information that is gained after a particular transformation in the dataset. Mutual Information is given by

$$I(X,Y) = \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \tag{2.1}$$

where $p(x)$ and $p(y)$ are the marginal density functions of vectors X and Y, and $p(x,y)$ is the joint probability density function. This function determines the similarity between the joint distribution and the product of the marginal distributions. In a scenario where X and Y are completely independent and unrelated, $p(x,y)$ would be equal to $p(x)p(y)$, and the integral would be zero.

Knowing this allows for the MI measure to be utilized to compare features within datasets and select the best set. A general function derived to obtain a subset from an original dataset can be given by

$$f_t = \arg \max_{i \notin S^{t-1}} I(x_i, y) - \left[\alpha \sum_{k=1}^{t-1} I(x_{f_k}, x_i) - \beta \sum_{k=1}^{t-1} I(x_{f_k}, x_i|y) \right] \tag{2.2}$$

where S^{t-1} represents a set of features selected at threshold $t-1$ [8]. Within the function, $I(x_i, y)$ gives us the relevance of the selected feature, while the block in square brackets gives us the redundancy of that same feature. There exist several feature selection methods that utilize this formula, with each one selecting their specific values for α and β:

- Mutual Information Maximization (MIM) [9]: $\alpha = 0$ and $\beta = 0$.
- Maximum Relevance Minimum Redundancy (MRMR) [10]: $\alpha = \frac{1}{t-1}$ and $\beta = 0$.
- Joint Mutual Information (JMI) [11]: $\alpha = \frac{1}{t-1}$ and $\beta = \frac{1}{t-1}$.

All these are Feature Selection methods that can be categorized as both filter and supervised. Each of them requires for a threshold t to be given, as well as a set of data X and the target values Y. In return, the methods output a subset of X, according to equation 2.2.

2.1.2 Chi-Squared

The Chi-Squared (X^2 or CS) is a statistical test used to determine whether a statistically significant difference exists between the observed vector O and the expected vector E. Chi-Squared is given by

$$X_c^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{2.3}$$

where c represents the degree of freedom [12].

When applying this test to a particular feature vector X and the target vector Y, the test allows to determine whether a dependence exists between the two. In a scenario where X and Y are completely independent and unrelated, the observed values would be closer to the expected values, resulting in a smaller Chi-Squared value.

In feature selection, features with high dependence on the target are ideal. The Chi-Squared algorithm for feature selection runs the X^2 test on every feature/target pair, obtaining the score for each. The algorithm is given a threshold t, as well as a set of data X and the target values Y, and returns a subset of X with all feature which have a X^2 score greater than t. Similarly to the MI based methods, the CS method can be categorized as both filter and supervised.

2.1.3 Recursive Feature Elimination

Recursive Feature Elimination (RFE) is a supervised wrapper method that relies on a classifier to determine the most important features in a dataset [13]. This technique begins by fitting the given classifier to the entire dataset. The model will then have computed different coefficients for each feature in the original dataset. These coefficients reflect the importance of each feature as determined by the classifier. The features deemed least important are removed from the dataset and the process is iteratively repeated with the newly obtained dataset. The cycle is repeated until a stopping criterion is met.

As was mentioned before, this technique relies on the feature coefficients obtained by the classifier being utilized. However, not all classifiers are able to determine such values, which limits the pool of classifiers that can be used alongside this method. Some of the classifiers that can be used with RFE include Multiple Linear Regression, Logistic Regression, Linear Discriminant Analysis and Support Vector Machine (more about these classifiers on Section 2.4). In the place of a classifier, a statistical model (such as Mutual Information or Chi-Squared test) can be utilized instead, resulting in a combination of wrapper and filter methods.

Some classifiers have shown to benefit particularly from the use of RFE, such as the Random Forest (RF) classifier [14]. As it is explained in Section 2.4.3, this classifier creates its own subsets of data, with each containing a random selection of features. By utilizing the RFE method, the chances of the RF classifier creating subsets with irrelevant features is greatly decreased.

2.2 Missing Value Imputation

The presence of missing data within a dataset is a common occurrence in the field of data science. Missing data can be present in datasets for a variety of reasons, foreseeable or not, making it at times impossible to avoid. The absence of data can pay an important role in the classification process, as the missing values could have been directly related to the sample's class. To overcome this problem a process called Missing Value Imputation (MVI) is used. MVI consists of generating passable values to replace instances of missing data within a dataset, so that all classifiers can utilize the imputed data and some can manage to improve their predictions.

There are several models to address MVI, which can be divided into two categories: Discriminative and Generative [15]. Discriminative Models merely model the decision boundaries between the different classes while Generative Models focus on learning the probability distribution of the individual classes. Considering inputs x and class y, Discriminative models learn the conditional probability distribution ($p(y|x)$), while Generative models learn the joint probability ($p(x, y)$). This property also allows Generative models to calculate the posterior ($p(y|x)$), by applying the Bayes rules to the learned probability distributions [16].

Considering the problem of mapping input x to a given class, Discriminative models are generally more efficient, as they tend to require less computational power. However, Generative models acquire deeper understanding of the data distribution, which in turn can be an advantage for problems such as the one of generating new data samples.

Discriminative models include approaches based on: Mean/median/mode, Regression, Stochastic Regression, K-Nearest Neighbours, Random Forests and Last Observation Carried Forward, among others.

Generative models include approaches based on: Expectation Maximization, Denoising Autoencoders, Variational Autoencoders, Generative Adversarial Networks and Recurrent Neural Networks, among others.

Some of these models will be explored in further detail in Section 3.1.

2.3 Class Balance

Class imbalance exists when the different classes in a given dataset are not evenly distributed among all the data instances. In a binary dataset, such as the datasets handled in this work, class imbalance simply means that significantly more than 50% of the entries belong to one of the classes.

The majority of classifiers require a balanced dataset to perform accurate predictions. It has been shown that when these classifiers are built and trained with an imbalanced dataset they become more likely to develop a bias that favours the most common class. Performing classification with training and testing datasets where the classes are similarly unbalanced might result in an acceptable accuracy, but the true positive and true negative rates would present seriously disparate values (more on this in section 2.5). This is of special importance when dealing with medical data.

To mitigate this problem, the best and most common approaches rely on re-sampling techniques. This process can be divided into two categories: undersampling and oversampling.

2.3.1 Undersampling

The process of undersampling can be simply described as the removal of instances belonging to the majority class of the dataset until class balance is reached. There are several different techniques to perform this, some of them are Random Undersampling, Tomek Links and Cluster-Based Instance Selection, as described below.

2.3.1.A Random Undersampling

The simplest and most common undersampling technique. As the name suggests, this technique makes it so that the instances to be removed from the dataset are picked at random from the majority class.

Random Undersampling (RU) allows the final dataset to be a subset of the original one without modifying any of the instances. However it comes with several downsides [17]. By randomly removing data entries, there is the possibility that some of the more important or useful instances are chosen to be removed. Undersampling also results in an overall smaller dataset, which can impact the learning process, resulting in a lower classification accuracy.

2.3.1.B Tomek Links

This method utilizes a neighbour-based approach to perform undersampling [18]. It attempts to find ”Tomek links”, which are pairs of data instances of opposite classes that are each other’s nearest neighbour. Once these links are found, the element belonging to the majority class is usually eliminated, although this method can also be utilized to remove instances from the minority class.

Essentially this method allows for the clarification of the border between the majority and minority classes, by removing instances close to the border, as well as some outlier instances.

2.3.1.C Cluster-Based Instance Selection

Cluster-Based Instance Selection (CBIS) is an approach to undersampling that combines clustering analysis with instance selection techniques [19]. Clustering analysis is a type of unsupervised machine learning where data samples are grouped into clusters according to similarities in their features. Data samples within each cluster should be more similar to each other than to any other sample outside their respective cluster. Instance selection (similarly to Feature Selection 2.1) is a data preprocessing process that aims to remove unrepresentative and noisy data samples from a dataset. The principle behind this method is that not all data is equally informative to a classifier, and some samples can even hinder its process.

The CBIS method is divided into two steps: First, the chosen clustering algorithm is used on the majority class, assigning each data sample to a cluster. After the majority class has been organized into clusters, temporary new classes are assigned to the samples according to their cluster. Then, the chosen instance selection algorithm is used on each cluster, with the selected data samples being eliminated from the final dataset. The final dataset might not achieve a class balance, as the amount of removed data samples is determined solely by the instance selection algorithm.

2.3.2 Oversampling

By contrast to undersampling, oversampling consists of adding new (synthetic) data instances to the minority class. Some of the oversampling techniques are Random Oversampling [20], SMOTE [21], ADASYN [22] and MAHAKIL [23], as described below.

2.3.2.A Random Oversampling

In an analogous way to the Random Undersampling process, there exists Random Oversampling (RO) [20], where samples are picked at random from the minority class of the dataset and are duplicated until there is a class balance. Similarly to its counterpart, this method comes with some drawbacks, namely that it can significantly increase the chances of overfitting, resulting in an even worse performance for the classifier [20].

2.3.2.B SMOTE

One of the most well-known is the Synthetic Minority Oversampling Technique (SMOTE) [21]. This technique's behaviour can be described as:

1. A random instance from the minority class is chosen and its k nearest neighbours are found (by comparing their values for all features).
2. Out of the k instances, one is randomly picked. This randomly picked instance will be compared to the initial instance and a vector will be determined from the differences in values for all the features of these two entries.
3. After the vector is obtained it will be multiplied by a randomly generated number between 0 and 1
4. The new vector will then be added to the original instance in order to determine the new data entry.

Essentially SMOTE generates a new instance whose values are contained in-between two other minority instances. This approach allows it to maintain the same information in the dataset while not resulting in significant increases to the chances of overfitting the data. The SMOTE method can also be considered a mixed-method for class balance, as it can perform undersampling and oversampling simultaneously, by adding a step that randomly removes samples from the majority class to the algorithm previously described.

2.3.2.C ADASYN

The Adaptive Synthetic method [22] (ADASYN) generates new samples with a focus on increasing the representation of the hardest instances to learn. Its algorithm is quite similar to the SMOTE method's, differing mainly in the first step: With SMOTE a random minority instance is chosen and its k nearest neighbours are then determined. With ADASYN however, the k nearest neighbours for all data instances are found and a ratio r_{uv} based on the Euclidean distance between each pair (u, v) is determined.

$$r_{uv} = \frac{\frac{\Delta_{uv}}{k}}{\sum_i \frac{\Delta_i}{k}} \tag{2.4}$$

The data instances will then be chosen based on the computed ratios, prioritizing the instances with the largest ratios, $i.e.$, the instances with the furthest k nearest neighbours. After this selection, new data samples are generated in an identical manner to the SMOTE algorithm's. New data samples are generated until a class balance is reached. This approach increases the representation of more isolated data instances, theoretically making it easier for classifiers to learn these instances.

2.3.2.D MAHAKIL

This oversampling technique was developed to tackle heavily imbalanced datasets, while aiming to avoiding overlaps in data samples [23]. It is based on two principles: the chromosomal theory of inheritance, $i.e.$, the principle that children receive half of their chromosomes from each parent; and the belief that the Mahalanobis distance metric [24] can be better at addressing the problem of class balancing than more commonly used distance metrics, such as the Euclidean distance. The Mahalanobis is given by:

$$D = \sqrt{(\vec{x} - \vec{\mu})^T \Sigma^{-1} (\vec{x} - \vec{\mu})} \tag{2.5}$$

where $\vec{x}$ represents the vector of an observation, $\vec{\mu}$ represents the vector of mean values and Σ represents the covariance matrix. This metric is chosen for the MAHAKIL method due to the tendency of imbalanced datasets to have noise and duplicate values. This potential high correlation in such datasets would not be recognized by algorithms utilizing other metrics such as the Euclidean distance.

This MAHAKIL algorithm is divided into three steps:

1. The Mahalanobis distance is computed for all minority samples in the dataset. The samples are sorted in descending order of the computed distance.
2. The middle sample is found and used to divide all the samples into two bins, still sorted in de-

scending order of distance, in a way that the last element of the first bin has a larger distance value than the first element of the second bin.

3. Both bins are iterated simultaneously and the pair of samples selected from them will be used to generate the new data sample. The synthetic sample will be obtained from the average of its parents and added to the final dataset. This step is repeated along the two bins until a class balance is achieved. If the two bins are completely iterated without a class balance being reached, the whole algorithm is preformed again, now with the inclusion of the synthetic data samples.

2.4 Classifiers

The process of assigning a class to an observation is described as classification, *i.e.*, through the application of algorithms that perform supervised learning on a set of data and then use the learned model to predict the classes of other data instances. With the immense variety of data that exists, no single classifier is the universal best for all data. This means that throughout the years several approaches were developed. In this section we will be exploring some of the most commonly used and most highly regarded classifiers.

2.4.1 Naive Bayes

The Naive Bayes (NB) classifier is a model based on the Bayes theorem, while assuming that features are independent. Despite the fact that feature independence is usually a poor assumption, this relatively simple model can provide comparable results to those of more sophisticated classifiers [25]. It has proven to be effective in text classification and medical diagnosis, among other practical applications.

The Naive Bayes classifier assigns the most likely class to every given example described by its feature vector. The model works in the following way:

1. Let $X = (x_1, x_2, ..., x_n)$ be an instance of the dataset with n independent features.

2. Let the final result belongs to one of K existent classes.

3. For every class $C_k \in (C_1, ..., C_K)$ the model will compute $P(C_k \mid X)$. The goal is to find the class C_k that maximizes the probability $P(C_k \mid X)$. The highest probability value is called the Maximum Posteriori Hypothesis.

4. Knowing that $P(X)$ is constant for all classes and utilizing the Bayes Theorem

$$P(X_k \mid C_k) = \frac{P(C_k \mid X_k)P(X_k)}{P(C_k)} \tag{2.6}$$

we only need to maximize $P(X|C_k)P(C_k)$.

5. This expression represents the joint probability of the two events, and is given by $P(C_k, X) = P(C_k, x_1, x_2, ..., x_n)$.

6. Since by definition all features are independent, we can use the chain rule to obtain $P(C_k, X)$:

$$P(C_k, X) = \prod_{j=1}^{n} P(x_j|C_k) \tag{2.7}$$

7. When dealing with continuous features, the probability function $P()$ is replaced by the probability density function p. Several models can be used to obtain the probability density $p(x_j|C_k)$. The most commonly used is the Gaussian model, given by the following formula:

$$p(x_j \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \tag{2.8}$$

   where μ and σ represent the expected value of x_j and the standard deviation, respectively.

8. Formulas (2.7) and (2.8) can then be used to obtain the Maximum Posteriori Hypothesis, which will give us the most likely class.

2.4.2 Decision Trees

The Decision Tree (DT) classifier [26] is based on the Tree data structure, where each node represents a test and each outgoing branch represents an outcome for the test. This structure mimics human level reasoning, which means that this model can be simple to understand even for people without a statistical or analytical background.

The Decision Tree model receives a complete dataset as well as method for attribute selection. The model can be constructed in the following way:

1. The tree is built starting on its root node. First, the attribute selection method is performed on all the features present in the dataset to formulate a logical test upon them. The feature which provides the best result on this test will be used to branch the node into two or more leaf nodes.

2. If the selected feature relates to categorical values, then the child nodes relate directly to the values of the feature. This feature is then removed from the dataset for future calculations.

If the feature relates to continuous values, the node will be split into different branches based on splitting values (for example, $\leq splittingValue0$ and $> splittingValue0$). The feature may or may not then be removed from the dataset depending on the results from the attribute selection method.

3. The algorithm is then performed recursively for every newly generated node, using only the instances of the dataset that satisfy the test of the previous branch.

4. The algorithm terminates when all the instances in each relative dataset belong to the same class, or when there are no more features to select.

5. After the tree is built, a pruning algorithm can be performed to remove subtrees according to a predetermined function, usually related with the classification accuracy.

The algorithm for this model varies slightly based on the metrics used for attribute selection. The two most commonly used metrics are Information Gain [27] (or Entropy) and Gini Index [27]. Information Gain, as the name suggests, attempts to organize features by the amount of information that is gained from each one. Alternatively, the Gini Index is used to measure inequality in a distribution of values.

This model has proven to be particularly effective at classifying medical conditions from a list of symptoms [26]. It is also known to work well with outlier values and missing data to a fair degree. However, it does not provide the best results for continuous variables and it can be prone to overfitting.

2.4.3 Random Forests

The RF classifier is based on the Decision Tree classifier. It attempts to solve Decision Tree's biggest fault while taking advantage of its biggest strengths. Essentially, this classifier constructs a group of Decision Trees (hence the name) from different variations of the same dataset in a way that assures that all the different predictions have low correlation with each other. The principle is that having a combination of good classifiers (each in their own way) will provide better results than one single very good classifier.

The Forest is created in the following way:

1. Let k be the number of Decision Trees in our forest and l be the length of the training dataset. The Bagging (or Bootstrap Aggregation) algorithm is used to create k different datasets. The way that this algorithm works is by sampling entries uniformly and with replacement from the original dataset to create the new sets. This means that, when the Bagging algorithm is complete, we will have k datasets all of size l, but all different, as each will have some duplicate entries and some missing from the original dataset.

2. After the datasets for each tree are determined, Feature Bagging is performed. This means that

each of the trees will be working with a random selection of the features (for n features in the original dataset, each tree uses about $\sqrt{n}$ features).

3. At this point the Decision Tree algorithm is performed, as described previously in the Decision Tree subsection, for each of the new sets of data.

4. In the end, all the trees try to predict the class for a given test sample, and all of them return their respective classification. All those classifications are collected and the most common result is returned as the final classification.

With Bagging, each of the Decision Trees will be trained on a different chunk of the dataset (with each chunk representing $1 - \frac{1}{e} \approx 63.2\%$ of the original dataset). This makes it so that each of the trees is more sensitive to the noise in their respective dataset, but as long as the trees are not strongly correlated, the average of all the trees will not be as sensitive, and therefore the variance decreases.

The sampling made in both the entry and feature bagging steps makes it so that each of the trees will not have complete information. Hence, any of the k trees created will almost certainly have a worse individual performance when compared to a tree built with the complete dataset. Relying on this forest will make overfitting significantly less likely than on a regular Decision Tree, where a branch could theoretically be built based upon a small subset of the data.

Random Forests come with the downsides that the model itself is not as easy to interpret as the Decision Tree classifier and also, due to the fact that several trees have to be constructed, it has a higher computational cost. However, the results can rival those of the best classifiers available.

2.4.4 K-Nearest-Neighbour

The K-Nearest-Neighbour (KNN) classifier is one of the more simple classifiers to understand and is a go-to for classification studies performed with little to no prior knowledge of the dataset [28]. For every test instance, the model will determine the k closest instances in the training dataset and output the most common class out of those k instances. In order to determine the k most similar instances a distance metric will be used, with the most commonly used metric being the Euclidean distance. It is highly recommended to normalize all the values for each feature beforehand, although this is not an obligatory step.

While the KNN model is intuitive and easy to understand, it can be somewhat inexact, when compared to others. The model is sensitive to outlier values and does not address missing data and imbalanced datasets in the most optimal way [28].

2.4.5 Logistic Regression

The Logistic Regression (LR) model is similar to a Linear Regression one, with the main difference being that it is solely used to predict categorical data. It is typically used to predict binary data, although it can be modified to predict data with more than two categories. This model reaches a prediction by utilizing a *logit* (or logistic unit): a function that maps values from $[0, 1]$ to $(-\infty, +\infty)$. The algorithm works in the following way:

1. Let Q denote the probability of a given instance X belonging to class 1 ($Q = P(X = 1)$). Since we are dealing with a binary classification problem, the probability of the instance belonging to class 0 is given by $1 - Q$. We then have that the logit function l is given by:

$$l = \log_b \frac{Q}{1-Q} = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n \tag{2.9}$$

   Where x_j represents the j^{th} feature of sample X_i, β_j represents the model's parameter for said feature and n represents the amount of features in the dataset.

2. By performing some algebraic manipulation, we obtain:

$$Q = \frac{1}{1 + b^{-(\beta_0 + \beta_1 x_1 + ... + \beta_n x_n)}} \tag{2.10}$$

3. From equation (2.9) we can see that l is given by a polynomial function. We can take advantage of that knowledge and determine the values for each β_j in a similar fashion to that of a linear regression. From there we can determine the value for Q, and thus we get a classification.

Logistic Regression is a good approach to binary classification, as it can output accurate results with little computational costs. That said, it starts to falter when using a dataset with a number of features that is too large or when dealing with missing data.

2.4.6 Support-Vector Machine

Support-Vector Machine (SVM) are models used for linear binary classification of data. SVMs first map the training-set data as points in high-dimensional space [29] and then determine the hyperplane that separates the two classes, maximizing the margin between them. To perform the mapping from the original feature space to the high dimensional space, SVMs utilize non-linear transformations called *kernels*. The most typical kernels are:

- Polynomial: $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$
- Gaussian: $K(x_i, x_j) = \exp\left(\frac{||x_i - x_j||^2}{2\sigma^2}\right)$
- Radial Basis Function (RBF): $K(x_i, x_j) = \exp\left(-\gamma ||x_i - x_j||^2\right)$

Where $K(x_i, x_j) = \varphi(x_i) \cdot \varphi(x_j)$, d is the polynomial degree and σ and γ are modifiable kernel parameters. The Gaussian and RBF kernels are the preferred default kernels in situations without prior knowledge of the dataset, while the Polynomial kernel can be good when processing images.

Hence, let $z_i = \varphi(x_i)$ denote the mapping that transforms any point $x_i \in R^n$ to $z_i \in Z$. The optimal hyperplane is then given by:

$$w \cdot z + b = 0 \tag{2.11}$$

where w is a weight vector and b is the bias. We have that D is linearly separable if there exists a pair (w, b) such that:

$$\begin{cases} (w \cdot z_i + b) \geq 1, & \text{if } y_i = 1 \\ (w \cdot z_i + b) \leq -1, & \text{if } y_i = -1 \end{cases} \forall\, 1 \leq i \leq l \tag{2.12}$$

is true for all instances in dataset D. We can rewrite (2.12) to look like:

$$y_i(w \cdot z_i + b) \geq 1 \tag{2.13}$$

The equations in (2.12) give us H_1 and H_2, which are our dividing planes. H_0, given by (2.11), is the median plane between H_1 and H_2. To obtain the optimal hyperplane, we need to maximize the margin of separation between the median plane and these boundary planes. Maximizing the distance between H_1 and H_2 is equivalent to minimizing $||w||$ [30]. This minimization, in conjunction with the constraint in (2.13), gives us the general optimization problem required to obtain a hard-margin.

Generally, SVMs can be particularly accurate when dealing with high dimension data, such as is the case for medical data. They can, however, be prone to overfitting, especially when dealing with high dimensional datasets with a smaller amount of samples. They also struggle when working with unbalanced datasets.

2.4.7 Neural Networks

The Artificial Neural Network (NN) is a model loosely based on the design and behaviour of animal brains. The model is made up of a collection of interconnected units called Neurons. The idea behind this classifier is that each neuron by itself is very limited and only capable of a limited set of behaviours, but a network of these can become extremely complex and great at finding patterns.

Neural Networks are organized into layers of neurons, with each network being composed of two required layers (input and output) and any number of hidden layers.

Every neuron n_j not located in the input layer holds a set of weights w_{ij}, each related to the connection between the node n_j and a node n_i from the previous layer. These weights are what neurons use to determine their respective output value:

$$X_j = f(\sum_i w_{ij}x_i + B_j) \tag{2.14}$$

where x_i is the output value of node n_i, w_{ij} is the weight associated with the connection between nodes n_i and n_j, B_j is the bias value associated with node n_j, and f is a previously defined function used to transform or limit the result of the net sum inside it. To perform a classification, each neuron in the neural network determines their output, using formula (**2.14**), layer by layer from input to output.

To perform classification, the model needs to first be constructed and then trained. As for the construction of the model, the number of layers and the number of neurons per layer is up to the person constructing it. Different structures will inherently lead to different results, and figuring out the best structure for each problem can be a tiresome process and can eventually come down to trial and error.

As for training the model, the most commonly used method is called back-propagation. This method consists of updating the weights of all neurons sequentially, from the output layer to the input layer. Each of these updates takes into consideration the error between obtained and expected values, as well as the weights of the neurons in layers ahead.

Neural Networks do not necessarily have a predetermined stopping criterion, they will train on the entire dataset for as many times as they are told to. For this reason, Neural Networks are extremely likely to overfit the data unless a validation dataset is used to prevent it. This classifier is also difficult to interpret as each neuron holds little information and the model as a whole can be too complex. That said, Neural Networks are incredible versatile as they tolerate all types of data, and can handle noisy data extremely well. They are also great at finding patterns in the data where most other classifiers fail.

2.5 Metrics and Validation

When dealing with data classification problems, an element that cannot go overlooked is that of classification metrics. With this work aiming to obtain the best classification results, it is pivotal to determine how to rate and compare results.

To choose the set of metrics to utilize, it is important to first understand what information we want to get from them. Different metrics evaluate different characteristics of a classifier [31]. It is also necessary to understand the data being used, as classifiers can be binary or multi-class, and the appropriate metrics must be picked for each. Ideally we would want a set of metrics that provide us with results that are informative and which ultimately will allow us to determine the best overall model.

With all this in consideration, and knowing that we are performing binary classification, the metrics that were chosen to evaluate our results are the following:

2.5.1 Accuracy

Accuracy is by far the most commonly used metric in classification problems, as it is easy for humans to understand and can be used with both binary and multi-class problems. It gives us the ratio of correct predictions to the size of the dataset and is defined as:

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{2.15}$$

where tp refers to the true positive values (*i.e.* the number of correctly predicted positive values), tn refers to the true negative values, fp refers to the false positive values and fn to false negative values.

In addition to being the easiest metric to understand and compare, accuracy also has a rather small computational cost. On the downside, accuracy returns less differentiated values than other metrics do. By grouping false positive and false negative values, it does not provide enough information on the classifier's behaviour. It can also disproportionately favour classes in the presence of an imbalanced dataset.

2.5.2 Sensitivity and Specificity

Sensitivity (also known as recall or true positive rate) and specificity (also known as true negative rate) are two binary classification functions, and they're respectively defined as:

$$Sensitivity = \frac{tp}{tp + fn} \tag{2.16}$$

$$Specificity = \frac{tn}{tn + fp} \tag{2.17}$$

Knowing the values of these two metrics is of great importance when dealing with medical data, as they can provide more information on the result of a classification. In fact, these metrics make up for the aspects where the accuracy metric lacks, by providing more differentiated results and by presenting constant results regardless of the dataset being balanced or not.

2.5.3 Area Under the ROC Curve

The Receiver Operating Characteristic (ROC) Curve is a graphical plot obtained by computing the sensitivity against the false positive rate $(1 - Specificity)$. This graph allows us to visualize and compare the sensitivity and specificity of two different models. In some cases, visual comparison of the two curves can be enough to determine the best classifier. However, in the cases where this is not evident, it is more accurate to compute the Area Under the Curve (AUC) and draw conclusions based on that. The AUC is given by:

$$AUC = \frac{S_p - n_p(n_n + 1)\frac{1}{2}}{n_p n_n} \tag{2.18}$$

Where S_p is the sum of all instances ranked as positive, n_p is the number of actual positive instances and n_n the number of actual negative instances. The value that this function returns can be interpreted as the probability of a randomly picked negative instance having a lower estimated probability of belonging to the positive class than a randomly picked positive example [32].

This metric comes with some downsides, like the possible noisiness of the generated graph, or the higher computational cost it requires, compared to that of the accuracy metric. However, it does fulfill some of the requirements we set up: it takes into consideration sensitivity and specificity, which provide more discriminate results; it always returns a value between 0 and 1, with 0.5 being the result for an uninformative classifier, which makes the results easier to comprehend; furthermore, the AUC metric has been shown to be better than the accuracy metric at evaluating and comparing the performance of classifiers [31] [32].

2.5.4 K-Fold Cross Validation

After determining the metrics we want to use, we are ready to compare and evaluate our models. To do so, we need to divide our dataset into a training set and a testing set. Each model will then be trained on the training set and evaluated based on the data in the test set. However, machine learning models are severely influenced by the data they're trained on, so the way in which we divide the dataset can have a big impact on the final results.

To safeguard this we use K-Fold Cross Validation. This method consists in dividing the dataset into different smaller groups and utilizing different combinations of them to test each model. It can be executed as follows:

1. The dataset is randomly divided into K groups of equal size.

2. One of the groups is chosen to be used as the testing set, while the other $K-1$ groups are joined to be used as the training set.

3. The model is trained on the training set and tested (using the desired metrics) on the testing set.

4. The testing results are stored and steps **2-3** are repeated, using a different group as the testing set. In total, these steps will be executed K times, with each group being used as the testing set exactly once.

5. In the end, a list with the scores for each testing iteration will be returned. From the list, values such as the average score and standard deviation can be determined.

This approach will help reduce some undesired variability that could appear from a random training/testing set division. It is also important to determine an appropriate value for K, as a poorly chosen value can influence the final results. A K value that is too small can provide results with high variance. Alternatively, a K that is too large can result in high bias. This value will depend on the dimensions and variance of the datasets.

2.6 Machine Learning in Alzheimer's Disease problems

This thesis is far from the first attempt at tackling problems surrounding Alzheimer's Disease with Machine Learning techniques. Numerous studies have applied ML techniques to determine the diagnosis or prognosis of AD. Each of these studies can focus on entirely different sources of data. The data utilized in these studies can be categorized as follows:

- Imaging data - The bulk of these studies relating to AD utilize imaging data. More commonly the imaging data corresponds to Magnetic Resonance Imaging (MRI) scans of the patients' brains, although the analysis of Positron Emission Tomography (PET) scans can also be useful in these problems [33]. These scans can be directly interpreted by machine learning setups or they can go through a pre-processing step beforehand. Beyond the traditional two dimensional imaging data, there have been studies performed on three dimensional data from MRI scans [34].

  The abundance of studies utilizing imaging data can be partially attributed to the robust databases of MRI data publicly available, such as the Alzheimer's Disease Neuroimaging Initiative (ADNI) [35] or the Open Access Series of Imaging Studies (OASIS) [36].

- Clinical data - This category can include data retrieved from a number of laboratory tests, which can be converted into numerical or categorical data. Studies have been performed on differently sourced data, such as Cerebrospinal Fluid (CSF) biomarkers [37] or Electroencephalography markers [38].

- Neuropsychological data - This type of data is collected through Neuropsychological tests (NPTs). These are typically pen-and-paper tests that include ways to test a patient's cognitive ability, such as language or memory tests. This is the type of data used in this thesis and more information regarding these tests will be explored in Section 4.1.1. There have been some similar studies performed with these tests. In [39] for example, NPTs were used to distinguish between AD or other forms of cognitive impairment.

- Mixed data - It is not uncommon to see studies that rely on distinct types of data as these data types are not interchangeable and can all provide important and useful information. Each different source of data can help by providing information that would not be obtainable exclusively with the remaining sources. Some examples of studies performed with this type of data include [40] where multiple dataset combinations were tested with data obtained from MRI scans, CSF and PET biomarkers, in order to predict the future decline in MCI patients. Furthermore in [41] a risk classification for developing AD is performed utilizing data from Neuropsychological Tests (NPTs), CSF biomarkers, and MRI scans.

While these papers differ in methodology, data types and sources of data, they can also differ in another important aspect, their goal. There should be a distinction made between these approaches when it comes to the problem that they are trying to solve, between approaches that search for a diagnosis and those that search for prognosis. Diagnosis is used to identify a patient's present illness or condition, where as prognosis is used to predict the course of an illness or condition. Typically a prognosis always comes after a diagnosis.

Several of the articles previously mentioned, such as [37] [38] [39], focus on the diagnosis of AD or dementia, with fewer focusing on prognosis. As these are entirely different problems, approaches used to solve one can't directly be used to solve the other. There has been work done on predicting the evolution of patients with AD or MCI, such as [42] and [43] where machine learning and deep learning models respectively are trained on multimodal neuroimaging data.

However, when it comes to both utilizing Neuropsychological data and using it to predict the evolution of AD (or MCI to AD specifically) the amount of published studies is significantly smaller. This highlights the importance of the work developed in [1], upon which this thesis attempts to build.

2.7 Summary

In this chapter we studied the background information related to Machine Learning, specifically the subjects that matter to this work. Initially we described the basic setup of a data mining pipeline and the steps it requires: Data cleaning, feature selection, missing value imputation, balancing classes, classification and evaluation. We elaborated upon this by studying what each step does and what different solutions there can be to solve them. Then we explored state-of-the-art techniques that exist to address each of those steps.

Finally, by the end of the chapter, we explored what other machine learning approaches have been developed to address similar problems. These similar problems were then categorized according to their goal and the type of data they use.

3

Evaluating Missing Value Imputation Methods

Contents

Missing data can be classified as one of three different types [44]. The first is Missing Completely At Random (MCAR), which occurs when the subjects with missing data are a random subset of the complete set of subjects and the data that is missing is not dependent on other data entries. An example of this would be a patient whose test results got lost before being registered.

The second missing data class is Missing Not At Random (MNAR) data, which occurs when the information is missing due to a non-observable factor, such as the result of the missing information. It could happen, for example, that a patient who had scored high on a depression exam would not be comfortable sharing the results of the exam. The problem with this type of missing data is that, by nature, it is rather difficult to know whether any missing data is MNAR or not.

The third and final class corresponds to Missing At Random (MAR) data. In this case the lack of information can be traced to observable factors. A good example for this would be the lack of a specific exam used solely to detect Alzheimer's Disease on a 50 year old patient. This represents the most common type of missing data [44].

Knowing that the majority of missing data is MAR, we can deduce that utilizing strategies like the sample mean for imputation can result in biased values. A clear scenario where this could happen is in a data-set that has imbalanced classes. In this scenario, the imputed values would be biased towards the most common class, which would influence the classifier learning the data. For this reason, when dealing with data-sets missing data in a substantial amount of parameters, it is much preferred to use MVI methods based on more complex models.

In this chapter we will explore and evaluate state-of-the-art models to address MVI. We will then compare the results obtained from these models and the best performing ones will be selected and utilized in the following chapter, to address the main problem of this thesis.

3.1 Overview of MVI Methods

MVI models can be divided into two categories: discriminative and generative. Discriminative models merely model the decision boundaries between different classes while generative models focus on learning the probability distribution of the individual classes. Considering inputs x and class y, Discriminative models learn the conditional probability distribution $(p(y|x))$, while generative models learn the joint probability $(p(x, y))$. This property also allows generative models to calculate the posterior $(p(y|x))$, by applying the Bayes rules to the learned probability distributions [16].

Considering the problem of mapping input x to a given class, discriminative models are generally more efficient, as they tend to require less computational power. However, generative models acquire deeper understanding of the data distribution, which in turn can be an advantage for problems such as the one of generating new data samples.

3.1.1 Discriminative Methods

3.1.1.A Overall Sample Mean

Being one of the simpler methods, it has become somewhat of a standard, or a base method to be used when comparing different MVI methods. The Overall Sample Mean (OSM) method consists on filling in missing data with the overall mean value for each parameter (for continuous data) or with the most common value (for categorical data). If the categorical data is ordered, then something like the rounded mean value should be used instead.

Unless it is being used on a dataset with a small amount of features or to input mostly MCAR data, the OSM method is generally worse than other methods based on regression. Besides introducing bias to the imputed data, it reduces variance. It can also "dilute" the value of a given feature, as having multiple entries with the same value will bring the feature to non-significance for classifiers. This effect can be more or less apparent depending on the classifier being used [44].

3.1.1.B KNN-Impute

The K-Nearest Neighbours Impute (KNNI) method is one of the most widely used methods for missing data imputation. It is also one of the most worked upon methods, with several new MVI models having been developed on top of the KNNI foundation, namely in biological fields [45] [46]. The KNNI method is based on the K-Nearest Neighbours algorithm described in **2.4.4**. It works in the following way:

1. First it receives a dataset with some missing data, as well as a number of "neighbours" k to utilize.

2. For every data entry e with a missing value (for a given feature f), the method will run through all the data in the dataset searching for the k most similar entries. To determine the most similar entries, a distance metric is used (usually the euclidean distance, or a variation of it).

3. The distance comparison between entries is performed with their values for all of the features. If one of the entries that is being compared to the original entry e happens to have a value missing that e doesn't, it will be discarded in this particular comparison.

4. When the k most similar data entries are found, their values for the missing feature f will be used to impute this value. The most common value will be chosen in the case of a categorical feature, or the average between the k entries in the case of a continuous feature.

5. In the case that there are no k similar results without any missing data and the criterion in **3.** isn't met, then the average value for the feature in the entire dataset will be chosen.

6. These steps are repeated until there are no more samples with missing data in the dataset.

The KNNI has the particularity of being one of the fastest classifier-based MVI methods at imputing data. It does have some limitations and can struggle when dealing with datasets with large amounts of missing data. It should not, however, perform worse than the OSM method described previously.

3.1.1.C MissForest

MissForest (MF) [47] is a method for missing value imputation that is prepared to handle mixed-type data (continuous and categorical) simultaneously. The method is based on Random Forests, which also can handle mixed-type data and can work under adverse conditions such as high dimensional data. The MissForest method trains a Random Forest on the available data, similarly to how it is explained in section 2.4.3. It then proceeds to calculate a prediction for the missing values. It then takes that prediction in conjunction with the original dataset and repeats this cycle iteratively until a stopping criterion is met. This approach allows it to make as few assumptions about structural aspects of the data as possible. This method has proven to be comparable to, if not better than, the KNNI and MICE methods when dealing with categorical and continuous MCAR data [47].

3.1.1.D MICE

Multiple Imputation by Chained Equations (MICE) [48] is one of the leading methods for MVI designed solely to address MAR data. This method is known for computing multiple completed datasets and cross-analyzing all the imputed values in order to choose the best one. Its process can be summarized in the following way:

1. Initially a basic imputation, such as the overall mean, is performed on every missing value. These values are meant to be plausible and are not final.
2. One of the imputed values in the previous step is returned to its initial empty state.
3. One or more regression models are built using some (or all) of the variables in the dataset, not counting the variable with the missing value from step 2. The models are then used to predict the missing value of the aforementioned variable.
4. The missing value is filled in with the output from the chosen regression model.
5. Steps 2 to 4 are then repeated for every value initially imputed in step 1.
6. This whole process is repeatedly performed on the entire dataset as many times as the user so determines.

During the execution of the algorithm, previous and current results are constantly pooled and analyzed to determine the best or the preferred approach.

The fact that the MICE method develops different classifiers for each feature, makes it great and instantly better than most methods at dealing with mixed data. Moreover, knowing that the majority of medical data is MAR and knowing that this method was constructed specifically for this type of data, makes MICE one of the best approaches to this type of problem.

3.1.2 Generative Methods

3.1.2.A Expectation Maximization

The Expectation-Maximization (EM) algorithm is a method for density estimation of a dataset according to the probability distributions of its parameters. Maximum Likelihood Estimation (MLE) is a statistical method of determining parameter distribution within a dataset. However, the MLE method works exclusively under the assumption that the dataset is complete. For cases where not all the relevant variables can be observed, the MLE method is ineffective. The EM algorithm allows for the calculation of this distribution in datasets with latent variables.

The EM algorithm is structured in a cycle of two iterative steps, the E-step and the M-step. The E-step (or estimation-step) attempts to estimate any latent variable in the dataset. While the M-step (or maximization-step) attempts to optimize the model's parameters in order for them to best describe the data. These optimized parameters are then utilized to determine the distribution of latent variables in the following E-step. These steps are performed iteratively until an accurate distribution of the dataset is achieved.

Since this algorithm is capable of learning the distribution of a dataset it can reasonably be adapted or incorporated to generate data for missing values in a dataset [49]. In [50] a method utilizing the EM algorithm is used to handle missing values in the data. However, the method in this paper fails to generalize well when the datasets contain both categorical and continuous data.

3.1.2.B Autoencoder

Autoencoders (AEs) are Neural Networks that are trained to learn a latent (typically lower dimensional) representation of data. These networks transform the input into an internal code which can then be roughly reconstructed into the original input. Autoencoders consist of 4 main components:

- Encoder - The component responsible for deconstructing and compressing the input into the model's latent representation.

- Internal code - The model's latent representation of a given input. Usually a lower-dimensional representation of the input data.

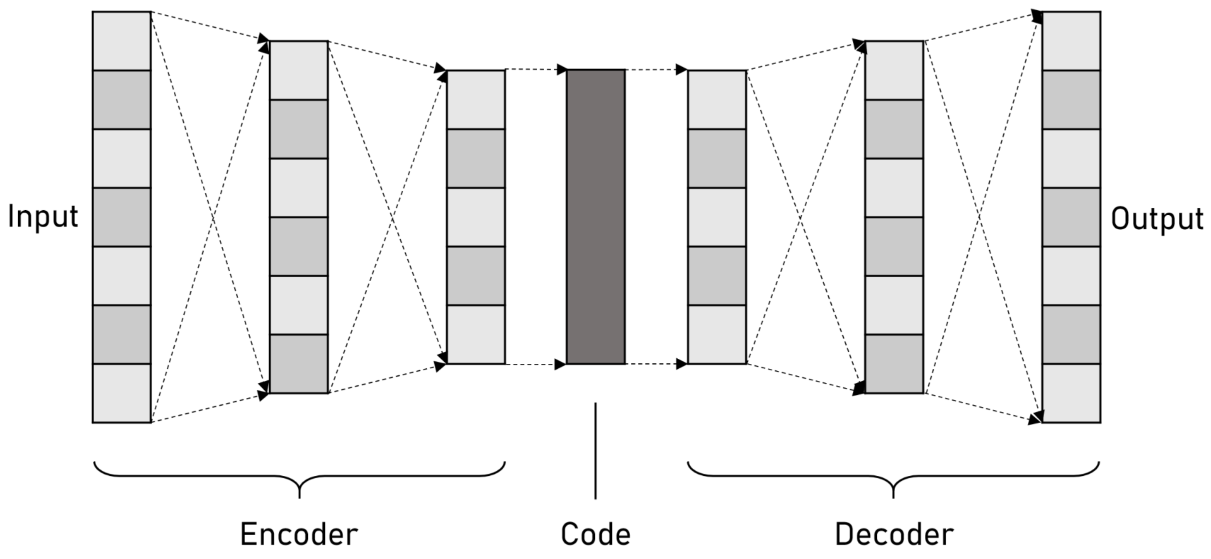


Figure 3.1: Diagram depicting the architecture of an autoencoder

- Decoder - The component responsible for reconstructing the original input from the latent representation.
- Loss function - The function used to compare inputs and outputs during the training process. Usually the Mean Squared Error (MSE) or the Binary Crossentropy (BCE) are utilized.

The encoder and decoder both consist of consecutive layers of connected neurons (typically with symmetrical architectures). The learning process is performed through backpropagation, similarly to other Neural Network models. During this process both the encoder and decoder are adjusted.

Autoencoders perform unsupervised learning, as they don't require any explicit label for the training process. The entire deconstruction and reconstruction process is very data-specific, meaning that this model will only be accurate for data similar to the one used in its training process.

Figure 3.1 depicts the structure of an autoencoder, where each column of squares represents a layer of neurons. The number of layers, number of neurons per layer and dimension of the internal code can all be adjusted according to the use case.

Autoencoders were initially applied on problems of dimensionality reduction and feature learning. However, more recently, exploiting the model's inferred representation of the data has proven to be useful for addressing the problem of imputing missing data. In [51] a solution based on the autoencoder architecture was constructed to address MVI, outperforming state of the art discriminative models. To further address this problem, some specific variations of the AE model emerged, namely Denoising Autoencoders.

Denoising Autoencoder (DAE) represent a stochastic version of AEs. This model's core idea is to introduce noise to the training data and have the AE recover the original noise-free data. This forces

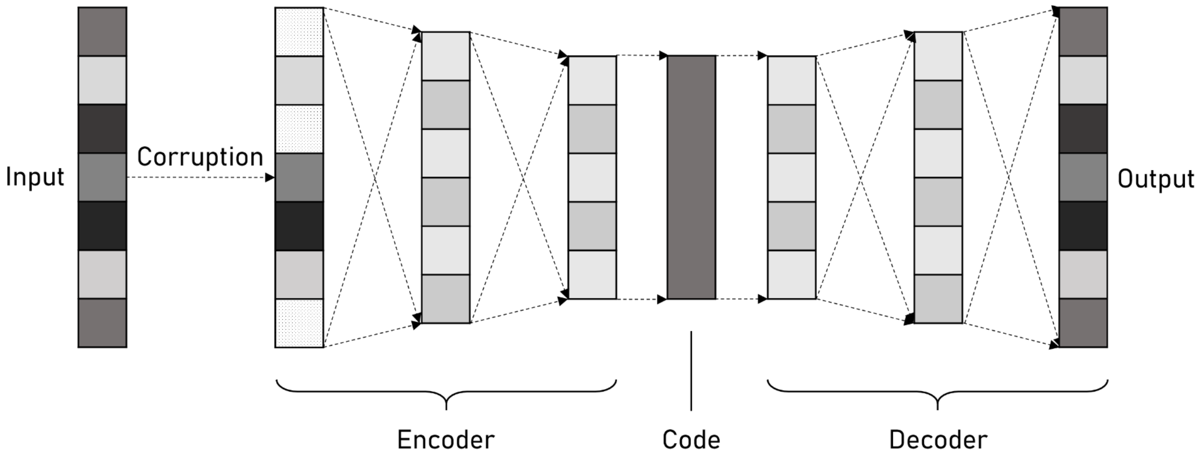


Figure 3.2: Diagram depicting the architecture of a Denoising Autoencoder

the AE to learn more robust features, preventing it from attributing too much importance to only some of them. These models don't require any structural modification to the typical AE architecture, aside from the addition of a data corruption step in the beginning of the training process. With this change, the loss function utilized during training is now computed between the output and original noise-free inputs. Figure 3.2 shows the architecture of a DAE, including the addition of the corruption step in the beginning.

The idea of removing noise from data can be applied to missing value imputation problems by a simple paradigm shift. By perceiving the missing values as corruption in the data, and by perceiving complete data as noise-free, we can adapt DAEs to impute those missing values. In fact, DAEs have already been utilized to solve such problems. In [52] a Denoising Autoencoder is utilized to impute missing data for DNA and RNA sequences. In [53] an approach based on DAE was also shown to work well in practice, although it required complete data during training. Other approaches like [52] and [54] allow for incomplete datasets, utilizing only the observed components to learn the representations of the data. In both these cases, real life datasets are utilized and the obtained results are comparable or outperform those obtained from discriminative models.

3.1.2.C Variational Autoencoder

The AE architecture, described previously, explains how the model can reconstruct a given input, however this architecture as it was described is not sufficient to generate new data values.

One approach to generate new data values, could be to sample a random vector from the latent space of a trained AE and feed the vector to the Decoder. In an AE with regular latent space, the Decoder would output a plausible sample analogous to the training dataset. However, depending on the distribution of data in the training dataset and on the Autoencoder's specific architecture, its latent space can often be irregular. AEs typically present a bottleneck architecture, where the internal code

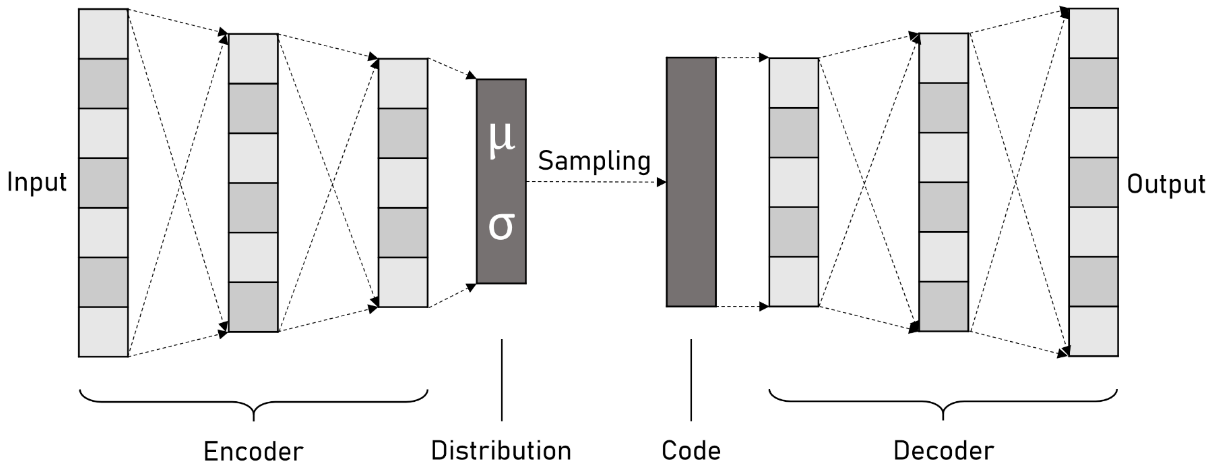


Figure 3.3: Diagram depicting the architecture of a Variational Autoencoder

has smaller dimensionality. This leads to a loss of information and can cause severe overfitting, which in turn results in an irregular latent space.

Variational Autoencoder (VAE) address the problem of latent space irregularity by having the Encoder return a distribution over the latent space as opposed to a single vector. This distribution can be enforced to resemble a normal distribution which allows for a regularized latent space. The inclusion of this step entails the addition of a regularisation term to the VAE's loss function, which can be derived using a Bayesian inference method called Variational inference. Figure 3.3 depicts the architecture of a VAE, with this extra-step between the Encoding and Decoding processes.

Some work with VAEs has already been performed in the field of MVI. In [55] a model based on VAEs and Importance-Weighted Autoencoders was proposed, performing both single and multiple imputation. The presented model returned results that outperformed some state of the art methods on popular real life datasets. In [56] another VAE-based model was explored to impute data on a real life dataset. It ended up presenting better results than some discriminative models on both high and low corruption data.

3.1.2.D Generative Adversarial Network

Generative Adversarial Network (GAN) are a relatively recent concept in the world of Deep Learning. The idea consists of facing two Neural Networks against each other, in order to generate credible synthetic data samples. We refer to the two Neural Networks that make up this model as Generator and Discriminator. Figure 3.4 depicts the general structure of a GAN. The Generator receives an input vector and uses it as a base to generate new data samples. The Discriminator is then given a vector with both real data and the generated samples, and is tasked with differentiating between them. Both these networks learn through the same backpropagation process, in a double feedback loop. This allows the

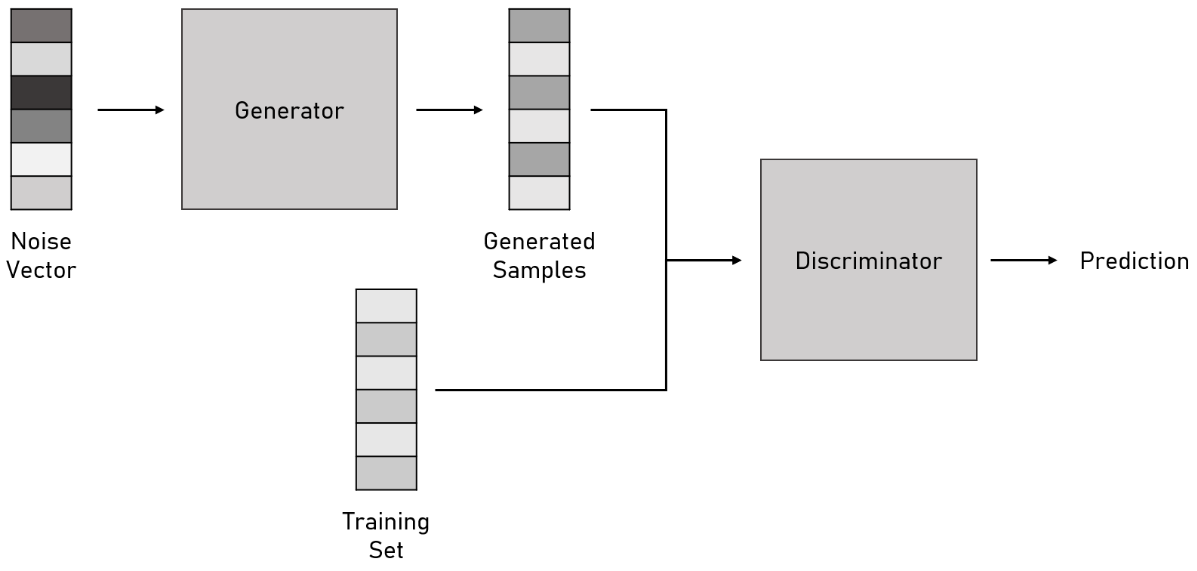


Figure 3.4: Diagram depicting the architecture of a Generative Adversarial Network

Generator to create increasingly more credible samples, while the Discriminator gets better at identifying them. The training process terminates once the Generator is deemed capable of deceiving the Discriminator.

This model's architecture allowed it to address missing value imputation problems by utilizing the generator to create values to replace the missing data. In [57], an approach based on GAN is utilized to impute missing values in medical data. In [58] several models that address MVI based on GANs are studied and compared. Additionally, in [59] another GAN-based model is proposed and shown to have competitive results with other state of the art generative and discriminative methods.

3.1.2.E Recurrent Neural Networks

Recurrent Neural Network (RNN) have a cyclical architecture, such that for each classification problem, both the current input and the previous outputs are considered. Among the models utilizing this structure, the Long Short-Term Memory (LSTM) model is the most well-known. This model is called LSTM due to its ability to "remember" past data without losing too much importance for each classification iteration. This model is trained by back-propagation, similarly to how it is described in section 2.4.7. Utilizing past outputs in the classification process allows RNNs to model sequences in data. LSTMs can be particularly useful for addressing missing values in datasets where the data is organized in a sequential manner, such as temporal data or speech recognition data. In [60] and [61] imputation methods based on LSTMs are utilized to tackle missing value imputation on temporal and textual data, respectively. In both papers the LSTM-based solutions present results comparable with the discriminative alternatives.

3.2 Experimental Setup

This section describes the MVI methods that were selected to be tested and their individual configurations. It also describes the datasets and metrics that were utilized to test those MVI methods.

3.2.1 Imputation Methods

Applying the knowledge acquired previously, this setup utilizes 6 models for missing value imputation: A Denoising Autoencoder, a Variational Autoencoder, a Generative Adversarial Network, MICE, MissForest and OSM. These models are configured as follows:

3.2.1.A Denoising Autoencoder

The DAE used is similar to the one described in the MIDA paper [54]. The encoder and decoder present an atypical architecture, where they are composed of 3 layers each, with an increasing number of neurons in successive hidden layers. This amounts to the first encoder layer having the same amount of neurons as there are features in the dataset (n), and each successive layer (l) having an additional $l \times \Theta$ neurons (*i.e.*, $n + (l \times \Theta)$ neurons per layer, considering the first layer as layer 0). In our tests, we utilized $\Theta = 7$, which implies that the internal code will have a dimension of $inputsize + (2 \times 7)$. Finally, $Tanh$ is utilized as the activation function.

The model is trained for 500 epochs, with the help of an adaptive learning rate with a time decay factor of 0.99 and Nesterov's accelerated gradient [62]. Mean Squared Error is used as the Loss Function.

3.2.1.B Generative Adversarial Network

The GAN used is similar to the one described in the GAIN paper [59]. The Network utilizes reconstruction error in the loss and a hint vector. In both the Generator and Discriminator, the sigmoid function is utilized as the activation function.

The values for the hint rate h and loss hyper-parameter α were obtained through grid search, on the datasets reserved for testing (3.2.3). Their final values were 0.9 ($h \in [0.1, 0.9]$) and 10 ($\alpha \in [10, 100]$).

The model is trained for 2000 epochs, with a single random batch of data being chosen for training with each epoch. Each batch has a size of 64.

3.2.1.C Variational Autoencoder

The VAE used is similar to the one described in the MIWAE paper [55]. The encoder and decoder are composed of 3 layers each. The first encoder layer has the same amount of neurons as there are features in the dataset (n), while the following two layers have 128 neurons each. The model uses a deep latent variable model with a Gaussian prior. The encoder's output has only 2 dimensions, consisting of the mean and diagonal covariance, which allow for the computation of the distribution.

The Decoder's architecture mirrors the Encoder's, with the exception of the output size. For each dimension in the Encoder's input, the Decoder will output 3: The mean, the scale and the number of degrees of freedoms. These values will then be utilized to compute the loss and generate new data samples. $ReLU$ is utilized as the activation function since it leads to faster training, despite $Tanh$ having been used in the paper.

The model is trained for 2000 epochs, with a single random batch of data being chosen for training with each epoch. Each batch has a size of 64.

3.2.1.D MICE

The MICE method [48] runs for a maximum of 20 iterations. All the default values for its other parameters are used.

3.2.1.E MissForest

The MissForest method runs for a maximum of 10 iterations. It utilizes 100 trees during the training process and it uses RMSE as a loss function. All other parameters utilize their default values.

3.2.1.F OSM

With this method, the mean value for the feature is used.

3.2.2 Early Stopping Criterion

The three Generative methods (DAE, VAE and GAN) use the same early stopping criterion. Each training process utilizes 95% of the training data provided for actual training, saving the remaining 5% as a validation set. With each training epoch, the model attempts to generate values for the validation set, after which the respective RMSE is computed and stored. The training process reaches an early stop if 30 epochs elapse without the validation error decreasing.

The amount of epochs utilized as the early stopping criterion was obtained through grid search ($\#epochs \in [20, 50]$).

3.2.3 Datasets

Considering our goal of testing the different generative methods, it was of great importance that we collected a group of diverse datasets. Our selection ranges from sets with high numbers of observations and attributes to sets with very few observations. Notably, we selected datasets of greater, smaller and similar sizes to the Alzheimer's Disease prediction datasets, on which this thesis focuses. All these datasets are complete, with the majority containing a mix of continuous and categorical data.

Deep Learning Models are known for providing more accurate results on datasets of greater dimensions while struggling with smaller ones. The diversity in these sets allows us to assess the robustness and versatility of the methods being evaluated.

All these datasets contain real data and were obtained from public online platforms. The names, initialisms and dimensions of the 10 datasets utilized are present in Table 3.1

Table 3.1: Datasets utilized for testing. Datasets initialisms are displayed in parenthesis, for future references

Datasets	Observations	Attributes
World Happiness Report (WH)	156	10
Heart Disease UCI (HD)	303	14
Boston Housing (BH)	506	14
Pima Indians Diabetes (PI)	768	9
Breast Cancer Wisconsin (BC)	569	30
Red Wine Quality (RW)	1599	12
Gender Recognition by Voice (GR)	3168	20
Predicting a Pulsar Star (PS)	17898	8
House Sales in King County (KC)	21613	17
UFC-Fight Historical Data (UF)	3582	158

3.2.4 Metrics

In order to compare results, the Root Mean Square Error (RMSE) function will be utilized. The generative methods previously described can generate new values for data which was previously existent. However, for the following experiments, the comparison will only be performed between values which were initially missing and their respective generated outputs.

3.3 Results

In this section, our aim is to evaluate the performance of the different generative methods for missing value imputation previously explored. Additionally it is useful to learn what limitations these methods may have and which scenarios allow them to obtain the best results. The experiments were structured as follows:

3.3.1 Influence of the Initial Imputation

3.3.1.A Description

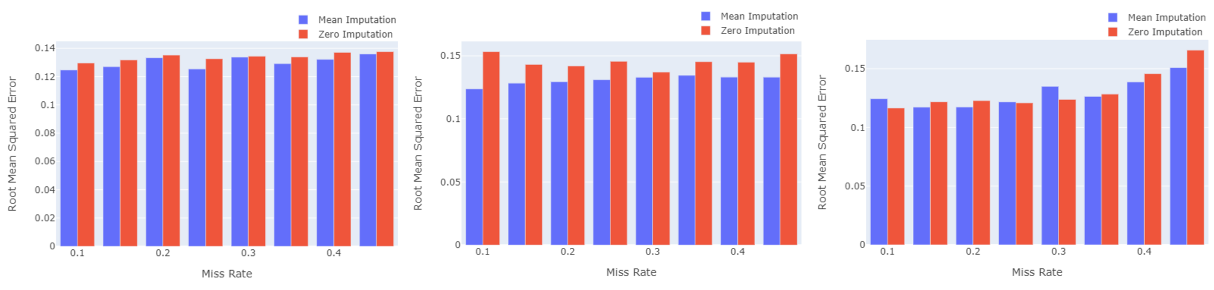
In the first set of experiments we analyze how each generative model can be individually improved. First, grid search was performed on all the models in order to determine the values to some of their internal parameters. The search was performed on all datasets with a fixed rate of missing data (missing rate $= 0.2$). These results were already presented in section 3.2.1.

All the methods were described in their respective papers as utilizing Zero-Imputation. This means that, during the imputation process, all missing values are initialized with the value zero. With this information, we saw an opportunity to test whether using Mean-Imputation initially would provide better results.

To test this hypothesis, the three generative methods were utilized as well as the ten datasets. For each method-dataset pairing, eight measurements were performed, with the amount of missing data in the range of 10% to 45%, with intervals of 5%. In each of these measurements, both Zero-Imputation and Mean-Imputation were performed, and the RMSE value for each imputation was stored. For each of these measurements 5-Fold Cross-Validation was used. Furthermore, the data corruption introduced to each dataset was identical when using either one of the initial imputations.

3.3.1.B Results and Discussion

The experiments were grouped according to the generative methods for MVI and datasets utilized. These variables allow us to compare how the results from the initial imputation method can be affected by the imputation method used, the dataset dimensions and the data missing rate. In Figure 3.5 we can see the results for each of the generative MVI methods on the Red Wine dataset.



(a) MIWAE **(b)** MIDA **(c)** GAIN

Figure 3.5: RMSE for imputation performed by the three generative methods (with architectures based on VAE, DAE and GAN respectively) on the Red Wine Quality Dataset (RW) for different missing rates. In blue are the results when utilizing Mean-Imputation initially and in red the results for utilizing Zero-Imputation initially

According to the results obtained we can withdraw the following conclusions:

- Dataset Size - The size of the dataset did not appear to have any major influence on the results observed (see Appendix A for test results obtained from all datasets). Even so, it is possible to note that, as the dataset's dimensions increase, the difference in the error obtained from using either of the methods decreases. This might not be a direct influence of the initial imputation method, yet we can conclude that the method chosen for it is not as important as the dataset's size increases.
- Missing Rate - The Zero-Imputation method's results, when compared to the Mean-imputation ones, improved as the missing rate increased. This means that, in test cases where the Mean-Imputation method provided overall better results, the difference in error would decrease with the increase of missing data. Or in the case that both methods performed comparably overall, the Zero-Imputation method would have the better results for higher values of missing data. This can be justified by some of the consequences we explored on 3.1.1.A regarding Mean Imputation.

Overall, the difference in results was never overpowering, nevertheless both the MIWAE and MIDA methods performed better with the use of Mean-Imputation,on average. The GAIN method, on the other hand, performed identically with either of the initial imputation methods.

Considering these conclusions, from this test onward, the MIWAE and MIDA methods will utilize Mean-Imputation while the GAIN method will utilize Zero-Imputation. This will be the case unless a specific experiment has conditions that favour a different method of initial imputation. On such cases it will be specified that a different method was chosen and the reasons why.

3.3.2 Influence of Dataset Size

3.3.2.A Description

In the second set of experiments we aim to understand how the dimensions of the dataset being used can influence the generative MVI methods we explored. The datasets we selected have diverse sizes as well as diverse compositions, meaning that, for the sake of this experiment, there could be factors that influence the result other than the size of the dataset. Furthermore, it would be ideal to have regular intervals in the sizes of the datasets being tested. Considering this, we decided to pick a single dataset and divide it into progressively bigger subsets, for each experiment.

The datasets with the highest amount of observations (House Sales in King County) and the highest amount of attributes (UFC-Fight Historical Data) were chosen for experiments varying the amount of observations and attributes respectively. The subsets were created according to an exponential scale. The first test had subsets created from the KC dataset with the amount of observations equal to $2^n, n \in [7, 14]$. The second test had subsets created from the UF dataset with the amount of attributes equal to $2^n, n \in [3, 7]$. Both complete datasets were also added to the end of their respective subset lists.

Knowing that different observations and features can carry different weights in the imputation process, we decided to perform the tests with 5×5-Fold Cross-Validation. With each iteration of the CV process different features were selected, so as to prevent the influence of particular sets of features in the final results. A fixed rate (0.2) of missing data was utilized for every observation.

Finally, for this experiment we decided to evaluate the three generative methods described previously (MIWAE, MIDA and GAIN), as well as the state-of-the-art discriminative methods (MICE and MF) and also OSM for reference.

3.3.2.B Results and Discussion

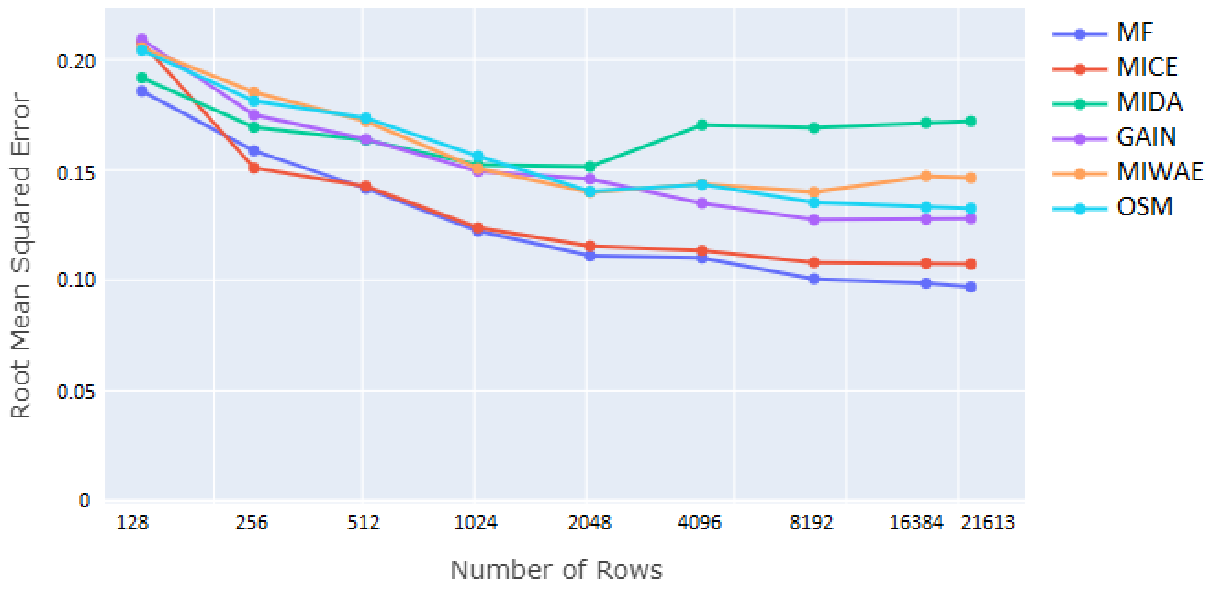


Figure 3.6: RMSE for imputation performed by all the MVI methods (MF, MICE, MIDA, GAIN, MIWAE, OSM) on the House Sales in King County Dataset (KC) with 20% missing data, for different numbers of observations.

In the first experiment, we test how the length of the dataset can affect the imputation process. In Figure 3.6 we can see the results for the imputation performed on the exponentially increasing subsets of the KC dataset. These results allow us to draw some conclusions. First, as was expected, an increase in the amount of observations results in a decrease in RMSE for all of the MVI methods. The increase in the number of observations, for the most part, affected all the MVI methods similarly, with the exception of the MIDA and MIWAE methods. These two methods struggled with the increase in the dataset's size, with their error values eventually stagnating, while the others kept decreasing. In this experiment the MICE and MF methods still performed the best out of all. Nonetheless, the high customizability of the generative methods still leave room for further improvement, as their setups were not altered to address the different sizes of the datasets.

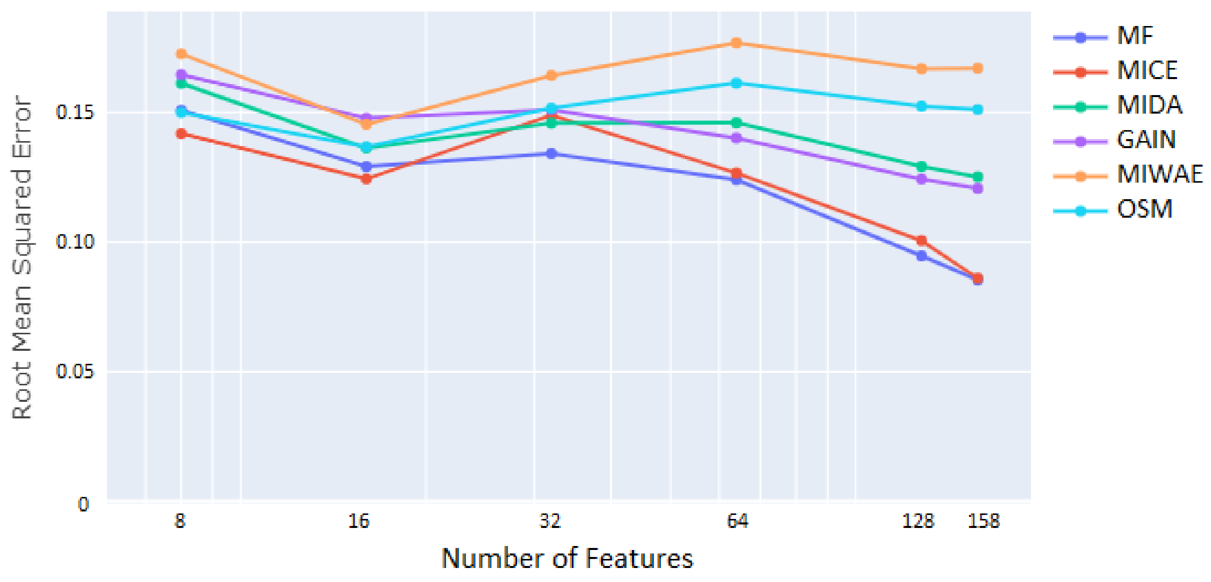


Figure 3.7: RMSE for imputation performed by all the MVI methods (MF, MICE, MIDA, GAIN, MIWAE, OSM) on the UFC-Fight Historical Data Dataset (UF) dataset with 20% missing data, for different numbers of attributes.

Figure 3.7 presents the results for the imputation performed on the subsets of the UF dataset with an exponentially increasing number of features. These results allow us to draw some conclusions. Notably, the OSM method used for reference, resulted in fairly consistent RMSE values, which was to be expected in this experiment. Again, as was expected, an increase in the amount of features resulted in a slight decrease in RMSE for all of the MVI methods, with the exception of the MIWAE method which returned somewhat constant values. The increase in the number of features affected all other MVI methods in a comparable manner. Similarly to the previous test, in this one the MICE and MF methods still performed the best out of all methods, with the MIDA and GAIN presenting promising results.

3.3.3 Comparison of MVI methods

3.3.3.A Description

In the final set of experiments we aim to compare the overall performance of all MVI methods explored so far, in order to determine which models perform the best.

To perform this experiment, all the imputation methods described in 3.2.1 were used, on the ten datasets. As was the case in 3.3.2, the OSM method is used mainly for reference, as it is expected to be the worst performer. For each imputation method-dataset pairing, eight measurements were performed, with the amount of missing data $\in [0.1, 0.45]$. For each of these measurements 5-Fold Cross-Validation was used. The data corruption introduced to each dataset was identical when testing the different imputation methods.

3.3.3.B Results and Discussion

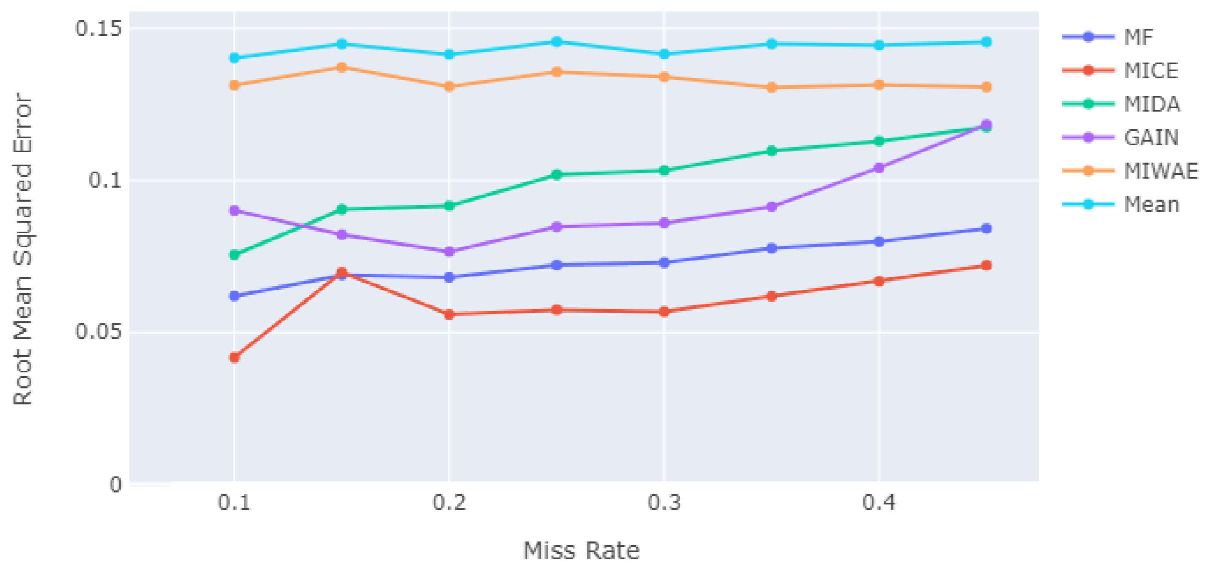


Figure 3.8: RMSE for imputation performed by all the MVI methods (MF, MICE, MIDA, GAIN, MIWAE, OSM) on the Breast Cancer Wisconsin Dataset (BC) for values of missing data $\in [0.1, 0.45]$.

Similarly to the structure in 3.3.1, in this experiment we compare the different methods when varying datasets and the rate of missing data. In Figure 3.8 we can see the RMSE for each of the MVI methods on the BC dataset (see Appendix B for the RMSE results obtained with the remaining datasets). Out of the 10 datasets utilized, the BC dataset is the most similar in size to the MCI-AD datasets further explored in this dissertation.

Grouping our measurements according to the dataset and missing-data rate, allows us to assert how these factors affect each of the MVI methods. From the obtained results we can extract the following information:

- Dataset Size - The MICE and MF methods were the clear best performers in the larger datasets, confirming what we observed in 3.3.2. The MIDA method had the worst drop in performance as the datasets increased in size. Nonetheless it was one of the best performers in datasets of smaller size, along with MICE and MF. The GAIN and MIWAE methods had rather consistent results, independently of the amount of observations in the dataset utilized. MIWAE had worse results in datasets with more features.

- Missing Rate - Compared to the remaining methods, MF, MIDA and MIWAE presented extremely consistent results as the rate of missing data increased. On the other hand, the GAIN and MICE methods sometimes struggled as the amount of missing data neared 50%. Overall, the MF, MICE and MIDA methods performed the best, regardless of missing rate. GAIN performed better than

MIWAE for smaller missing rates, and the reverse was true for larger missing rates.

Overall, MF performed better than the remaining methods in the majority of the experimental setups. Immediately following it, the MICE and MIDA methods were consistently among the best methods. The GAIN method had situational setups that allowed it to outperform some of the other methods. It presented particularly good results in larger datasets with smaller miss rates. Inversely, MIWAE's best performing test scenarios were those with smaller datasets and larger amounts of missing data.

3.4 Summary

In this chapter, we studied the problem of missing values in data and how to address it with state of the art methods. These methods are divided into generative and discriminative, where generative models learn the underlying structure of the data, while discriminative make few assumptions of the model's structure. Three methods from each of these two classifications were selected and utilized for testing. To test them, 10 datasets of real life data were selected, presenting diverse sizes and data. Utilizing these datasets we were able to determine how each of the imputation methods' results are affected by factors such as: Initial imputation (exclusively for generative methods); Amount of features in the dataset; Amount of observations in the dataset; Rate of missing data.

The results obtained allowed us to understand under which scenarios each of these methods can thrive. This allows us to determine the best methods to utilize in the main problem of this dissertation. The four CCC-3 datasets studied are composed of 400 to 600 observations, with more than 40 features and varying rates of missing data. Considering these constraints and the results obtained in this chapter, the MissForest method stands out as the best approach. This method consistently performed better than the remaining under scenarios similar to the one described. The MICE and MIDA methods emerge as close second and third best approaches. MICE was consistently one of the best methods, although not specifically for these constraints. Alternatively, these constraints coincide with those that allowed for the MIDA method's best results. After this, the GAIN method presented better results than MIWAE in tests with similar constraints.

In the tests performed in this chapter, the configurations of the generative methods were kept consistent, regardless of dataset. However, in 3.1.2 we learned that the structure of the neural network in generative methods can be of great importance. This allows for the possibility of the generative methods increasing in performance once their configuration is perfected for the datasets being handled.

4

Alzheimer’s Disease

Contents

4.1 Mild Cognitive Impairment and Alzheimer's Disease

Mild Cognitive Impairment (MCI) is responsible for slight yet noticeable declines in cognitive abilities. Such declines are significant enough to be noticed by the people affected and those surrounding them, while not having a serious impact on their daily life. The declines can come in the form of memory loss (Amnestic MCI), or in the form of non-memory related impairments, such as impairments to logical thinking, time perception or visual perception, among others (Nonamnestic MCI). The causes of MCI are not yet fully understood, although MCI has been found to be linked with risk factors common with dementia, like aging and a family history of dementia. Currently it is estimated that 15% to 20% of people aged 65 and older have MCI.

People diagnosed with MCI are not considered demented [63], and some can see their condition remain stable or even revert to normal. MCI can also be erroneously diagnosed, as it can be difficult to differentiate between cognitive impairments caused by MCI from those caused by aging or medication. Nevertheless, people diagnosed with MCI face a substantial risk of having their condition progress to dementia, more specifically to Alzheimer's Disease (AD), responsible for the majority of dementia cases worldwide. It is presently estimated that the cumulative progression from MCI to AD over 3 years is around 50% [64].

The World Alzheimer Report 2019 estimates that 50 million people live with dementia globally, of which two thirds are caused by AD [2]. This number is expected to continue rising in the coming years as a result of the ageing of global population, with the expectation that it will reach 152 million people by 2050. This can be evidenced by the statistic that shows that more than 30% of people aged 85 and older have AD. There is no known treatment or cure to AD, and the current medical goal is to mitigate its symptoms and delay their evolution.

4.1.1 Neuropsychological Tests

Currently, there are several ways of detecting early traces of Alzheimer's Disease in a patient. These include: brain scans, such as Computed Tomography (CT), MRI or PET; laboratory tests, such as blood, urine and cerebrospinal fluid analysis; and NPTs, like language, memory or problem solving tests, among others. Usually a combination of these is chosen in order to improve detection accuracy.

Traditionally, CT and MRI present results with higher accuracy, although they come with some challenges, as they can be expensive and invasive to a patient. They are also not widely available to all patients. NPTs on the other hand, are not invasive, inexpensive and can easily become accessible to the general population, while providing competitive results. As a result of these factors, NPTs can be used to detect neurodegenerative diseases in their early stages, when any declines are subtle and might be missed in routine health check-ups.

NPTs typically consist of paper-and-pencil tests that can assess either the memory cognitive domain, the non-memory cognitive domain or a combination of both. They consist of problems that test memory, motor function, perception, language, problem-solving and decision making, among others. NPTs are particularly useful in the study of neurodegenerative diseases, as these are associated with a continuous cognitive decline, but can also be useful when dealing with certain emotional disorders and brain injuries.

Multiple NPT batteries have been developed, with the Alzheimer's Disease Assessment Scale - Cognitive Subscale (ADAS-Cog) and the Mini Mental State Exam (MMSE) being the most well known [65] [66]. The ADAS-Cog battery was designed to assess the severity of the most important symptoms of AD. It is widely used to evaluate AD, although it is not the most sensitive to early changes in patients with MCI. MMSE is also widely used to evaluate the decline during both the MCI and dementia stages [67]. It can, however, be prone to low sensitivity results in early stage MCI and AD.

In Portugal, the *Bateria de Lisboa para a Avaliação de Demência* (or Lisbon Battery for Dementia Evaluation) (BLAD), was developed and validated specifically for the Portuguese population [68]. BLAD evaluates multiple cognitive domains, specifically: attention, motor initiative, verbal comprehension, verbal and non-verbal abstraction, visuoconstructional abilities and executive functions, calculation, immediate memory, working memory and learning and verbal memory [69].

4.2 Data

The Cognitive Complaints Cohort (CCC), a study conducted by researchers from the *Universidade de Medicina de Lisboa* (or Faculty of Medicine of the University of Lisbon) (FMUL), was launched to evaluate the progression of MCI and AD on Portuguese patients [67]. The study admitted patients initially diagnosed with MCI and had them take Lisbon Battery for Dementia Evaluation (BLAD) tests. These were performed and evaluated with the help of several institutions (Laboratory of Language Studies at Santa Maria Hospital and a Memory Clinic, both in Lisbon, and the Neurology Department at University Hospital in Coimbra). After their initial tests, the patients had follow-up appointments where they were tested for dementia.

The initial dataset (CCC-1) with the patients' test scores and diagnosis was released on April 2017. Since this release, and as of present day, the dataset has been updated twice, with CCC-2 being released in October2017 and CCC-3 being released in October 2018. The dataset is divided into two subsets: one for patients recruited in Lisbon and another for patients in Coimbra.

The original datasets were organized according to a First-Last approach, where one single dataset would hold the data for all patients and each data instance had information regarding their initial and last evaluation. In [5], T. Pereira proposed reorganizing the dataset into a Time Window approach. This approach groups data according to a specific temporal frame. For example, the dataset referring to the five year window would have the initial tests performed by each patient and a class indicating whether they converted to dementia after that five year interval.

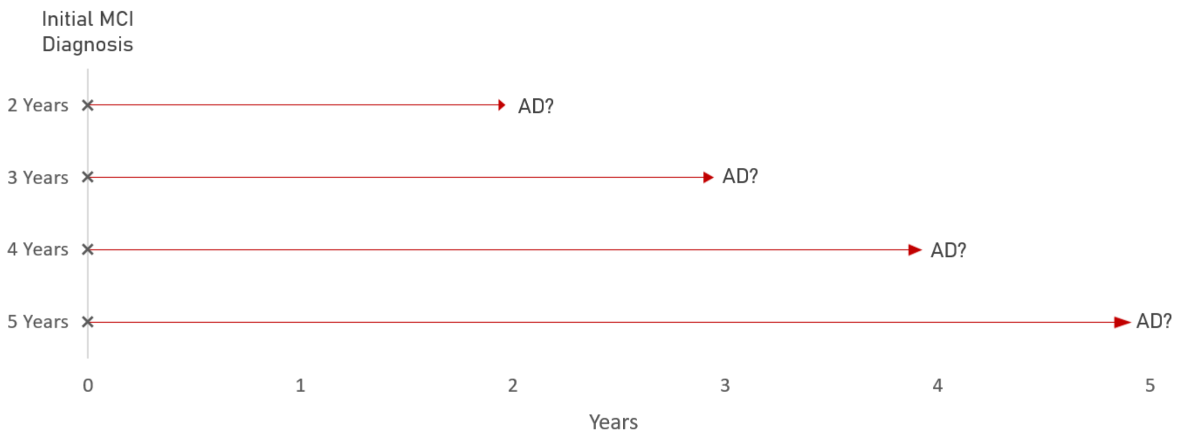


Figure 4.1: Representation of the four datasets

The Time Windows approach allows for more homogeneous datasets, which can provide more accurate models. Furthermore, it enables the creation of as many models as there are datasets, allowing not only to predict the progression of a given patient from MCI to dementia, but also a time window for that conversion.

In this thesis we will be utilizing CCC-3 following the Time Windows approach, as proposed by T. Pereira. The data is organized into four different datasets, with each referring to a different time window (2, 3, 4 and 5 years). In these datasets each instance represents a patient, and each feature represents a neuropsychological test from BLAD taken by the patient at the time of the initial MCI diagnosis. The final class in each dataset reveals whether the patient converted to dementia (cMCI) or had their condition stabilized (Stable MCI (sMCI)) in the given time window of the respective dataset. Each patient appears no more than once in each of the datasets.

The characterization of these datasets is presented in table 4.1.

Table 4.1: Characterization of the datasets utilized: Number of stable MCI cases (sMCI), number of converter MCI cases (cMCI), total number of samples and percentage of missing data.

Dataset		sMCI	cMCI	Total	Missing Data
CCC-3 Lisbon	2Y	394	107	501	28.38%
	3Y	305	163	468	28.27%
	4Y	227	204	431	28.34%
	5Y	175	235	410	28.96%
CCC-3 Coimbra	2Y	64	10	74	32.72%
	3Y	50	17	67	32.46%
	4Y	40	21	61	32.64%
	5Y	30	23	53	33.89%

4.3 Pipeline Setup

Before beginning our experimental work, we need to determine a pipeline setup. This setup should be versatile and allow for swapping methods at any point along the classification pipeline. Assuring this allows us to compare different techniques at each step, while maintaining an otherwise identical setup. By the end, this setup will allow us to obtain the most accurate results possible, using the technology studied so far. To determine the final pipeline setup we will need to perform several tests on the different state of the art technologies explored previously.

As it was explained in Chapter 2, there are several steps along the classification pipeline. In this section we will explain how the proposed pipeline is structured and what techniques were considered for each of those steps. fig. 4.2 illustrates the proposed classification pipeline.

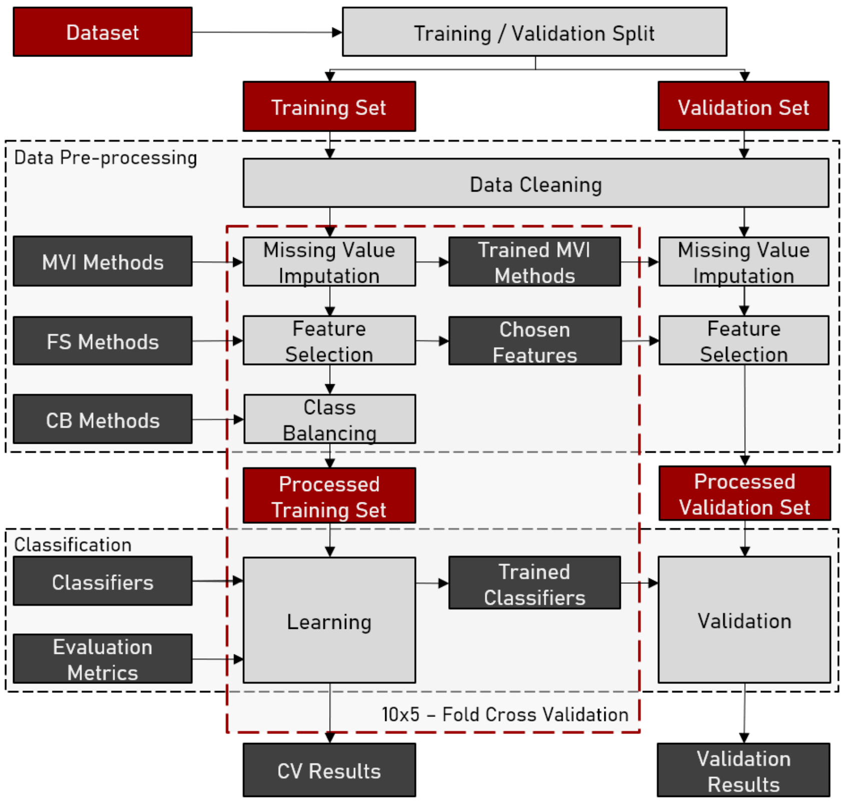


Figure 4.2: Pipeline for the entire classification process, including the evaluation of the obtained results

The first step in our pipeline is to split the dataset into a Training dataset and a Validation dataset, where the goal is to correctly classify the validation set. This division has to be performed early in the process because, even though both datasets will go through similar steps, some of the actions performed on the validation set will be performed based on knowledge obtained from the training set. After the split, both datasets will go through a data cleaning step, where similar changes are performed on both.

This setup utilizes 5-Fold Cross Validation with fold randomization which is repeated 10 times. The remainder of the data pre-processing steps are performed within each iteration of the cross validation process. After the train/test division in the CV process, the selected MVI and FS methods are performed consecutively on the training dataset. The trained MVI methods are then used to impute the testing and validation datasets based on the training data. The features determined by the FS methods are selected from the training and validation sets as well. The order in which these two steps are performed is not a consensual topic within the scientific community. In T. Pereira's PhD dissertation [1], FS was performed before MVI, but in that case the only MVI method explored was the OSM method. Considering that a substantial part of our work was performed on the MVI step, we opted to perform it first so as to allow the MVI methods to potentially highlight the importance of some features.

Afterwards, class balancing is performed exclusively on the training dataset ahead of the learning process, completing the data pre-processing steps. The selected classifier is used to learn the training data and then classify both the test and validation sets. To assess the results, one or more evaluation metrics are selected and used to compare the predicted values to the expected ones. The final results presented are averages of all the results obtained during the CV process.

The methods selected to test each of these classification steps are described in the following sections.

4.3.1 Data cleaning

Most of the work required for this step has been performed beforehand, with the help of the teams responsible for gathering the data. This includes reporting errors and detecting inconsistencies or outlier values.

Additionally, in this pipeline, features with more than 70% missing data are eliminated from the datasets. This is done so as to optimize the following imputation performed by the MVI methods.

4.3.2 Feature Selection

In Section 2.1 several feature selection methods were explored. Out of those described, the MIM, Chi-Squared (CS) and Recursive Feature Elimination using SVM (SVM-RFE) methods were chosen for testing. The internal parameters for each of these methods were obtained through grid-search performed on different classifier/dataset combinations. The results for this grid search are presented in Section 5.4. In addition, the SVM-RFE method utilizes L1 regularization.

Each of these methods will perform its selection solely based on the training dataset. After the selection, both the training and testing datasets will be modified according to the chosen features.

Besides the methods described, a specific set of features will be utilized for measurements where other variables are being compared. This set of features will be referred as "T. Pereira's FS", and it consists of the best features chosen for classification in Telma Pereira's PhD dissertation [1]. According to the dissertation, different features were selected for each of the year windows, but since the majority of the features were common to all datasets and knowing that several other FS methods will be tested, we decided to utilize a single set of features for all the windows. "T. Pereira's FS" in this work includes all features that were utilized in at least one of the year windows in the Pereira's dissertation.

Regrettably, the datasets seem to have suffered some reorganization, which means that not all the features utilized in T. Pereira's dissertation are present in the latest version of the datasets. Specifically, the features present in Table 4.2 could not be identified in the latest version of the datasets. Table 4.3 presents the list of features in this selection.

Table 4.2: Features utilized in [1] that were not included in T. Pereira's FS

Feature Names
CVLT A list (3thtrial)
CVLT A list (4thtrial)
Fi_LM_a
Digit Span – Forward (Z-score)
Digit Span – Total (Z-score)
Information (Z-score)
Orientation - MSQ (Z-score)
Cancelation task -Toulouse- Pierón (concentration index) (Z-score)
CVLT A list (five learning trails total) (Z-score)

4.3.3 Classifiers

In order to explore as many testing configurations as possible, we decided to utilize at least one classifier from each of the ones explored in Section 2.4. The classifier setups presented in Table 4.4 were determined through grid search on the four datasets. The datasets were preprocessed utilizing the OSM method for missing Value Imputation and the SMOTE method for class balancing as well as T. Pereira's features, described in Section 4.3.2.

The final Neural Network configuration consisted of a three layer setup, where the first layer has an amount of neurons equal to the input size, the middle layer has an amount of neurons equal to half of the input size and the final layer has one single neuron (as the output is binary). The middle layer utilizes dropout regularization of 0.25 and uses ReLU as the activation function, while the final layer utilizes the Sigmoid activation function.

Several other Neural Network configurations were tested having up to six layers of neurons, with each layer containing between 1 neuron and $1.5 \times inputsize$ neurons. Multiple values for dropout regularization and multiple activation functions were also tested, including ReLU, Sigmoid and Softmax.

Table 4.3: Features included in T. Pereira's FS, according to [1]

Feature Names
Age
Age of first symptons
Cancelation Task - A's time
Cancelation Task – A's total
Digit Span - Forward
Digit Span - Backward
Verbal Paired-Associate Learning – Easy
Verbal Paired-Associate Learning –Difficult
Verbal Paired-Associate Learning – Total
Logical Memory Immediate A free recall
Logical Memory - A Immediate Cued
Logical Memory with Interference-A
Word Recall – Free recall
Word Recall –Total
Orientation (Total)
Orientation – Personal
Orientation – Spatial
Orientation – Temporal
Verbal Fluency
Token Orders (total)
Cube Draw
Calculation
Raven Progressive Matrices
Interpretation of Proverbs - (Verbal Abstraction)
Trail Making Test (Part B) - time
CVLT A list (1sttrial)
CVLT A list (Total intrusions in 5 recalls)
CVLT A list (five learning trails total)
Orientation- MSQ
Blessed Dementia Scale (Total of Part 1 - Daily living activities)
Fi_LM_a_m100
Cancelation Task – A's total (Z-score)
Digit Span – Backward (Z-score)
Verbal Paired-Associate Learning (Z-score)
Word Recall –Total (Z-score)
Orientation (Total) (Z-score)
Verbal Fluency (Z-score)
Raven Progressive Matrices (Z-score)
Interpretation of Proverbs – (Verbal Abstraction) (Z-score)
CVLT A list (5sttrial) (Z-score)
Logical Memory Immediate A free recall (Z-score)
Logical Memory with Interference-A (Z-score)

4.3.4 Missing Value Imputation Methods

In Section 3.4 we concluded that the MICE, MF, MIDA and GAIN methods were the best candidates to address the problem discussed in this chapter. For that reason we decided to utilize those four methods in our tests. We also utilized the OSM method as a baseline, since it was also the method utilized in [1].

Table 4.4: Parameters and configurations for the classifiers, determined through grid search performed on the four datasets. Considering $n \in \mathbb{N}$.

Classifier	Parameter	Chosen Value	Tested Interval
Naive Bayes	probability density	Gaussian	-
	smoothing	10^{-6}	$10^n, n \in [-9, -1]$
Decision Tree	attribute selection	Gini Index	{Gini Index, Entropy}
Support-Vector Machine	kernel	RBF	-
	gamma	scale	{'scale', 'auto'}
Support-Vector Machine	kernel	Polynomial	-
	degree	1	{1, 2, 3}
	gamma	auto	{'scale', 'auto'}
K-Nearest-Neighbours	number of neighbours	11	$n \in [1, 15]$
Random Forest	attribute selection	Gini Index	{Gini Index, Entropy}
Logistic Regressor	ridge	10^{-6}	$10^n, n \in [-1, -9]$
Neural Network	batch size	150	$10n, n \in [2, 30]$
	epochs	50	$10n, n \in [2, 20]$
	configuration	*	

Grid search was preformed on the MVI methods to determine their setup, and the results are presented in Table 4.5.

Table 4.5: Parameters and configurations for the MVI methods, determined through grid search performed on the four datasets. Considering $n \in \mathbb{N}$.

MVI Method	Parameter	Chosen Value	Tested Interval
MICE	initial strategy	Mean	-
	max iterations	10	-
MissForest	number of trees	100	-
	max iterations	10	-
MIDA	batch size	1/2 dataset size	$n^{-1}, n \in [1, 5]$
	epochs	250	$10n, n \in [2, 40]$
	theta	16	$n \in [2, 40]$
GAIN	batch size	64	$2^n, n \in [5, 9]$
	epochs	1000	$100n, n \in [1, 25]$
	alpha	400	$10n, n \in [1, 100]$
	hint rate	0.92	$0.01n, n \in [10, 98]$

4.3.5 Class Balancing Methods

In Section 2.3 several methods for overcoming class imbalance were described. Out of these methods we decided to test five that were diverse and that better fit the problem at hand. Those five include two undersampling methods (Random Undersampling and TOMEK Links) as well as three oversampling methods (SMOTE, ADASYN and MAHAKIL). Grid search was performed for some of the methods in order to determine their parameters. The selected parameters are as follows:

Table 4.6: Parameters and configurations for the MVI methods, determined through grid search performed on the four datasets

Class Balancing Method	**Parameter**	**Chosen Value**	**Tested Interval**
Random Undersampling	-	-	-
TOMEK Links	-	-	-
SMOTE	number of neighbours	7	[1, 11]
ADASYN	number of neighbours	5	[1, 11]
MAHAKIL	-	-	-

4.3.6 Metrics

To test the classifiers we performed 5-Fold Cross Validation (as described in Section 2.5.4) and repeated the process 10 times for each setup, resulting in a 10$\times$5-Fold Cross Validation procedure with fold randomization. The four metrics discussed in Section 2.5 (Accuracy, ROC AUC, Sensitivity and Specificity) were used to evaluate the results.

4.4 Summary

In this chapter we explained Alzheimer's Disease, how it differs from Mild Cognitive Impairment and the contexts in which it can emerge. Then we described the data that will be used in this thesis, and how it is obtained through Neuropsychological tests. These tests come with certain advantages, such as their availability and monetary cost. In spite of this, there are big disadvantages, like human error at the collection point, discrepancies between collection sites, missing data, among others. Disadvantages that we can only try to overcome resorting to machine learning models. We characterized the datasets that will be used and how they are organized, providing some relevant information for future tasks.

The chapter ends with a description and explanation of the proposed pipeline setup for classifying Neuropsychological test results. This section explained the pipeline's setup in detail as well as the selection of methods that we chose to test. It continues by detailing the configurations of each of these methods and the metrics that will be used to evaluate them.

5

Predicting Conversion from MCI to AD

Contents

In this section, we aim to evaluate the performance of the methods explored so far. The goal is to determine the combination of methods that allows us to achieve the best classification results possible for each dataset. Considering the amount of variables at play, our tests were performed according to a funneling strategy, where some of those variables are eliminated with each test that is performed. The experiments were structured as described next.

5.1 Previous Result Recreation

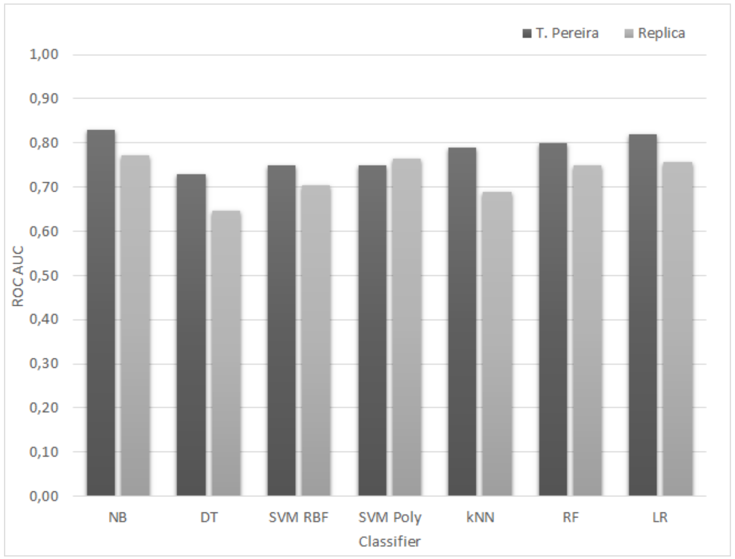
As was mentioned before, this thesis aims to build upon the work performed on Telma Pereira's PhD dissertation [1]. Considering this, results obtained in said dissertation can be utilized as a baseline, allowing us to compare future results.

The results presented in the dissertation were restricted to some specific testing scenarios, which would restrict us to perform tests with similar setups (such as the train/test division and scoring metrics utilized). For this reason, we decided to first replicate the described pipeline setup as closely as possible in order to have our own frame of reference.

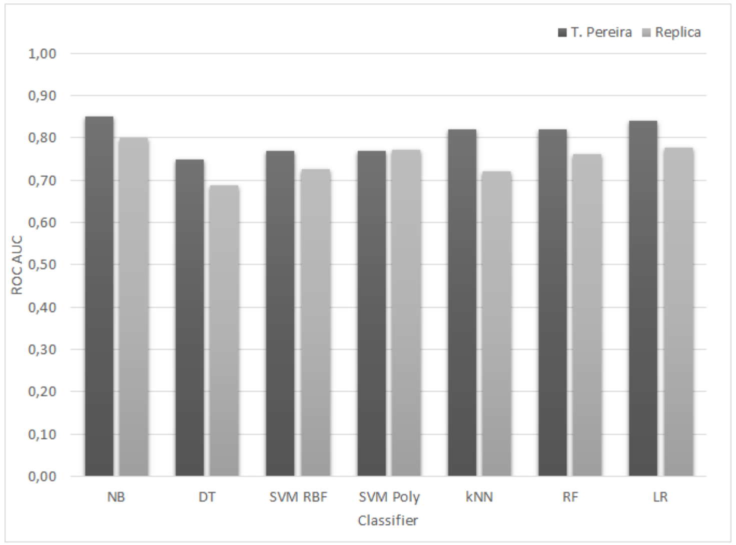
The pipeline was set up using the SMOTE method for handling class imbalance, the OSM method for missing value imputation and seven different classifiers: NB, DT, SVM RBF, SVM Poly, KNN, RF and LR. The CCC-3 Lisbon datasets were utilized for training and the CCC-3 Coimbra datasets for validation. The features presented in Section 4.3.2 were utilized. 10$\times$5-Fold Cross Validation was performed and AUC, Sensitivity and Specificity were measured.

Nonetheless, some differences were inevitable between the two setups: As mentioned in Section 4.3.2, some features were missing from the dataset. There was no information available in the dissertation regarding the values of certain classifier-specific parameters, therefore grid search was performed to determine the ideal values for each. Finally, the OSM method utilized in this thesis distinguishes between continuous data and ordered categorical data, which was not the case for the OSM method utilized in the dissertation.

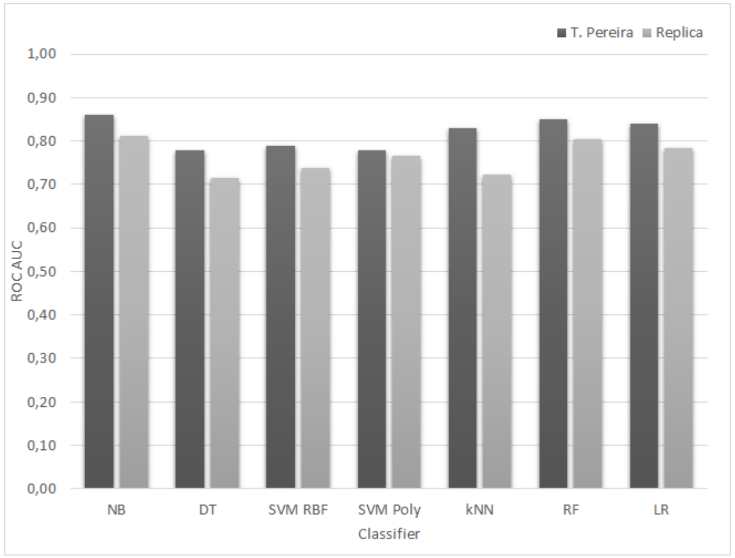
To assess whether our setups were similar we attempted to replicate the results present in chapter $4.2.3$ of the Dissertation. Figure 5.1 presents the comparison between the AUC scores present in the referred chapter and the ones obtained by us for this thesis. These scores represent the testing score obtained during the Cross Validation process performed solely on the CCC-3 Lisbon dataset. From the graphs we can observe that our results are somewhat consistent with the ones in Pereira's dissertation, albeit with our results consistently lagging behind. The most notable differences come when using the DT and KNN classifiers, which consistently presented worse results for all datasets. On the other hand, the SVM Poly classifier returned identical results to the ones expected. Overall, the NB, RF and LR classifiers, which were consistently the best performers in Pereira's work, present some of the most similar results. This similarity allows us to have some confidence in emulated pipeline setup.



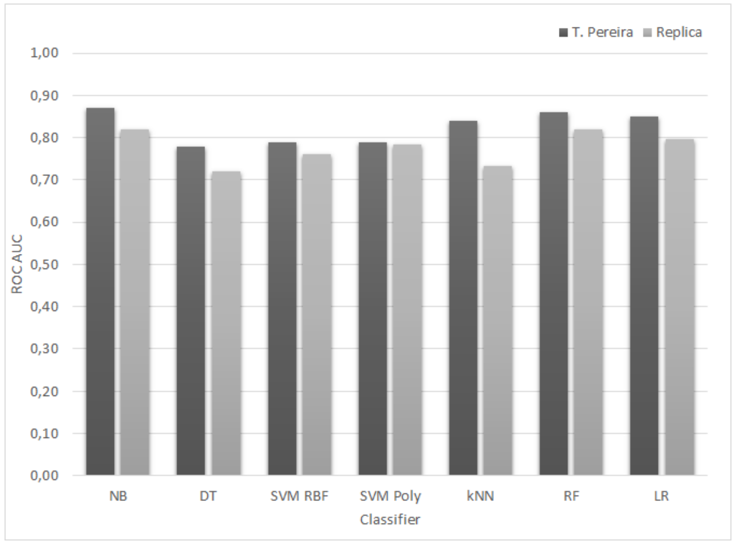
(a) 2 Year Dataset



(b) 3 Year Dataset



(c) 4 Year Dataset



(d) 5 Year Dataset

Figure 5.1: Comparison of the ROC AUC score between results presented on Telma Pereira's PhD dissertation (dark grey) and results obtained for this thesis (light grey), while using comparable pipeline setups.

Following these results, Pereira's dissertation presents the classification results obtained using the validation set (CCC-3 Coimbra). The presented values refer exclusively to the best performing classifier (Naive Bayes) and include AUC, Sensitivity and Specificity scores. Seeing as the Naive Bayes classifier was also the overall best performer in our results, we attempted to reproduce these values as well. Table 5.1 contains the comparison between those results and the ones obtained with our pipeline. The obtained AUC scores, again, are fairly similar, never differing by more than 0.04, although the values differ more significantly in the Sensitivity and Specificity scores. However this difference can be explained by the fact that the AUC score is obtained from both the Sensitivity and Specificity scores, and an increase in either leads to an increase in the AUC score. Classifiers are trained exclusively to improve the AUC scores, regardless of whether this is achieved by improving Sensitivity or Specificity. Altogether, the relative consistency between values allows us to accept the results as a baseline for future comparison.

Table 5.1: Comparison between the results presented on Telma Pereira's PhD dissertation (T. Pereira) and results obtained for this thesis (Replica). The results were obtained using the Naive Bayes model and the independent validation set (CCC-3 Coimbra), for the four datasets. The model was fine-tuned to the CV set (CCC-3 Lisbon).

	AUC				Sensitivity				Specificity			
	2Y	3Y	4Y	5Y	2Y	3Y	4Y	5Y	2Y	3Y	4Y	5Y
T. Pereira	0.66	0.67	0.64	0.63	0.77	0.80	0.68	0.60	0.50	0.41	0.57	0.65
Replica	0.68	0.66	0.61	0.59	0.73	0.72	0.65	0.57	0.62	0.61	0.57	0.60

5.1.1 Setup modifications

Having replicated Pereira's testing setup, we sought ways to improve upon it. One aspect of the original setup that stood out was the usage of the Coimbra CCC-3 dataset as a Validation dataset, and particularly the large discrepancies in scores from the Cross-validation results to the validation dataset results. After some initial experiments and analysis of the datasets we were able to determine that several features that were prevalent in the Lisbon dataset were completely blank in the Coimbra dataset. Further experiments with multiple Feature Selection (FS) algorithms highlighted that some of those features were of high importance to determine the classification result. While these discoveries explaining the results, they also showed why utilizing the Coimbra dataset as a validation set was not ideal.

We believe that utilizing a validation set is of great importance to the classification pipeline. Utilizing Cross Validation (CV) is beneficial in some issues, such as reducing over-fitting and increasing the robustness of the model, but it is not enough for the purposes required for this thesis. Having a fixed validation set allows for an unbiased comparison between the results of different models, or different pipeline setups.

For these reasons we opted to perform a different dataset split from the one utilized in Pereira's dissertation. In all the following tests (unless otherwise stated) we will utilize the Lisbon CCC-3 dataset exclusively. The dataset will be split according to an 80/20 split into training and validation sets. All the presented results correspond to those obtained with the validation set. The validation sets will be created with weighted sampling, meaning that the final validation set will have an identical class distribution to the one in the training set. The validation sets will always be identical within the same experiment, so as to properly compare the different models, although they might differ from one experiment to another. The difference in approaches is illustrated in fig. 5.2.

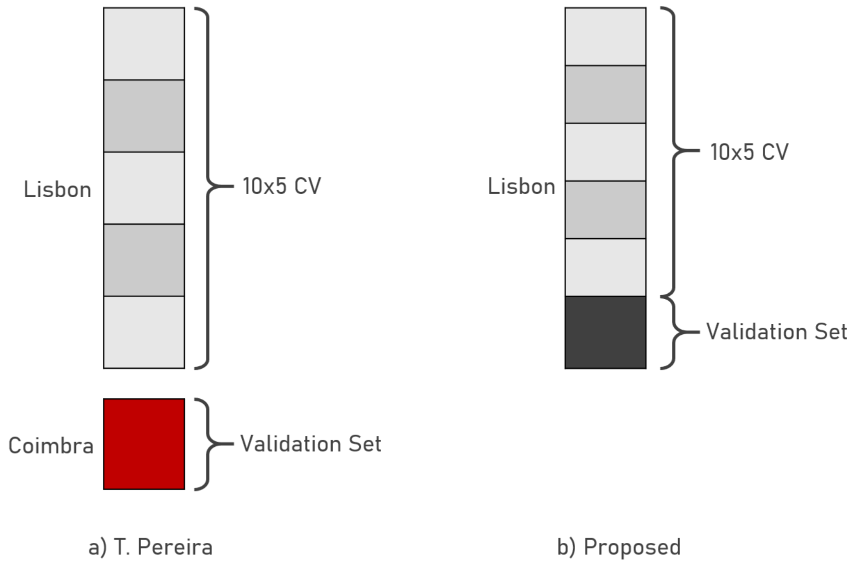


Figure 5.2: Division of the CCC-3 dataset into training and validation sets according to T. Pereira's dissertation (a) and to the proposal in this thesis (b).

5.2 Determining the best Classifiers

In the first set of experiments our aim was to reduce the list of potential classifiers available. The tests were performed on the eight classifiers described in Section 4.3, after grid search was performed to tune their respective parameters. All tests were performed exclusively on the CCC-3 Lisbon datasets, utilizing 10×5-Fold Cross Validation, with an 80/20 train/validation split.

To perform these tests we were first required to setup the remaining methods in our classification pipeline. The features in T. Pereira's FS were chosen from the original datasets and the SMOTE method was picked to address class imbalance. These methods were chosen according to the replica in section 5.1. For missing value imputation we opted to use the MissForest method, as we concluded in Chapter 3 that it was the one that provided us with the best results.

The classification scores obtained on the testing dataset are available in Table 5.2. To assess these

Table 5.2: Classification results of 10×5-Fold Cross Validation on all four datasets.

Dataset	Classifier	Accuracy	AUC	Sensitivity	Specificity
2Y	NB	0,78 ± 0,05	0,76 ± 0,06	0,79 ± 0,07	0,74 ± 0,13
	DT	0,72 ± 0,05	0,63 ± 0,05	0,79 ± 0,06	0,47 ± 0,11
	SVM RBF	0,69 ± 0,06	0,72 ± 0,05	0,67 ± 0,08	0,76 ± 0,08
	SVM Poly	0,77 ± 0,03	0,71 ± 0,05	0,81 ± 0,04	0,61 ± 0,11
	kNN	0,67 ± 0,04	0,64 ± 0,04	0,69 ± 0,06	0,59 ± 0,09
	RF	0,82 ± 0,03	0,70 ± 0,05	0,91 ± 0,03	0,49 ± 0,08
	LR	0,78 ± 0,04	0,74 ± 0,05	0,81 ± 0,06	0,67 ± 0,09
	NN	0,76 ± 0,04	0,73 ± 0,05	0,78 ± 0,06	0,69 ± 0,10
3Y	NB	0,77 ± 0,04	0,77 ± 0,04	0,78 ± 0,05	0,75 ± 0,08
	DT	0,67 ± 0,05	0,64 ± 0,06	0,74 ± 0,07	0,54 ± 0,12
	SVM RBF	0,70 ± 0,05	0,74 ± 0,04	0,67 ± 0,09	0,75 ± 0,06
	SVM Poly	0,75 ± 0,04	0,74 ± 0,04	0,80 ± 0,06	0,67 ± 0,08
	kNN	0,69 ± 0,04	0,68 ± 0,04	0,71 ± 0,07	0,66 ± 0,10
	RF	0,77 ± 0,03	0,75 ± 0,05	0,84 ± 0,05	0,66 ± 0,12
	LR	0,74 ± 0,05	0,73 ± 0,06	0,77 ± 0,05	0,69 ± 0,10
	NN	0,75 ± 0,04	0,74 ± 0,04	0,78 ± 0,07	0,70 ± 0,09
4Y	NB	0,81 ± 0,03	0,81 ± 0,03	0,84 ± 0,06	0,78 ± 0,05
	DT	0,65 ± 0,04	0,65 ± 0,05	0,69 ± 0,08	0,61 ± 0,09
	SVM RBF	0,73 ± 0,06	0,73 ± 0,06	0,69 ± 0,11	0,76 ± 0,07
	SVM Poly	0,76 ± 0,04	0,75 ± 0,04	0,79 ± 0,05	0,72 ± 0,07
	kNN	0,69 ± 0,05	0,68 ± 0,05	0,76 ± 0,06	0,61 ± 0,10
	RF	0,77 ± 0,03	0,77 ± 0,03	0,81 ± 0,04	0,73 ± 0,06
	LR	0,75 ± 0,05	0,75 ± 0,05	0,76 ± 0,07	0,74 ± 0,09
	NN	0,75 ± 0,04	0,75 ± 0,04	0,82 ± 0,06	0,68 ± 0,09
5Y	NB	0,79 ± 0,04	0,80 ± 0,04	0,84 ± 0,06	0,75 ± 0,06
	DT	0,69 ± 0,05	0,69 ± 0,05	0,67 ± 0,09	0,71 ± 0,08
	SVM RBF	0,75 ± 0,04	0,74 ± 0,04	0,70 ± 0,06	0,78 ± 0,06
	SVM Poly	0,76 ± 0,04	0,76 ± 0,04	0,78 ± 0,06	0,74 ± 0,05
	kNN	0,71 ± 0,05	0,71 ± 0,05	0,75 ± 0,09	0,68 ± 0,07
	RF	0,77 ± 0,04	0,76 ± 0,05	0,73 ± 0,10	0,79 ± 0,06
	LR	0,74 ± 0,04	0,73 ± 0,05	0,72 ± 0,12	0,75 ± 0,07
	NN	0,76 ± 0,06	0,77 ± 0,05	0,78 ± 0,11	0,75 ± 0,13

results we turned to the Friedman Test [70], which helped conclude that the Accuracy and AUC results achieved were significant ($p < 0.01$).

From the observed results some patterns can be noted, such as the Specificity results being significantly worse in the two year dataset (the most unbalanced dataset) and getting progressively better for the following datasets. It can also be observed that the Naive Bayes classifier consistently returned better Accuracy and AUC results than the remaining classifiers. Nonetheless, this test was not aimed at selecting the single best classifier, as other variables that were not yet tested can have direct impact on the classifier's performance. Therefore, four classifiers were selected as the best performers: NB, SVM Poly, RF and NN. This selection was made due to the fact that these four classifiers provided the best accuracy and AUC results on average for the four datasets. Considering this selection, the following experiments will be performed on these four classifiers exclusively.

In **Figure 5.3** we can see the accuracy results for the four best performing classifiers on the four different datasets.

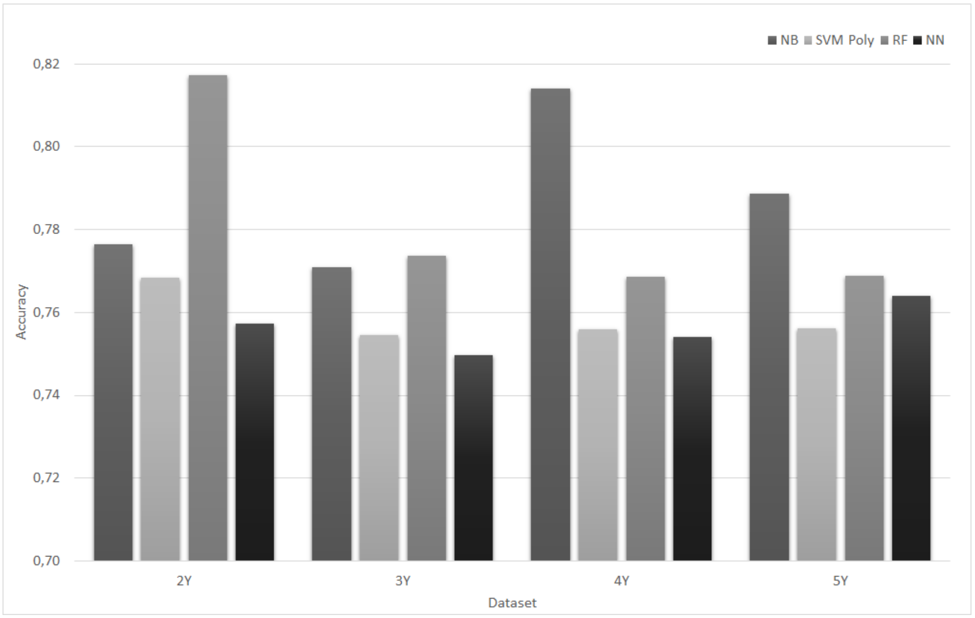
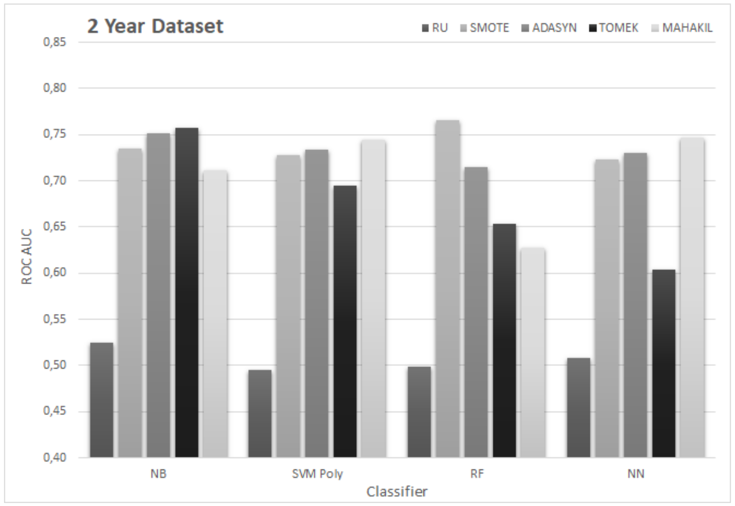


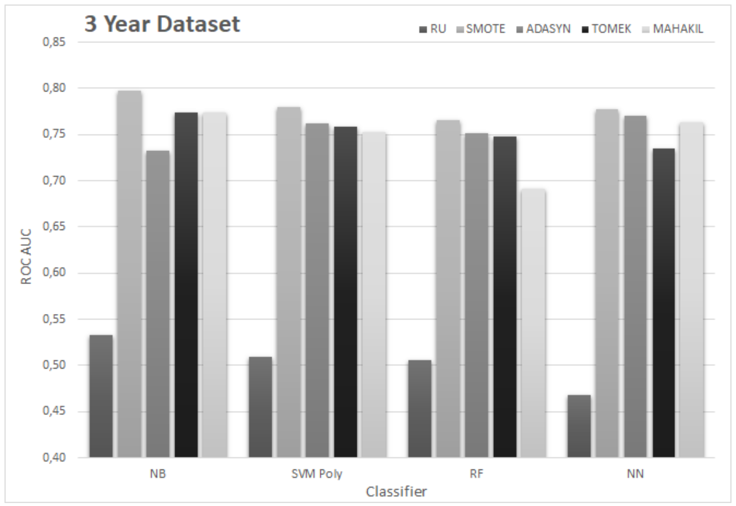
Figure 5.3: Accuracy results for the four best performing classifiers on all the datasets. The tests were performed utilizing the SMOTE method for overcoming class imbalance and the MissForest method for Missing Value Imputation.

5.3 Determining the best method for Class Balancing

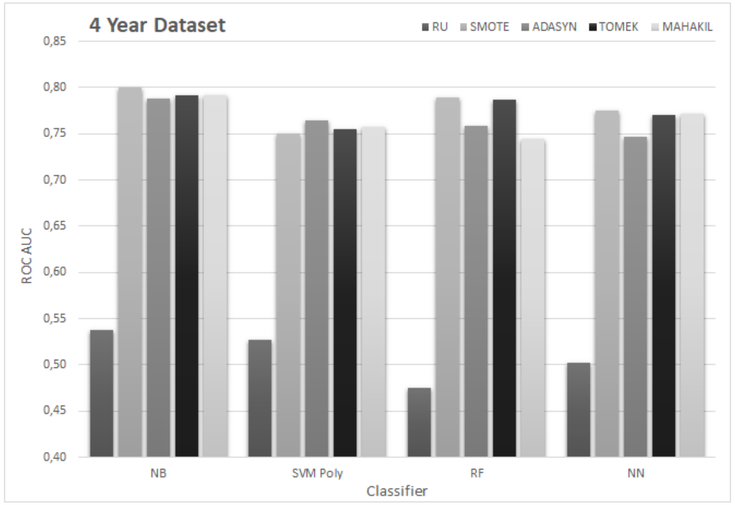
Having reduced the number of classifiers, the next step is to do the same with the proposed methods for class balancing. To do so, the same classification setup was utilized for the five class balancing methods being tested: The features in T. Pereira's FS were chosen and the MissForest method was utilized to impute the missing data. As discussed in the previous section, the four best performing classifiers were utilized for this test.



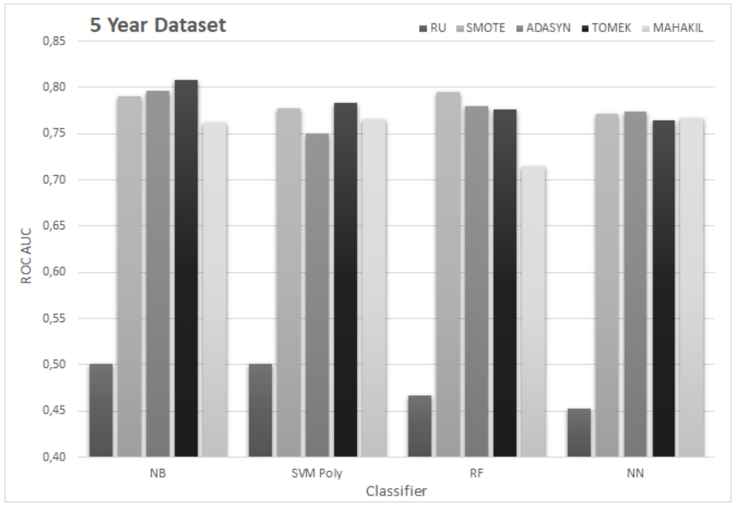
(a) 2 Year Dataset



(b) 3 Year Dataset



(c) 4 Year Dataset



(d) 5 Year Dataset

Figure 5.4: Comparison of the ROC AUC score obtained while using five different methods for class balancing (respectively Random Undersampling, SMOTE, ADASYN, TOMEK Links and MAHAKIL). The results were obtained using the four best performing classifiers mentioned in Section 5.2 (Naive Bayes, SVM with Polynomial kernel, Random Forest and Neural Network).

The ROC AUC scores, grouped according to the classifiers that were utilized, are available in Figure 5.4. According to the Friedman Test [70], the AUC results achieved were significant ($p < 0.01$). The obtained results allow for some immediate observations:

- The Random Undersampling method is clearly inefficient and significantly worse than others. We can attribute some of these poor results to the small size of the datasets utilized. Utilizing the chosen train/test split meant that the tests with the two year dataset, for example, ended up resulting in a training set with 172 samples and a testing set with 100.
- Even though the TOMEK Links method was overall one of the best, the Neural Network classifier performs worse on average when being paired with this method.
- The overall best AUC scores for each dataset were obtained with the SMOTE method (for datasets 2Y, 3Y and 4Y) and with the TOMEK Links method (for the 5Y dataset).

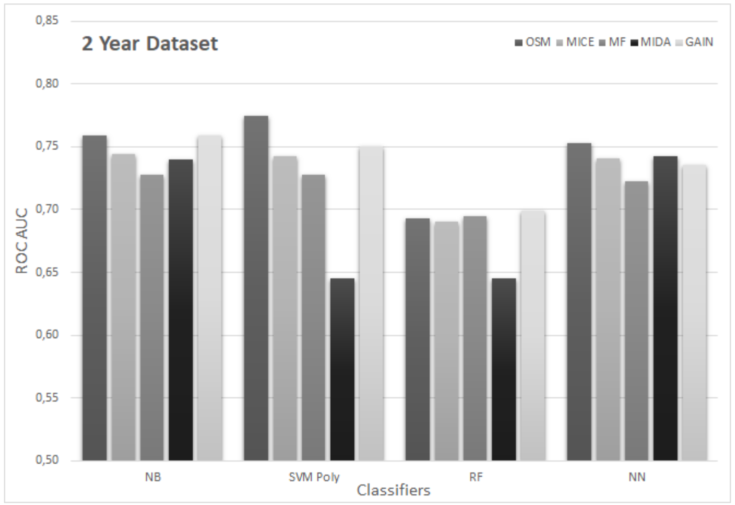
In conclusion, excluding the Random Undersampling method, all methods result in fairly comparable scores and each method outperformed the remaining in at least one testing scenario (considering all dataset and classifier pairings as a testing scenario). This means that there was no single method better than all others. Despite this, the SMOTE method resulted in the best results on average, as well as being responsible for some of the best results overall. After the SMOTE method, the ADASYN and TOMEK Links had fairly similar performances, with the MAHAKIL method falling slightly short of those.

Moving forward we decided to reduce the list of Class Balancing methods to SMOTE and TOMEK Links, for the following reasons: On average, both of these were among the three best performing methods. As was noted previously, all the overall best results for each dataset were obtained with one of these two methods. Furthermore, every time the ADASYN method outperformed SMOTE the difference was rather small, while the same cannot be said for the reverse situation. Finally, since the TOMEK Links method performs undersampling (while both SMOTE and ADASYN perform oversampling), choosing it allows for more diversity in future testing scenarios.

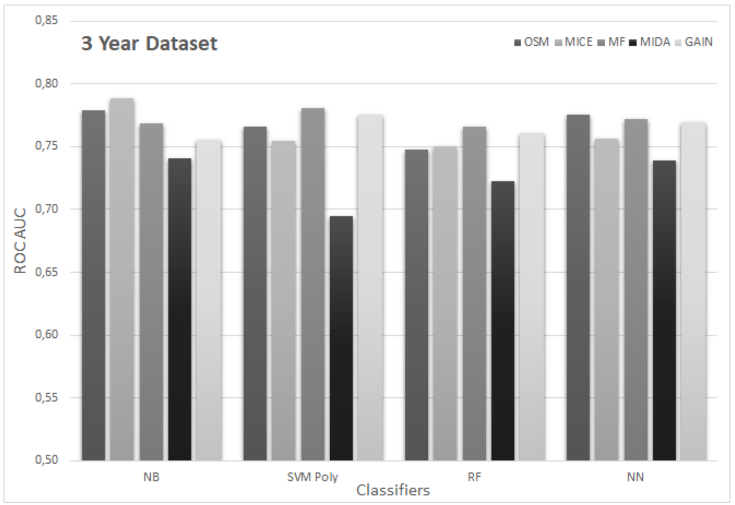
5.4 Determining the best Missing Value Imputation method

In this set of experiments, our aim is to apply some of the research performed in Chapter 3. Having reduced the number of classifiers and class balancing methods, the following tests aim to select the best performing methods for MVI.

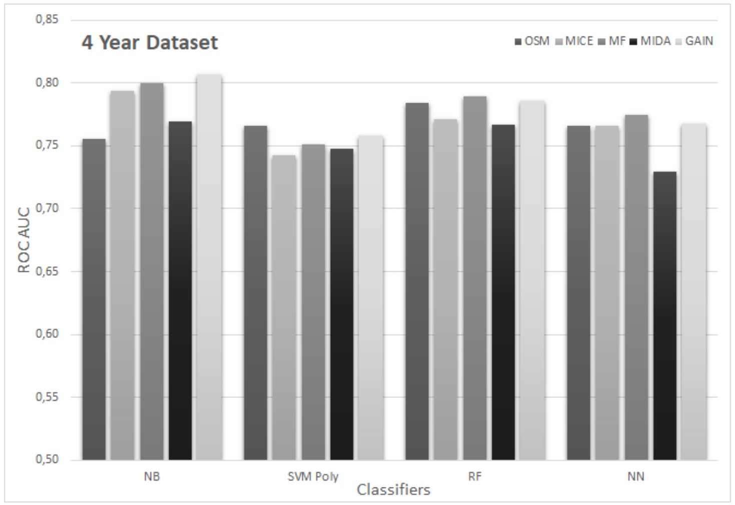
A similar pipeline setup to the ones utilized previously was selected: The features from T. Pereira's FS were chosen, the four best performing classifiers were utilized (NB, SVM Poly, RF and NN) as well as the two best performing class balancing methods (SMOTE and TOMEK).



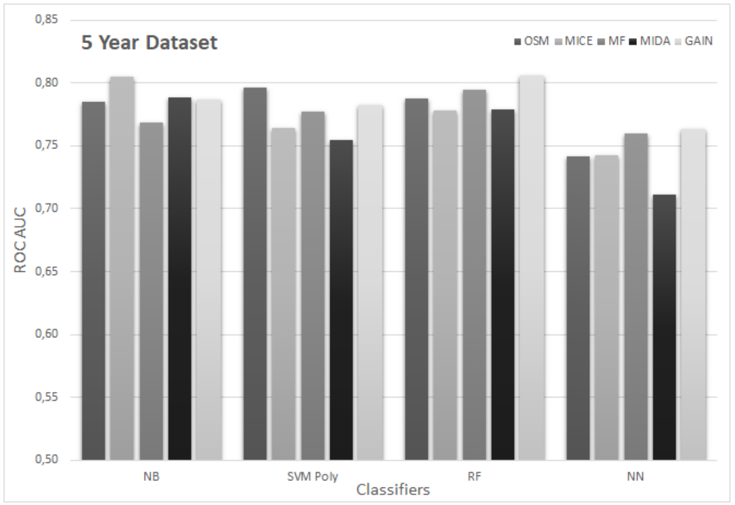
(a) 2 Year Dataset



(b) 3 Year Dataset



(c) 4 Year Dataset



(d) 5 Year Dataset

Figure 5.5: Comparison of the ROC AUC score obtained while using five different methods for MVI (respectively Overall Sample Mean, MICE, MissForest, MIDA and GAIN). The results were obtained using the four best performing classifiers mentioned in Section 5.2 (NB, SVM Poly, RF and NN) and the SMOTE method for class balancing.

The ROC AUC scores, grouped according to the classifiers that were utilized, are available in Figure C.1. These results only show the measurements obtained when using the SMOTE method, despite Tomek Links (TOMEK) having also been utilized (refer to Appendix C for the results obtained utilizing TOMEK). According to the Friedman Test [70], the AUC results achieved were significant ($p < 0.01$). The obtained results allow for some immediate observations:

- The MIDA method was the clear worst performer, seemingly struggling more with the more unbalanced datasets (2Y and 3Y).
- The MICE method performed significantly better when paired with the NB classifier than with any other classifier.
- The SVM Poly classifier returned its best results when paired with the OSM method.
- The RF classifier had the most consistent results, with relatively small variations for the different MVI methods.
- The overall best AUC scores for each dataset were obtained with the OSM method for the 2Y dataset, with the MF method for the 3Y dataset and with the GAIN method for the 4Y and 5Y datasets.

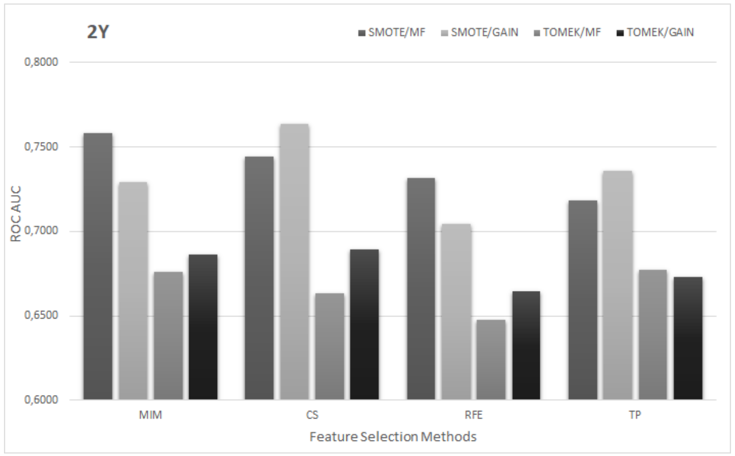
Further analysing the results, we can note that the class balancing method seemed to have minimal impact on the performance of the different MVI methods. Overall setups with SMOTE method returned better results than ones with TOMEK, and the best ROC AUC result for each dataset was always obtained utilizing SMOTE. If anything, the TOMEK method highlighted the difference between the best and worst performing MVI methods, as that difference was greater when looking exclusively at the results obtained with TOMEK.

Overall, the two best performing MVI methods were MF and GAIN. Not only did these contribute to some of the best results for each dataset, but they were also consistently among the best methods regardless of the classifier being used. There were a total of 32 different setup combinations (4 datasets, 4 classifiers and 2 class balancing methods) to compare the different methods. Out of those, the MF and GAIN methods were responsible for the best ROC AUC scores in 18 cases.

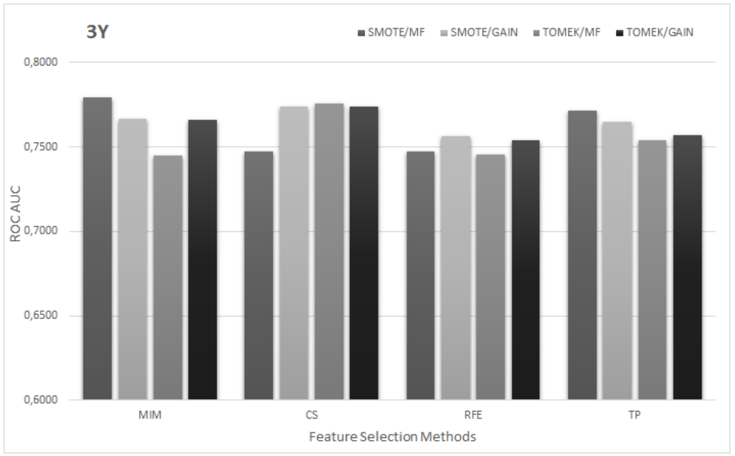
The MIDA method was the clear worst, but the MICE and OSM were not far behind the other two. Both these had specific setups where they outperformed their counterparts, but were slightly worse than the first two on average. The MICE method was also prone to poor specificity scores, particularly when used on the least balanced datasets (2Y and 3Y). Since there is a need to further restrict the amount of variables being used for testing, we opted to utilize exclusively the MF and GAIN methods in the following testing scenarios.

5.5 Determining the best method for Feature Selection

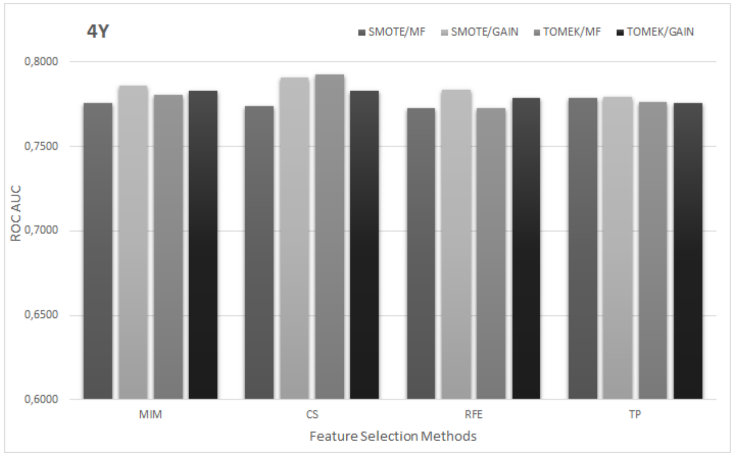
Finally, in this section, several methods for Feature Selection are compared. Again, the pipeline follows a similar setup to the previous ones. The four best performing classifiers were utilized (NB, SVM Poly, RF and NN) as well as the two best performing class balancing methods (SMOTE and TOMEK) and the two best performing MVI methods (MF and GAIN). Considering the four datasets, and the four FS methods being tested, this resulted in a total of 256 obtained measurements (or 64 setups to compare the performance of the FS methods). With this amount of information, we redirected our goals to group some of the results in order to find correlations or patterns, as well as having as many variables as possible so as to find the best setup configuration for each of the datasets.



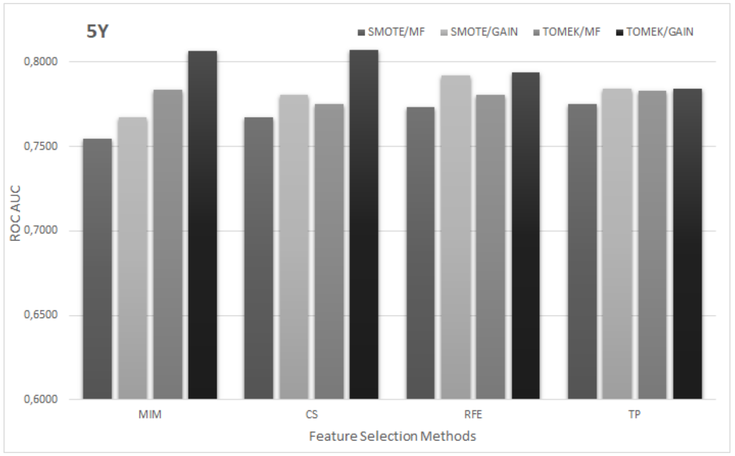
(a) 2 Year Dataset



(b) 3 Year Dataset



(c) 4 Year Dataset



(d) 5 Year Dataset

Figure 5.6: Comparison of the ROC AUC score obtained while using four different methods for FS (respectively MIM, CS, SVM RFE and the features used in T. Pereira's dissertation). Each FS method was tested in four setups that result of a combination of two CB methods (SMOTE and TOMEK) and two MVI methods (MF and GAIN). Each column contains the average of the scores obtained with the four best performing classifiers (NB, SVM Poly, RF and NN).

The vast amount of data collected makes it more difficult to illustrate. Figure 5.6 contains four graphs with the results for each of the training setups (4 datasets, 4 FS methods, 2 CB methods and 2 MVI methods). In order to ease understanding, each column contains the average of the scores obtained with the four different classifiers (NB, SVM Poly, RF and NN) for that exact setup. While it may ease the illustration of the results, grouping the data in such a manner can "dilute" the results, occluding some important outlier cases. Knowing this, all the following conclusions were drawn from the original data collected. Some of the conclusions noted, organized by the different variables tested, were:

- Classifier - When comparing between classifiers we could spot some of the same outcomes noted in previous sections, such as NN performing far better when paired with SMOTE rather than with TOMEK. While the different FS methods provided fairly consistent results, it was possible to notice some negative correlations. For example, the NB classifier was consistently worse when paired with the SVM RFE method. The same relation was noticeable between the NN classifier and the set of features used in T. Pereira's work (which can be relevant seeing as NNs were not utilized in T. Pereira's work).

- MVI Methods - While the MF method allowed for some of the best results recorded, the GAIN method slightly outperformed it on average. GAIN was significantly more consistent, and setups with GAIN presented smaller variability when swapping between datasets. The different FS methods did not seem to have a big impact on the performance of the MVI methods, with the notable exception being CS which returned significantly better results when paired with GAIN rather than MF.

- CB Methods - Generally the SMOTE method was the ideal choice for the more unbalanced datasets (2Y and 3Y), while TOMEK performed better with the more balanced ones (4Y and 5Y). SMOTE performed consistently regardless of which FS method it was paired with. TOMEK always performed better when paired with CS, and always worse when paired with SVM.

With every experiment performed so far, the difference in scores tends to be smaller and less noticeable, and that is evident in this set of measurements. Overall the four different techniques for FS performed comparably to each other, with only the SVM RFE performing slightly worse than all other methods.

While there was no clear best performing technique for FS, it was evident that some combinations of different parameters work significantly better than others, as was noted before. It is also relevant to point out the impact that each of these techniques had on the size of the datasets being tested. On average, the CS method resulted in datasets with the highest amount of features, while the features from T. Pereira were the smallest group (always 42 features).

5.6 Selecting the best Setup

The last set of experiments allowed us to test 64 different setups for each of the four datasets. Having all these measurements allows us to finally compare complete setups and determine the best ones. Table 5.3 contains the configurations of the setups that resulted in the highest ROC AUC scores for each of the datasets.

Table 5.3: Best pipeline setups for each of the datasets, according to data collected in Section 5.5, as well as the setup utilized in T. Pereira's work.

Dataset	Classifier	CB	MVI	FS
T. Pereira	NB	SMOTE	OSM	TP
2Y	NN	SMOTE	GAIN	CS
3Y	NB	SMOTE	MF	MIM
4Y	RF	TOMEK	MF	CS
5Y	NB	TOMEK	GAIN	CS

The different setups and the combinations of methods go along with several conclusions that were drawn over all the previous experiments. Such as:

- SMOTE performing better on more unbalanced datasets (2Y and 3Y) and TOMEK performing better on more balanced datasets (4Y and 5Y).
- NN performing better when paired with SMOTE.
- GAIN performing better when paired with CS.

It is also relevant to point out that the best setup for all the datasets on average is the setup that returned the best results for the 5Y dataset (utilizing NB, TOMEK, GAIN and CS). In Table 5.4 it is possible to compare the original results to the ones obtained with these setups. The original results were obtained in section 5.2 with the replica of T. Pereira's setup (utilizing NB, SMOTE and OSM). These can be compared with our two proposed setups: The first proposal being the best individual setup for each of the years, $i.e.$, the setups described in table 5.3 for each of the datasets; The second proposal being the best overall setup for all the datasets (NB, TOMEK, GAIN and CS).

Finally we decided to return to the testing scenarios utilized in T. Pereira's work, described in Section 5.1. In these scenarios the entire Lisbon CCC-3 dataset is utilized for training, while the Coimbra CCC-3 dataset is used as validation. Table 5.5 and table 5.6 contain the Cross-Validation results obtained from the Lisbon dataset, and the validation results obtained from the Coimbra dataset, respectively. In the Lisbon CV results (table 5.5) we can note a small improvement from the results obtained with replicated setup, although even the improved results lag behind those obtained in T. Pereira's dissertation. Nonetheless, as there is no way for us to recreate an exact replica of that setup, T. Pereira's results can only be used as a reference and not as direct comparison.

Table 5.4: Comparison between the original results obtained in section 5.2 and results obtained with two different proposed setups (best individual setup for each year and best overall setup, respectively).

Dataset	Setup	ROC AUC	Sensitivity	Specificity	Accuracy
2Y	Original	0.73	0.74	0.73	0.75
	Proposal #1	0.79	0.78	0.79	0.78
	Proposal #2	0.75	0.76	0.74	0.76
3Y	Original	0.77	0.77	0.78	0.76
	Proposal #1	0.79	0.77	0.82	0.78
	Proposal #2	0.79	0.77	0.81	0.78
4Y	Original	0.76	0.76	0.79	0.72
	Proposal #1	0.81	0.78	0.85	0.81
	Proposal #2	0.79	0.79	0.79	0.79
5Y	Original	0.76	0.76	0.81	0.71
	Proposal #1	0.82	0.84	0.79	0.81
	Proposal #2	0.82	0.84	0.79	0.81

In the Coimbra results (table 5.6) however, the results obtained from the proposed setups were equal or worse than those obtained with the replica setup. This discrepancy in results could suggest that our proposed setups are over-fit to the training dataset, but the fact that all the measurements were obtained using Cross Validation and an additional Validation set eliminates some of those concerns. Another possibility could be that the methods explored in this thesis managed to bring out more information from the features that are present in the Lisbon dataset but are entirely missing from the Coimbra dataset.

Table 5.5: Comparison between the results presented on Telma Pereira's PhD dissertation (T. Pereira), the results obtained while replicating that setup (Replica), and results obtained from the two best setups proposed in this thesis (Proposal #1 and Proposal #2). These scores were obtained from the CV process performed on the CCC-3 Lisbon dataset.

	AUC				Sensitivity				Specificity			
	2Y	3Y	4Y	5Y	2Y	3Y	4Y	5Y	2Y	3Y	4Y	5Y
T. Pereira	0.83	0.85	0.86	0.87	0.70	0.74	0.78	0.83	0.79	0.79	0.80	0.76
Replica	0.77	0.80	0.81	0.82	0.76	0.79	0.82	0.82	0.74	0.77	0.76	0.77
Proposal #1	0.78	0.80	0.82	0.84	0.78	0.79	0.79	0.83	0.74	0.83	0.80	0.77
Proposal #2	0.78	0.80	0.81	0.83	0.77	0.78	0.80	0.83	0.75	0.79	0.78	0.77

5.7 Summary

In this chapter we tested different methods and techniques to address each step of the classification pipeline. The setup used in T. Pereira's work was utilized to obtain a baseline, and several incremental experiments were performed in order to determine the best combination of methods to use at each step. Throughout the experiments we learned how some methods can interact with each other and what correlations can emerge in the results.

Table 5.6: Comparison between the results presented on Telma Pereira's PhD dissertation (T. Pereira), the results obtained while replicating that setup (Replica), and results obtained from the two best setups proposed in this thesis (Proposal #1 and Proposal #2). These scores were obtained from the validation dataset (CCC-3 Coimbra).

	AUC				Sensitivity				Specificity			
	2Y	3Y	4Y	5Y	2Y	3Y	4Y	5Y	2Y	3Y	4Y	5Y
T. Pereira	0.66	0.67	0.64	0.63	0.77	0.80	0.68	0.60	0.50	0.41	0.57	0.65
Replica	0.68	0.66	0.61	0.59	0.73	0.72	0.65	0.57	0.62	0.61	0.57	0.60
Proposal #1	0.63	0.66	0.61	0.53	0.76	0.71	0.64	0.55	0.50	0.60	0.59	0.52
Proposal #2	0.65	0.62	0.60	0.53	0.75	0.73	0.67	0.55	0.55	0.50	0.54	0.52

Throughout the experiments we progressively discarded the worst performing methods, leaving us with a manageable amount of variables to work with. By the end we presented two proposed setups made up of the best combinations of some of the best performing methods for each of their respective roles.

The obtained results allow us to confirm that our experiments were successful at incrementally improving the classification scores. It is not possible to unequivocally affirm that the final setups that we reached are the absolute best, as there could have been better setups composed of one or more methods that were discarded earlier. Nonetheless, our incremental testing approach minimizes such cases while reducing the total amount of testing scenarios to a manageable amount. As we can observe in Table 5.4, both the proposed setups ended up producing better results than the ones obtained with the original replicated setup.

6 Conclusion and Future Work

Contents

6.1 Conclusion

In this work we studied the prognosis and evolution of Alzheimer's Disease in patients initially diagnosed with Mild Cognitive Impairment. With the ever aging of the global population, predicting and monitoring the evolution of AD in patients surges in relevancy and importance. As such, being able to correctly diagnose AD at early stages and being able to predict its evolution can have significant social and economic impacts globally. To tackle this issue, we utilize data collected from Neuropsychological tests, as they provide a diagnosis tool that is both inexpensive and non-invasive to patients. However, this data comes with downsides, such as incongruities based on the collection sites and responsible teams, small sample sizes and unbalanced datasets. Our approach to overcome these problems was to research and implement state-of-the-art techniques and algorithms that address areas like Missing Value Imputation, Class Balancing, Feature Selection and Classification.

Initially we implemented six different algorithms to address MVI in datasets. In order to test those algorithms, we collected 10 datasets of diverse real-world data and set up a series of experiments. Each of those experiments allowed us to compare the different algorithms and draw conclusions based on their performance.

Afterwards we constructed a modular data mining pipeline, capable of swapping between different methods at each stage, so as to help determine the best overall setup. Before performing any tests, we had implemented a total of eight classifiers, five CB methods, six MVI methods and four FS techniques. In order to establish a baseline, we attempted to replicate the pipeline setup present in T. Pereira's work [1], which this work aims to improve upon. After replicating the pipeline, we recorded Accuracy, Sensitivity, Specificity and ROC AUC scores obtained on two different configurations of the datasets: The first where only the Lisbon CCC-3 dataset was used for both training and validation, and the second where the Lisbon CCC-3 set was used for training and the Coimbra CCC-3 for validation.

Having acquired baseline scores, we could now attempt to determine the best pipeline setup. To do so we adopted an incremental testing approach where, at each stage, all but one steps in the pipeline would be fixed so that the remaining step could be tested with the different methods available for it. After each testing iteration the worst performing methods would be discarded, leaving the best performing ones to be utilized in the next iteration. By the final set of tests, the available methods had been restricted to four classifiers, two CB methods, two MVI methods and four FS techniques. Pipelines with every possible combination of these methods were tested, allowing us to choose the combinations that resulted in the highest classification scores. From those scores we proposed two solutions to the original problem: The first proposal consists of the best setup for each individual dataset, with the setups being specified in table 5.3; The second proposal consists of the setup that resulted in the highest score on average for all the datasets, and it utilizes the NB classifier, TOMEK Links to overcome class imbalance, GAIN to address MVI and CS for FS.

Both proposed setups ended up returning higher scores than the established baseline. This is true for the validation scores in the first dataset configuration (only using the Lisbon CCC-3 dataset) as well as the cross-validation testing scores in the second dataset configuration (using Lisbon and Coimbra CCC-3 datasets). However, the proposed setups were not able to overcome the baseline validation scores (Coimbra CCC-3 dataset) of the second configuration. To ensure the validity and robustness of the results we utilized different approaches such as using 10×5 Fold Cross-Validation, using independent validation sets and collecting four different scoring metrics so as to better understand the results.

6.2 Future Work

In this thesis we explored state-of-the-art techniques to address several processes in the data mining pipeline. Nevertheless, there are plenty more state-of-the-art models and techniques that we did not study or implement, with new approaches constantly emerging. Studying and applying such techniques could improve the quality of the overall system. Particularly regarding the classification process, we opted to mostly replicate the classifiers used in T. Pereira's work [1] and did not attempt to significantly improve upon them. There is promising research being done in this field, specifically regarding Neural Networks and Deep Learning, that could lead to improvements in our pipeline.

Also regarding Neural Networks, an interesting approach to our problem could be an end-to-end Neural Network architecture. By this we mean a solution that would encapsulate the entire pipeline, from data input to prediction output. Such a model would perform both MVI and classification, which could be of great value seeing how relevant MVI is to this problem.

Finally, despite having tested a range of available models for each step in our pipeline, the order of these steps remained constant throughout all the tests. Modifying the order in which some steps are executed or adding some other variance to the pipeline could lead to improved results.

Bibliography

[1] T. Filipa, L. De, M. Pereira, S. Alexandra, and C. Madeira, “Prognostic models targeting time to conversion, stable predictors, and reliability at patient-level: Predicting progression from mild cognitive impairment to dementia Biomedical Engineering,” no. February, 2019.

[2] A. Association, “2019 Alzheimer ’ s disease facts and figures,” *Alzheimer’s & Dementia*, vol. 15, no. 3, pp. 321–387, 2019. [Online]. Available: https://doi.org/10.1016/j.jalz.2019.01.010

[3] P. M. Prince, G.-c. Ali, and G.-c. Ali, “World Alzheimer Report 2015 The Global Impact of Dementia,” 2015.

[4] L. Lemos, “A data mining approach to predict conversion from mild cognitive impairment to Alzheimer ’ s Disease.” Ph.D. dissertation, 2012.

[5] T. Pereira, L. Lemos, S. Cardoso, D. Silva, A. Rodrigues, I. Santana, A. De Mendonça, M. Guerreiro, and S. C. Madeira, “Predicting progression of mild cognitive impairment to dementia using neuropsychological data: A supervised learning approach using time windows,” *BMC Medical Informatics and Decision Making*, vol. 17, no. 1, pp. 1–15, 2017.

[6] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.

[7] D. Koller and M. Sahami, “Toward optimal feature selection,” Stanford InfoLab, Tech. Rep., 1996.

[8] P. A. Estevez, M. Tesmer, C. A. Perez, and J. M. Zurada, “Normalized mutual information feature selection,” *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[9] P. Viola and W. M. Wells III, “Alignment by maximization of mutual information,” *International journal of computer vision*, vol. 24, no. 2, pp. 137–154, 1997.

[10] C. Ding and H. Peng, “Minimum redundancy feature selection from microarray gene expression data,” *Journal of bioinformatics and computational biology*, vol. 3, no. 02, pp. 185–205, 2005.

[11] H. Yang and J. Moody, “Feature selection based on joint mutual information,” in *Proceedings of international ICSC symposium on advances in intelligent data analysis*. Citeseer, 1999, pp. 22–25.

[12] A.-M. Bidgoli and M. N. Parsa, “A hybrid feature selection by resampling, chi squared and consistency evaluation techniques,” *World Academy of Science, Engineering and Technology*, vol. 68, pp. 276–285, 2012.

[13] K. Yan and D. Zhang, “Feature selection and analysis on correlated gas sensor data with recursive feature elimination,” *Sensors and Actuators B: Chemical*, vol. 212, pp. 353–363, 2015.

[14] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, “Random forest: a classification and regression tool for compound classification and qsar modeling,” *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 6, pp. 1947–1958, 2003, pMID: 14632445. [Online]. Available: https://doi.org/10.1021/ci034160g

[15] R. Nallapati, “Discriminative models for information retrieval,” in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 64–71.

[16] A. Y. Ng and M. I. Jordan, “On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes,” in *Advances in neural information processing systems*, 2002, pp. 841–848.

[17] T. Hasanin and T. Khoshgoftaar, “The effects of random undersampling with simulated class imbalance for big data,” in *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, July 2018, pp. 70–79.

[18] D. Devi, B. Purkayastha *et al.*, “Redundancy-driven modified tomek-link based undersampling: a solution to class imbalance,” *Pattern Recognition Letters*, vol. 93, pp. 3–12, 2017.

[19] C.-F. Tsai, W.-C. Lin, Y.-H. Hu, and G.-T. Yao, “Under-sampling class imbalanced datasets by combining clustering analysis and instance selection,” *Information Sciences*, vol. 477, pp. 47 – 54, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020025518308478

[20] S. del Río, J. M. Benítez, and F. Herrera, “Analysis of data preprocessing increasing the oversampling ratio for extremely imbalanced big data classification,” in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 2. IEEE, 2015, pp. 180–185.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: synthetic minority oversampling technique,” *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[22] Y. E. Kurniawati, A. E. Permanasari, and S. Fauziati, “Adaptive synthetic-nominal (adasyn-n) and adaptive synthetic-knn (adasyn-knn) for multiclass imbalance learning on laboratory test data,” in *2018 4th International Conference on Science and Technology (ICST)*, Aug 2018, pp. 1–6.

[23] K. E. Bennin, J. Keung, P. Phannachitta, A. Monden, and S. Mensah, “Mahakil: Diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction,” *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 534–550, 2018.

[24] P. C. Mahalanobis, “On the generalized distance in statistics,” 1936.

[25] I. Rish *et al.*, “An empirical study of the naive bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, no. 22, 2001, pp. 41–46.

[26] J. R. Quinlan, “Induction of decision trees,” *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[27] L. E. Raileanu and K. Stoffel, “Theoretical comparison between the gini index and information gain criteria,” *Annals of Mathematics and Artificial Intelligence*, vol. 41, no. 1, pp. 77–93, 2004.

[28] L. E. Peterson, “K-nearest neighbor,” *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[29] C.-F. Lin and S.-D. Wang, “Fuzzy support vector machines,” *IEEE transactions on neural networks*, vol. 13, no. 2, pp. 464–471, 2002.

[30] O. Chapelle, “Training a support vector machine in the primal,” 2006.

[31] M. Hossin and M. Sulaiman, “A review on evaluation metrics for data classification evaluations,” *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, p. 1, 2015.

[32] J. Huang and C. X. Ling, “Using auc and accuracy in evaluating learning algorithms,” *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[33] Y. Fan, S. M. Resnick, X. Wu, and C. Davatzikos, “Structural and functional biomarkers of prodromal alzheimer’s disease: A high-dimensional pattern classification study,” *NeuroImage*, vol. 41, no. 2, pp. 277 – 285, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S105381190800178X

[34] Y. Zhang, Z. Dong, P. Phillips, S. Wang, G. Ji, J. Yang, and T.-F. Yuan, “Detection of subjects and brain regions related to alzheimer’s disease using 3d mri scans based on eigenbrain and machine learning,” *Frontiers in Computational Neuroscience*, vol. 9, p. 66, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2015.00066

[35] R. C. Petersen, P. S. Aisen, L. A. Beckett, M. C. Donohue, A. C. Gamst, D. J. Harvey, C. R. Jack, W. J. Jagust, L. M. Shaw, A. W. Toga, J. Q. Trojanowski, and M. W. Weiner, “Alzheimer’s disease neuroimaging initiative (adni),” *Neurology*, vol. 74, no. 3, pp. 201–209, 2010. [Online]. Available: https://n.neurology.org/content/74/3/201

[36] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies (oasis): cross-sectional mri data in young, middle aged, nondemented, and demented older adults," *Journal of cognitive neuroscience*, vol. 19, no. 9, pp. 1498–1507, 2007.

[37] D. Zhang and D. Shen, "Semi-supervised multimodal classification of alzheimer's disease," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 2011, pp. 1628–1631.

[38] L. R. Trambaiolli, A. C. Lorena, F. J. Fraga, P. A. Kanda, R. Anghinah, and R. Nitrini, "Improving alzheimer's disease diagnosis with machine learning techniques," *Clinical EEG and Neuroscience*, vol. 42, no. 3, pp. 160–165, 2011, pMID: 21870467. [Online]. Available: https://doi.org/10.1177/155005941104200304

[39] P. Gurevich, H. Stuke, A. Kastrup, H. Stuke, and H. Hildebrandt, "Neuropsychological testing and machine learning distinguish alzheimer's disease from other causes for cognitive impairment," *Frontiers in aging neuroscience*, vol. 9, p. 114, 2017.

[40] J. L. Shaffer, J. R. Petrella, F. C. Sheldon, K. R. Choudhury, V. D. Calhoun, R. E. Coleman, and P. M. a. Doraiswamy, "Predicting cognitive decline in subjects at risk for alzheimer disease by using combined cerebrospinal fluid, mr imaging, and pet biomarkers," *Radiology*, vol. 266, no. 2, pp. 583–591, 2013, pMID: 23232293. [Online]. Available: https://doi.org/10.1148/radiol.12120010

[41] B. Zhou, E. Nakatani, S. Teramukai, Y. Nagai, M. Fukushima, A. D. N. Initiative *et al.*, "Risk classification in mild cognitive impairment patients for developing alzheimer's disease," *Journal of Alzheimer's Disease*, vol. 30, no. 2, pp. 367–375, 2012.

[42] Y. Wang, C. Xu, J.-H. Park, S. Lee, Y. Stern, S. Yoo, J. H. Kim, H. S. Kim, J. Cha, A. D. N. Initiative *et al.*, "Diagnosis and prognosis of alzheimer's disease using brain morphometry and white matter connectomes," *NeuroImage: Clinical*, vol. 23, p. 101859, 2019.

[43] T. Jo, K. Nho, and A. J. Saykin, "Deep learning in alzheimer's disease: diagnostic classification and prognostic prediction using neuroimaging data," *Frontiers in aging neuroscience*, vol. 11, p. 220, 2019.

[44] A. R. T. Donders, G. J. van der Heijden, T. Stijnen, and K. G. Moons, "Review: A gentle introduction to imputation of missing values," *Journal of Clinical Epidemiology*, vol. 59, no. 10, pp. 1087 – 1091, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0895435606001971

[45] Y. Fu, H. S. He, T. J. Hawbaker, P. D. Henne, Z. Zhu, and D. R. Larsen, "Evaluating k-nearest neighbor (knn) imputation models for species-level aboveground forest biomass mapping in northeast china," *Remote Sensing*, vol. 11, no. 17, p. 2005, 2019.

[46] A. Wang, Y. Chen, N. An, J. Yang, L. Li, and L. Jiang, "Microarray missing value imputation: A regularized local learning method," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 16, no. 3, pp. 980–993, 2019.

[47] D. J. Stekhoven and P. Bühlmann, "MissForest—non-parametric missing value imputation for mixed-type data," *Bioinformatics*, vol. 28, no. 1, pp. 112–118, 10 2011. [Online]. Available: https://doi.org/10.1093/bioinformatics/btr597

[48] P. Royston, I. R. White *et al.*, "Multiple imputation by chained equations (mice): implementation in stata," *J Stat Softw*, vol. 45, no. 4, pp. 1–20, 2011.

[49] Y. Yuan, "Multiple imputation for missing data: Concepts and new development," 01 2005.

[50] P. J. García-Laencina, J.-L. Sancho-Gómez, and A. R. Figueiras-Vidal, "Pattern classification with missing data: a review," *Neural Computing and Applications*, vol. 19, no. 2, pp. 263–282, 2010.

[51] H. Zhang, P. Xie, and E. Xing, "Missing value imputation based on deep generative models," 2018.

[52] Y. L. Qiu, H. Zheng, and O. Gevaert, "A deep learning framework for imputing missing values in genomic data," *bioRxiv*, 2018. [Online]. Available: https://www.biorxiv.org/content/early/2018/09/03/406066

[53] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '8. New York, NY, USA: Association for Computing Machinery, 2008, p. 1096–1103. [Online]. Available: https://doi.org/10.1145/1390156.1390294

[54] L. Gondara and K. Wang, "Mida: Multiple imputation using denoising autoencoders," 2017.

[55] P.-A. Mattei and J. Frellsen, "Miwae: Deep generative modelling and imputation of incomplete data," *arXiv preprint arXiv:1812.02633*, 2018.

[56] J. T. McCoy, S. Kroon, and L. Auret, "Variational autoencoders for missing data imputation with application to a simulated milling circuit," *IFAC-PapersOnLine*, vol. 51, no. 21, pp. 141–146, 2018.

[57] H. Zhang and D. P. Woodruff, "Medical missing data imputation by stackelberg gan."

[58] J. Kim, D. Tae, and J. Seok, "A survey of missing data imputation using generative adversarial networks," in *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2020, pp. 454–456.

[59] J. Yoon, J. Jordon, and M. van der Schaar, "Gain: Missing data imputation using generative adversarial nets," 2018.

[60] J. Zhang, X. Mu, J. Fang, and Y. Yang, “Time series imputation via integration of revealed information based on the residual shortcut connection,” *IEEE Access*, vol. 7, pp. 102 397–102 405, 2019.

[61] F. Biessmann, D. Salinas, S. Schelter, P. Schmidt, and D. Lange, “”deep” learning for missing value imputationin tables with non-numerical data,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM ’18. New York, NY, USA: ACM, 2018, pp. 2017–2025. [Online]. Available: http://doi.acm.org/10.1145/3269206.3272005

[62] Y. Nesterov, “A method of solving a convex programming problem with convergence rate

$$o(1/k2)$$

o (1/k2),” in *Sov. Math. Dokl*, vol. 27, no. 2.

[63] I. S. Van Maurik, M. D. Zwan, B. M. Tijms, F. H. Bouwman, C. E. Teunissen, P. Scheltens, M. P. Wattjes, F. Barkhof, J. Berkhof, and W. M. Van Der Flier, “Interpreting biomarker results in individual patients with mild cognitive impairment in the alzheimer’s biomarkers in daily practice (abide) project,” *Jama Neurology*, vol. 74, no. 12, pp. 1481–1491, 2017.

[64] B. McGuinness, S. L. Barrett, J. McIlvenna, A. P. Passmore, and G. W. Shorter, “Predicting conversion to dementia in a memory clinic: A standard clinical approach compared with an empirically defined clustering method (latent profile analysis) for mild cognitive impairment subtyping,” *Alzheimer’s & Dementia: Diagnosis, Assessment & Disease Monitoring*, vol. 1, no. 4, pp. 447–454, 2015.

[65] W. G. Rosen, R. C. Mohs, and K. L. Davis, “A new rating scale for alzheimer’s disease.” *The American journal of psychiatry*, 1984.

[66] M. F. Folstein, S. E. Folstein, and P. R. McHugh, ““mini-mental state”: a practical method for grading the cognitive state of patients for the clinician,” *Journal of psychiatric research*, vol. 12, no. 3, pp. 189–198, 1975.

[67] D. Silva, M. Guerreiro, J. Maroco, I. Santana, A. Rodrigues, J. B. Marques, and A. de Mendonça, “Comparison of four verbal memory tests for the diagnosis and predictive value of mild cognitive impairment,” *Dementia and geriatric cognitive disorders extra*, vol. 2, no. 1, pp. 120–131, 2012.

[68] A. I. d. S. Rodrigues, “Ineco frontal screening (IFS): sensibilidade e avaliação breve das funções executivas na doença de parkinson,” 2011.

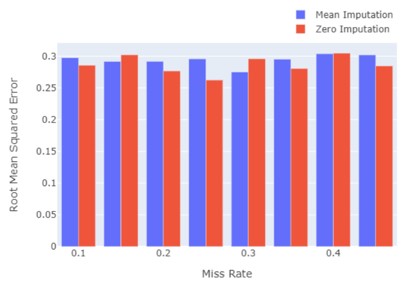
[69] D. Silva, I. Santana, F. S. d. Couto, J. Maroco, M. Guerreiro, and A. de Mendonça, “Cognitive deficits in middle-aged and older adults with bipolar disorder and cognitive complaints: Comparison

with mild cognitive impairment," *International Journal of Geriatric Psychiatry*, vol. 24, no. 6, pp. 624–631, 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/gps.2166

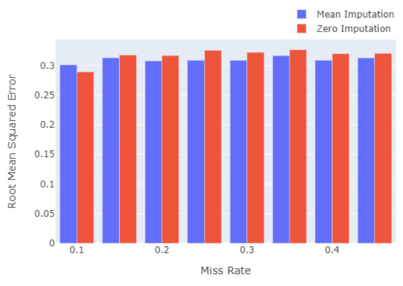
[70] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

A

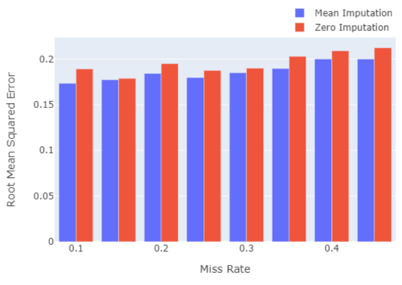
Additional Results for Section 3.3.1.B



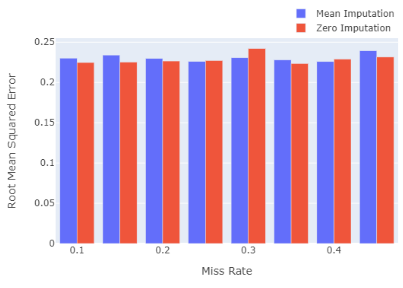
(a) WH



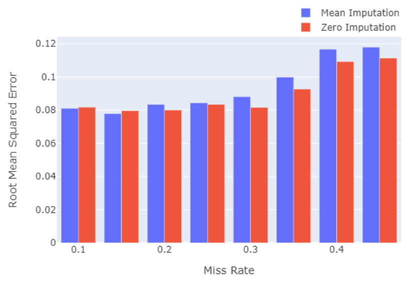
(b) HD



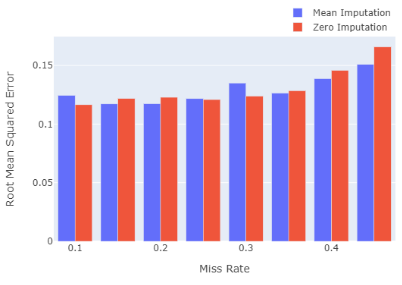
(c) BH



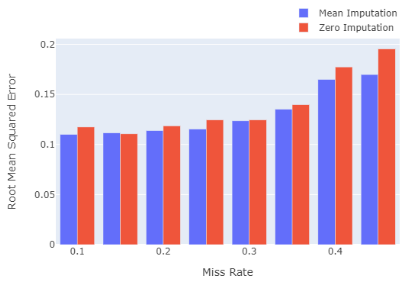
(d) PI



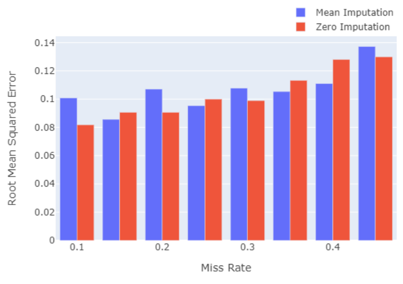
(e) BC



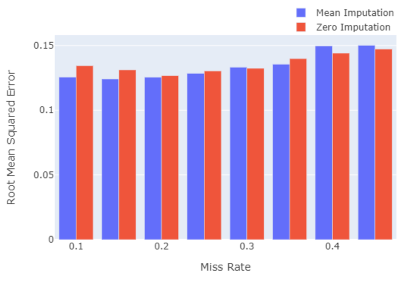
(f) RW



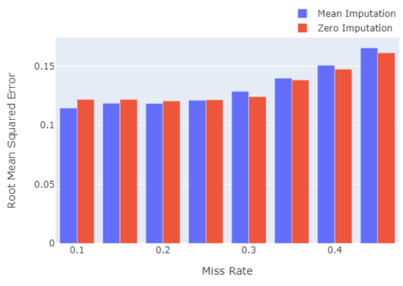
(g) GR



(h) PS



(i) KC

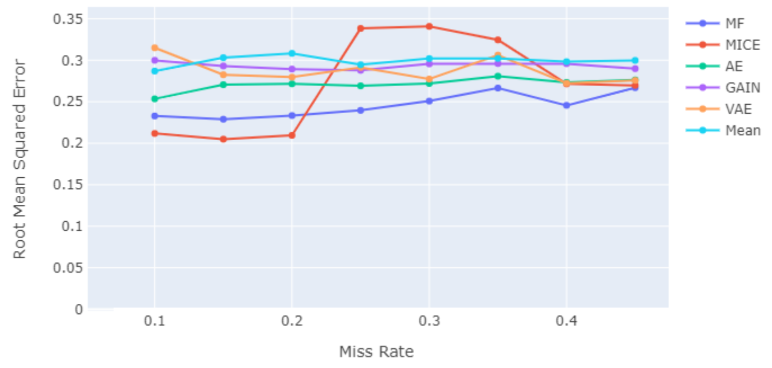


(j) UF

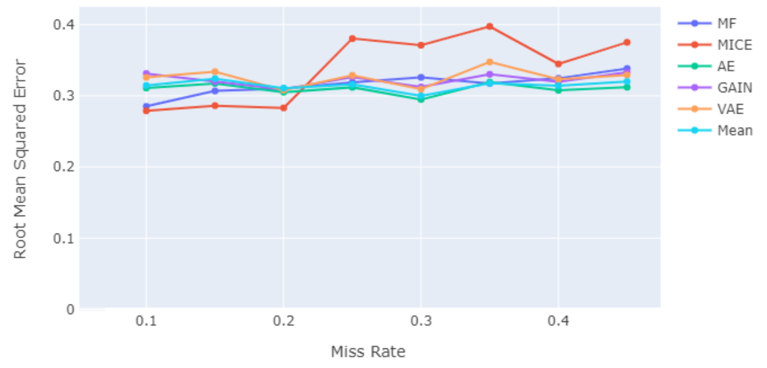
Figure A.1: RMSE for imputation performed by the GAIN method (with architecture based on GAN) on the ten different datasets, for different missing rates. In blue are the results when utilizing Mean-Imputation initially and in red the results for utilizing Zero-Imputation initially

B

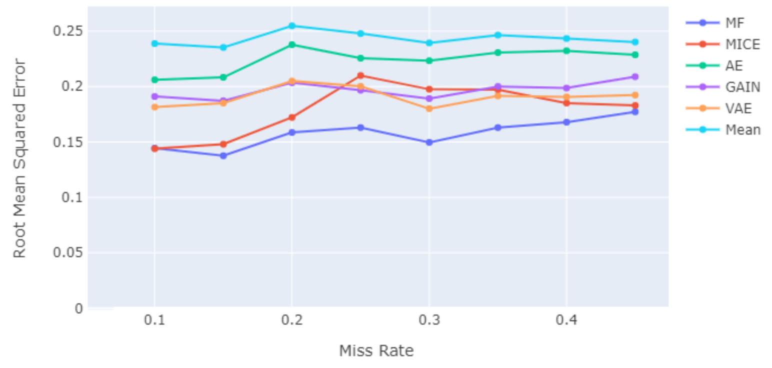
Additional Results for Section 3.3.3.B



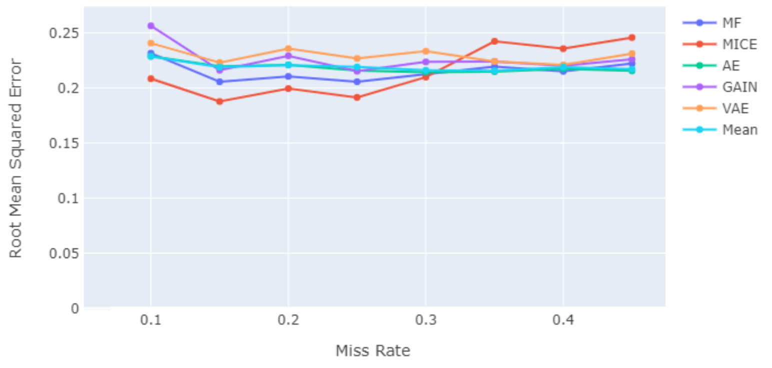
(a) WH



(b) HD

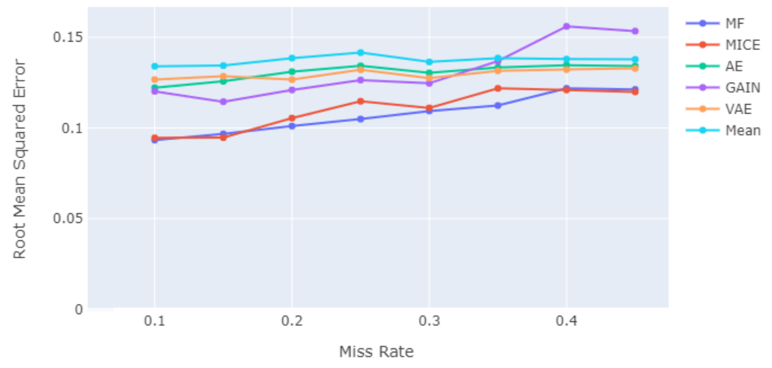


(c) BH

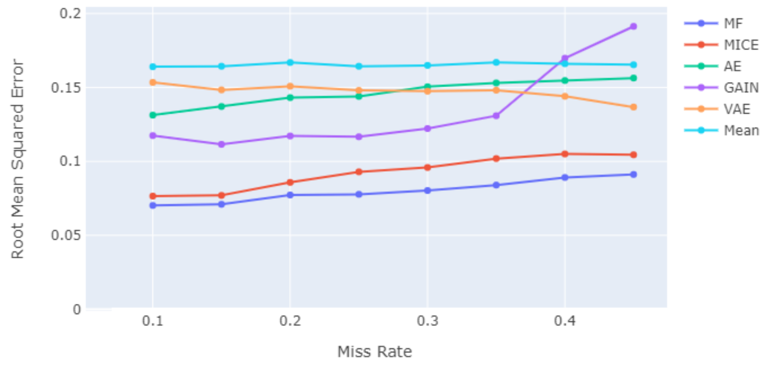


(d) PI

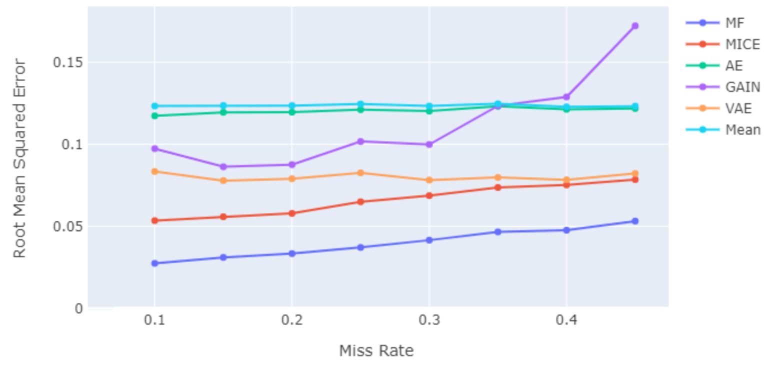
Figure B.1: RMSE for imputation performed by all the MVI methods (MF, MICE, MIDA, GAIN, MIWAE, OSM) on the ten datasets for values of missing data $\in [0.1, 0.45]$.



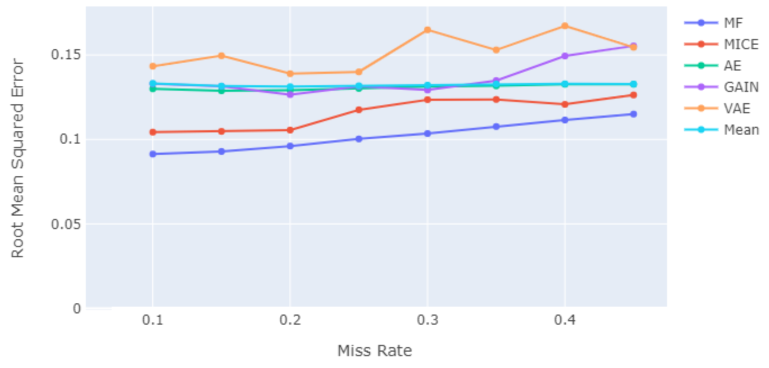
(a) RW



(b) GR



(c) PS

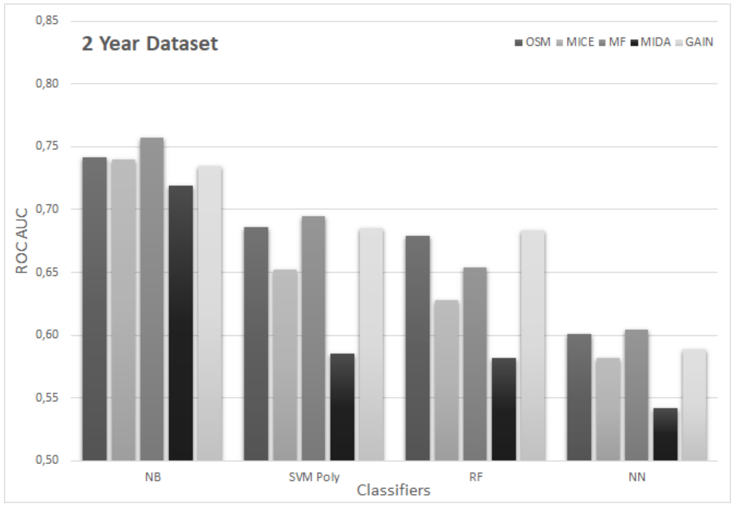


(d) KC

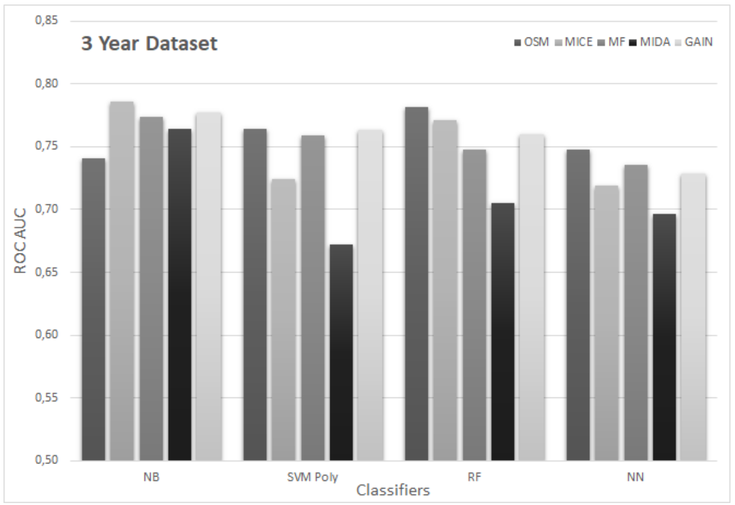
Figure B.1: RMSE for imputation performed by all the MVI methods (MF, MICE, MIDA, GAIN, MIWAE, OSM) on the ten datasets for values of missing data $\in [0.1, 0.45]$.

C

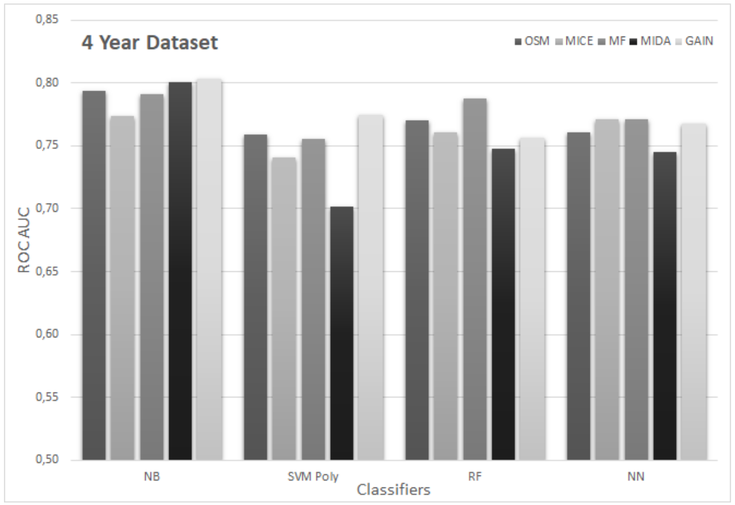
Additional Results for Section 5.2.4



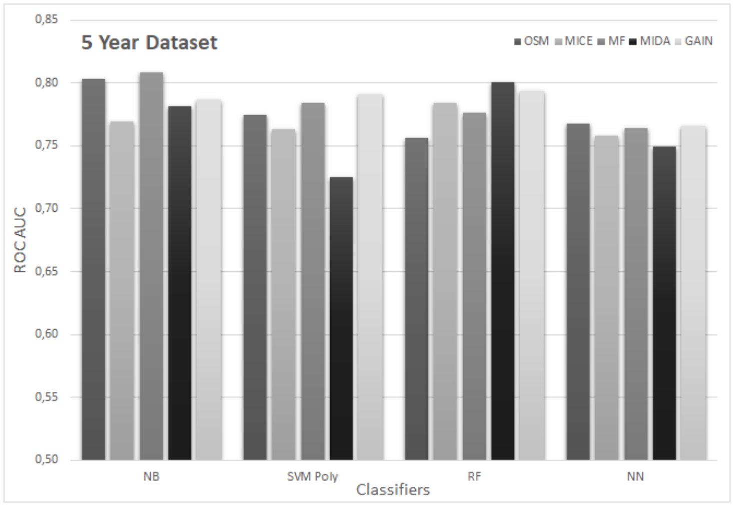
(a) 2 Year Dataset



(b) 3 Year Dataset



(c) 4 Year Dataset



(d) 5 Year Dataset

Figure C.1: Comparison of the ROC AUC score obtained while using five different methods for MVI (respectively Overall Sample Mean, MICE, MissForest, MIDA and GAIN). The results were obtained using the four best performing classifiers mentioned in Section 5.2 (NB, SVM Poly, RF and NN) and the TOMEK method for class balancing.