Master's Thesis 2022 30 ECTS
Fakultet for realfag og teknologi - REALTEK

Machine learning for detecting biomarkers of Alzheimer's disease: Data-centric approach with dynamic ensemble selection

Muhammad Muntazir Naqvi
Data Science

Machine learning for detecting biomarkers of Alzheimer's disease: Data-centric approach with dynamic ensemble selection

MUHAMMAD MUNTAZIR NAQVI

Master's Thesis 2022
Supervisor: Associate Prof. Oliver Tomic
Faculty: REALTEK
Norwegian University of Life Sciences
Ås, Norway

Abstract

Alzheimer's disease (AD) is a neurodegenerative disorder that progresses over time and results in gradual loss of cognitive abilities. It affects the patient to an extent that they become unable to perform daily routine tasks, eventually causing death. Alzheimer's disease is a significant health issue and it has no cure. Early detection of AD at preclinical or non-symptomatic stage allows for treatments that can slow down the progression of the disease. One of the biomarkers of AD that are measurable as early as in pre-clinical stage are deposits of amyloid beta peptides between neurons. In this study, our objective is to use machine learning and build a classification model to predict the presence of amyloid beta deposits given the patients' health history, and results of medical and cognitive ability tests. We build on the work done previously with the same data, and we follow a data-centric approach. The data is divided into five blocks based on the similarities between features in each block. We then plan a set of 17 data iterations, with each iteration using a different combination of five data transformation steps, i.e. (i) standardization, (ii) data distribution transformation, (iii) feature selection, (iv) oversampling, and (v) manifold learning. We repeat these iterations on four data blocks and train a dynamic ensemble selection classifier for each resulting dataset. We use Matthews Correlation Coefficient (MCC) as the primary performance metric to measure model performance. We also report five other performance metrics (accuracy, area under the ROC curve, F1 score, precision, and recall) to provide a comprehensive picture of model performance. We see that data iterations giving the best performance on training data mostly include transformation of data to a normal distribution, feature selection, and oversampling. However, the performance on test data varies greatly with the type of data (data block) and it is not clear which data iteration gives the best performance. In addition, the predictive performance is not very satisfactory for nearly all of the models and they suffer from overfitting. We believe that more research is needed on this data to determine the best performing classification approach.

Keywords: Alzheimer's disease, data-centric machine learning, data transformation, feature selection, oversampling, manifold learning, dynamic ensemble selection

Acknowledgements

This thesis marks the end of two years that I spent as a student at NMBU. Looking back, there are many people to whom I would like to extend my deepest gratitude. My special thanks to Associate Professor Oliver Tomic, my supervisor, whose has been immensely supportive and encouraging. His teaching in applied machine learning courses and guidance during this thesis enabled me to complete this work. I would also like to thank my co-supervisor, Professor Cecilia Marie Futsæther for her encouragement and guidance. I would be remiss if I did not thank Associate Professor Kristian Hovde Liland for his teaching in applied machine learning courses. I also want to thank other teachers during my graduate studies, especially Professor Hans Ekkehard Plesser, head of Data Science Department at REALTEK; Associate Professor Olvar Bergland, School of Economics and Business; and Lars-Gustav Snipen, Faculty of Chemistry, Biotechnology and Food Science; whose taught courses have contributed tremendously to my skills in programming and applied statistics.

My deepest thanks to my wife Iffat who has been my *wing woman* for over 6 years and counting. Her support and patience during my graduate studies has been invaluable. Special thanks to my father, Muhammad Abbas Naqvi and my mother, Naseem Zehra, whose love and prayers have got me through.

Muntazir Naqvi

August 15, 2022

Contents

1

Introduction

1.1 Alzheimer's disease

The deterioration of brain's cognitive function due to abnormal changes in the brain are referred to as *dementia*. Dementia is an umbrella term encompassing a wide range of medical conditions that lead to the decline in cognitive abilities. World Health Organization (WHO) reports that nearly 55 million people around the world currently have dementia and it is the seventh leading cause of death among all biological disorders [1]. WHO further states that 60-70% of dementia cases are because of Alzheimer's disease.

Alzheimer's disease (AD) is a neurodegenerative disorder that progresses over time and results in gradual loss of cognitive abilities. AD is characterized by loss of memory and decline in thinking and language skills to an extent that the affected individual becomes unable to perform daily routine tasks or operate independently. Figure 1.1 shows the physiological changes that happen to the brain during AD. The treatment options/drugs approved for Alzheimer's are only effective in symptomatic treatment and improvement in quality of life [2] but cannot cure the disease. Alzheimer's disease consists of essentially four stages viz. pre-clinical, mild, moder-

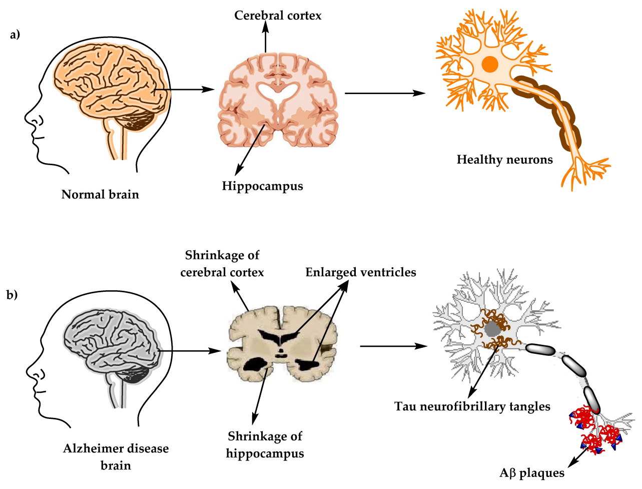


Figure 1.1: Physiology of **a)** normal brain vs. **b)** brain affected by Alzheimer's disease.[1]

ate, and severe. In pre-clinical stage, the disease begins to develop years (even more than a decade) before producing any clinical symptoms. Changes in the brain start to happen that make healthy neurons stop functioning, lose neural connections and die. The damage continues to spread over time and symptoms of mild Alzheimer's begin to appear such as memory loss, personality changes, behavioural changes etc. In moderate AD, damage spreads to areas of the brain responsible for language, senses, thinking ability. Affected person may also experience visions, delusions and problems in recognizing family and friends. This is followed by severe AD where damage spreads throughout the brain. Brain shrinks significantly resulting in the loss of ability to communicate and to perform even simple bodily functions such as swallowing food, ultimately leading to death [4].

Physiological changes in the brain that are characteristic features of AD and are measurable as early as in pre-clinical stage are *neuritic plaques* and *neurofibrillary tangles*. Neuritic plaques (also called Aβ plaques) are abnormal deposits of amyloid beta (Aβ) peptides[2] that accumulate between the neurons and disturb their function. Neurofibrillary tangles (NFTs) are deposits of *tau* protein inside neurons. Tau proteins in healthy neurons act as a stabilizer for microtubules[3]. In AD, tau molecules are detached from microtubules and tangle with each other to disrupt the transport system of neurons resulting in cognitive decline and neuronal loss [5]. Figure 1.2 illustrates the formation of NFTs. Apart from Aβ plaques and NFTs, other

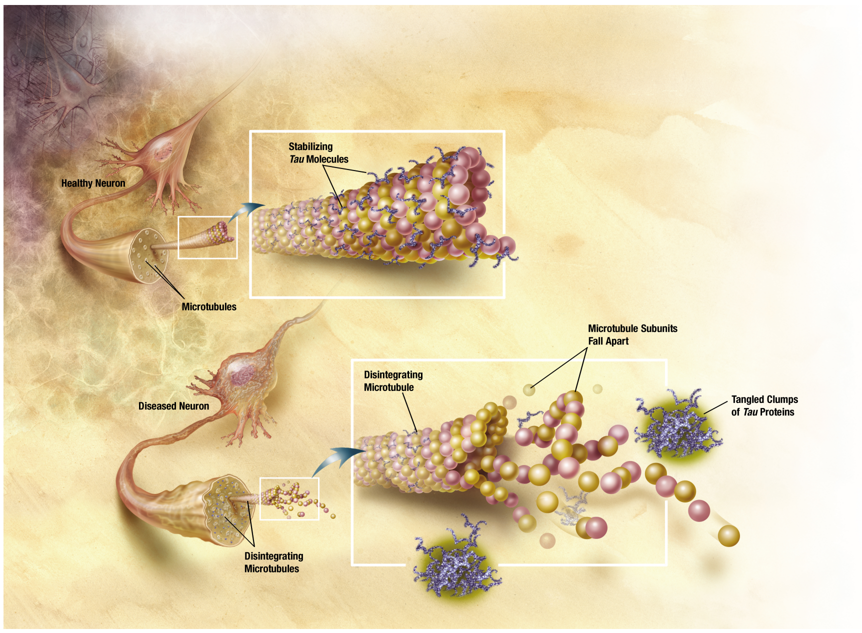


Figure 1.2: Illustration of the formation of tau neurofibrillary tangles.[4]

factors that contribute to neurodegeneration are oxidative stress, neuroinflammation, injury to cholinergic neurons (neurons that use acetylcholine neurotransmitter for communication), increasing age, head injury, infections, genetic and epigenetic factors, physiological disorders (including diabetes, obesity, and cardiovascular diseases) and environments factors [6], [7].

1.2 Diagnosis of AD and therapeutic strategies

Various tests are carried out to diagnose Alzheimer's disease including family history, medical history, neurological assessment, vitamin B12 estimation, MRI for neuronal examination and other tests. Association of vitamin B12 with the risk of AD development and progression has been reported in previous studies and its deficiency can damage the brain through oxidative stress. Estimation of vitamin B12 using patient's serum sample with serum homocysteine level and complete blood count is used to diagnose the vitamin B12 deficiency [8]. AD can also be diagnosed through identification of biomarkers including brain amyloid markers and neuronal injury markers. Amyloid markers of brain consist of cerebrospinal fluid (CSF) and positron emission tomography (PET) while markers of neuronal injury involve fluorodeoxyglucose (FDG) for metabolic activity, cerebrospinal fluid tau and MRI for atrophy (degeneration or shrinkage of nerve tissues) measurement [9].

Alzheimer's disease is a significant health issue whose cure has not yet been found. Various therapeutic approaches including drug and non-drug options including exercise and diet are recently in use to control and improve the symptoms of AD. Various methods to understand the pathology of AD have been proposed to develop successful therapeutic strategies including damage of free radical and cholinergic neurons, inflammatory response, and abnormal tau and amyloid protein metabolism [10], [11].

Early detection of Alzheimer's disease at preclinical or non-symptomatic stage is possible using imaging techniques such as positron emission tomography and resting state function MRI (rs-fMRI) to assess the brain ageing, its activity and accuracy of brain's cognitive functions. These imaging techniques use the markers such as genetic determinant and Aβ-peptide pathology for evaluating the neuronal activity. Through potential genetic mutations it is estimated that the individual is susceptible to progressive neurodegeneration and memory loss in later ages or not. Significant genetic alterations with aberrant Aβ pathology increases the chances of developing neurodegenerative disorders and accelerating the brain ageing. Accumulation of Aβ-peptide deposits also triggers another hallmark of AD which is tau protein tangles formation. Therefore, to stop the progression of AD, early detection is the only option [9].

1.3 Machine learning for early detection of AD

Several studies have been done in the domain of machine learning for early detection of Alzheimer's disease. Sharma and Mandal [12] provides a survey of research done on early diagnosis of Alzheimer's using neuroimaging data. They provide a summary of research articles that reviewed the work done in the domain of machine learning for diagnosing AD.

Most recently, Mezrar and Bendella [13] developed a cognitive tool called AlzCoGame. They claim that they overcame the limitations of traditional diagnostic techniques or tests by employing gamification techniques and machine learning in AlzCoGame.

They further report that they validated their model's performance using K-fold cross validation and multiple classification metrics.

Work by Ghazal et. al. [14] used MRI images for multi-class classification. They define the classes as four stages (four classes) of dementia viz. no dementia, very mild dementia, mild dementia, and moderate dementia. They report that their proposed approach which uses convolutional neural network (CNN) for image classification performs with over 91 percent accuracy.

Pirrone et. al. [15] report using supervised classification for analysing Electroencephalography (EEG) signals to predict the presence of biomarkers of cognitive impairment. They used multi-class classification for three classes of patients, i.e. healthy control group, patients with mild cognitive impairment, and patients diagnosed as having Alzheimer's. They also used one-versus-rest (OVR) approach and using binary classification for prediction of the three classes. They claim to have achieved accuracies of 80-90 percent with OVR approach and 75 percent with three-class classification approach.

Mofrad et. al. [16] present a framework for building models to predict cognitive decline using MRI images. They also employ the statistical approach of mixed effects model in addition to machine learning. The focus of this study was to predict conversion from normal cognitive function to mild cognitive impairment, and conversion from mild cognitive impairment to the diagnosis of Alzheimer's. They report that in case of conversion from MCI to AD, the model is 75 percent accurate in its prediction.

Revathi et. al. [17] put forward a two-stage classification approach. They use physical health factors in the first stage and cognitive ability test results in the second stage. In the first stage, they use support vector machines and random forest to estimate the influence of physical health factors (specifically hypertension and diabetes) on cognitive decline. In the second stage, they use multi-class logistic regression on the results of cognitive ability test to predict cognitive impairment in later stages of a subject's life. The target classes used in this study are no Alzheimer's, uncertain Alzheimer's and, definite Alzheimer's and accuracy of 89 percent is reported in the second stage.

Mohammed et. al. [18] based their research on Open Access Series of Imaging Studies (OASIS) dataset[5] and MIR image dataset from Kaggle[6]. They used transfer learning using two pre-trained convolutional neural networks, i.e., AlexNet and ResNet-50 combined with support vector machines. They also used other algorithms including t-distributed stochastic neighbour embedding (t-SNE) and classification algorithms including decision tree, random forest, and k-nearest neighbours. They report that the hybrid model (AlexNet combined with support vector machines) provided nearly 95 percent accuracy.

Mirzaei and Adeli [19] reviewed the research work done for detection of AD using ML. They reviewed the works that used a wide range of techniques including support vector machines, random forest classification, CNN, K-means clustering etc. They conclude that CNN approaches have shown to be the most promising. They also highlight ongoing efforts to build prediction models that use enhanced probabilistic neural networks, dynamic classification among many others.

1.4 Our work

In this thesis, we build on the block-wise analysis performed previously by Olofsson [20]. Data collection and sources have already been discussed by Olofsson in their thesis. Olofsson divided the data into five blocks based on similar set of features, performed feature selection with repeated elastic net technique (RENT) [21], and used different classification algorithms for predicting the presence of biomarkers of early AD. Olofsson's work can be categorized as a model-centric approach where data is kept the same and iterations are performed over the model and hyperparameters to improve performance. The objective of our work is the same as that of Olofsson, i.e., analyse patients' data to predict the risk factors and the presence of biomarkers of Alzheimer's disease. However, we adopt a data-centric approach, where the data blocks are iteratively improved (e.g., different pre-processing, feature transformations, resampling, embeddings etc.) with minimal changes in the machine learning model(s).

This thesis is divided into five chapters: introduction to Alzheimer's disease and a review of previous research in machine learning for early detection of the disease (Chapter 1); theoretical concepts used in our work (Chapter 2); the underlying data and the methods we use (Chapter 3); results and discussion (Chapter 4); and conclusions drawn from our work (Chapter 5).

Notes

[1]Figure reused from [3] under Creative Commons Attribution License.

[2]Amino acid chains linked together by peptide bonds.

[3]Cylindrical structures that are important architectural elements of a neuron.

[4]Image by ADEAR: "Alzheimer's Disease Education and Referral Center, a service of the National Institute on Aging." Source: `https://commons.wikimedia.org/wiki/File:TANGLES_HIGH.jpg`, Public domain, via Wikimedia Commons.

[5]`https://www.oasis-brains.org/`

[6]`https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images`

2

Theory

2.1 Model-centric vs. Data-centric AI

Leading experts in AI education and industry have come out as strong proponents of moving from the classic strategy of so-called model-centric AI towards data-centric AI [22], [23], [24]. Recently, there has been a lot of discussion around model-centric vs. data-centric approach to artificial intelligence [25], [26], [27], [28], [29]. Figure 2.1 is a simple schematic that highlights the difference between two approaches. The fundamental difference is that in model-centric approach, data is kept fixed and the machine learning model/code and its hyper-parameters are iteratively improved with the objective to achieve better model performance; while in data-centric approach, data is iteratively improved to enable more machine learning models to perform well. The need for data-centric approach stems from the requirement of building

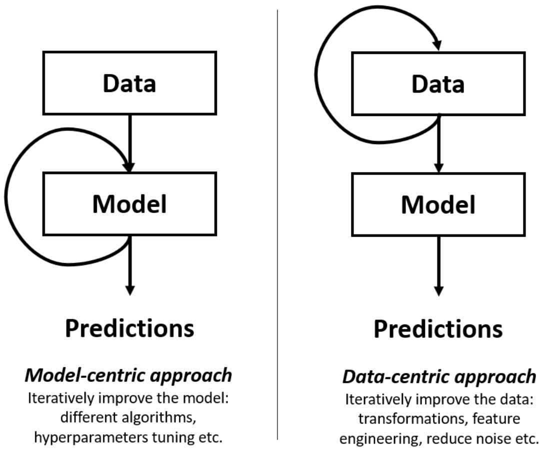

Data
Model
Predictions
Model-centric approach
Iteratively improve the model: different algorithms, hyperparameters tuning etc.
Data
Model
Predictions
Data-centric approach
Iteratively improve the data: transformations, feature engineering, reduce noise etc.

Figure 2.1: Model-centric and data-centric approach to artificial intelligence.

machine learning models that can be deployed in production environment. Although deployment of a production level machine learning pipeline is beyond the scope of our current work; we are attempting a data-centric approach to see whether adequate prediction performance can be obtained for our data with this approach. We try several methods to improve quality of the data while keeping the classification model almost fixed.

2.2 Data pre-processing

The pre-processing steps applied to the patients' data by Olofsson have been discussed at length in their work [20]. In this section, we briefly touch upon the steps that we apply to the data during our work.

2.2.1 Standardization

Many machine learning algorithms are known to benefit from scaling of the features, e.g., the gradient descent algorithm that minimizes the cost function in ML algorithms converges faster if the features are scaled. This can only be done to numerical features and one of the ways to scale the features for better convergence is *standardization*. Standardization brings the mean of each feature to 0, and standard deviation of each feature to 1. This is achieved using equation 2.1:

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \tag{2.1}$$

where for the feature j, $x_{i,j}$ and $x'_{i,j}$ are i^{th} original value and standardized value, respectively; and μ_j, σ_j are mean and standard deviation, respectively.

2.2.2 Encoding categorical features

Categorical features are the ones that have labels instead of numeric values, e.g., the feature *gender* in our data has two possible values: `male` and `female`. Such features do not have an order i.e., one value cannot be ranked higher or lower than the other; these are called *nominal* categories. If a categorical feature can be ordered e.g., a feature *size* can be small, medium, large etc.; it is called an *ordinal* category. In this thesis, we only have to pre-process nominal categories, so we restrict our discussion to such features. We use one-hot encoding to encode nominal categorical features. For example, *gender* observations shown in table 2.1 are one-hot encoded as shown in table 2.2. Notice that one-hot encoded features have numerical values (0 and 1), and they can now be standardized.

Table 2.1: Nominal categorical feature: *gender*

Observation	*gender*
1	`male`
2	`female`

Table 2.2: *gender* feature after one-hot encoding.

Observation	`male`	`female`
1	1	0
2	0	1

2.2.3 Shapiro-Wilk test

Many of the statistical analysis procedures e.g., correlation, analysis of variance etc. assume that the data follows a Gaussian/normal distribution. There are several statistical tests that can be done to check whether data is normally distributed. Visual assessment, e.g., Q-Q plots or P-P plots, and Shapiro-Wilk test are highly recommended to test normality of data [30]. Shapiro-Wilk test is a hypothesis test where the null hypothesis is that a subset of population (sample) belongs to a normal distribution. We use visual assessment and Shapiro-Wilk test to check for normality of all numerical features in our data.

2.2.4 Data transformation

Sometimes, the original features require some kind of transformation to change their distribution. It is often desirable to have normally distributed features for reliable statistical analyses. Depending on the original distribution of a feature, transformation is applied to the feature to bring it closer to a normal distribution. For the features in our data, we use two different transformations for this purpose.

Yeo-Johnson transformation:

Power transforms are data stabilization transformations that can remove the skewness from original data and make it more normally distributed. As mentioned earlier, normally distributed data can be advantageous in statistical analysis, and in building the ML models. Box-Cox transformation [31], introduced in 1964 by Box and Cox is perhaps the most widely used power transformation, but it is restricted in a way that it requires the data points to be strictly positive. In year 2000, Yeo and Johnson [32] developed a power transform that removed the restriction for the data to be strictly positive and we use this transformation for our data. Yeo-Johnson transformation is given by equation 2.2:

$$\psi(y,\lambda) = \begin{cases} \frac{(y+1)^\lambda - 1}{\lambda} & y \geq 0 \text{ and } \lambda \neq 0, \\ \log(y+1) & y \geq 0 \text{ and } \lambda = 0, \\ \frac{-((-y+1)^{2-\lambda} - 1)}{2-\lambda} & y < 0 \text{ and } \lambda \neq 2, \\ -\log(-y+1) & y < 0,\ \lambda = 0 \end{cases} \tag{2.2}$$

where ψ is the transformed value of y, and λ is the power parameter that needs to be estimated for approximation of normal distribution. If $\lambda = 1$, equation 2.2 gives identity transformation (no change and $\psi = y$).

Log transformation:

This is a straightforward transformation that changes the data to log of data. It removes skewness from data in cases where the data follows a log-normal distribution, therefore it is restricted in its usage. We use natural log transformation, and it is given by the equation 2.3 ($y^{'}$ is the transformed value of y):

$$y^{'} = \ln(y) \tag{2.3}$$

To demonstrate how these transformations change the data distribution, we generate random data that resembles a log-normal distribution (figure 2.2). The data distribution changes to a normal distribution after Yeo-Johnson transformation is applied to this data (figure 2.3). Because the example data has a log-normal distribution, it can also be changed to a normal distribution using log transformation (figure 2.4). As mentioned earlier, log transformation is restricted in its usage to normalize the data because it can only normalize data distribution if the original data has a log-normal distribution. However, it offers easy interpretability and even though Yeo-Johnson transformation also works for log-normal distributions, we prefer log transformation in cases where the data allows it i.e., if the data appears to have a log-normal distribution.

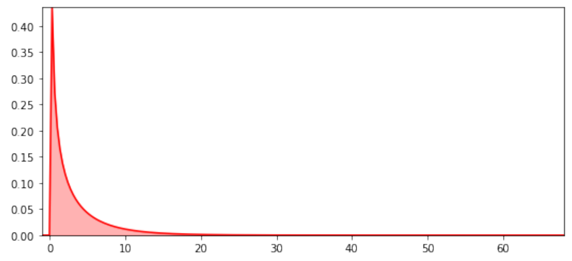


Figure 2.2: Example data, randomly generated, resembling a log-normal distribution.

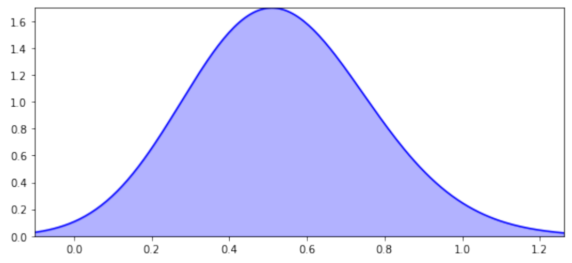


Figure 2.3: Data distribution after Yeo-Johnson transformation of example data.

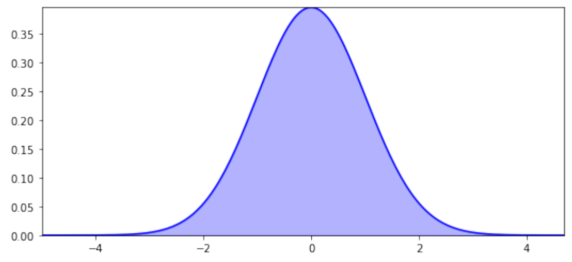


Figure 2.4: Data distribution after log transformation of example data.

2.3 Outlier detection and manifold learning

Data points that deviate from the data distribution are outliers. They can adversely impact any statistical analysis or machine learning model. Li et. al. [33] developed

Empirical Cumulative distribution-based Outlier Detection (ECOD), which is an unsupervised outlier detection method. ECOD uses high-density regions (inliers) and low-density regions (outliers) in the data to determine where data points are less likely to be present. It is dependent on setting a threshold which is the fraction of outliers expected in the data, hence knowledge about the data is important.

In addition to ECOD, we use manifold learning [34] to visualize our high-dimensional data in lower dimensions, and to help us in identifying potential outliers. We consider two manifold learning techniques: (i) t-distributed Stochastic Neighbour Embedding (t-SNE) [35] and (ii) Uniform Manifold Approximation and Projection (UMAP) [36]. We choose UMAP as our manifold learning algorithm. The rationale behind choosing UMAP over t-SNE is discussed more in section 4.1.6. Manifold learning methods are becoming increasingly popular [37] and they provide the ability to capture non-linear structure in data while performing dimensionality reduction. Hence, they can be a powerful tool to visualize the groups/clusters/patterns of data points and to identify outliers even if the original data resides in higher dimensions.

Manifold learning can be both supervised and unsupervised, but it is typically unsupervised and learns from high dimensional structure of the data to create a lower dimensional embedding. It can also be argued that these methods can give an idea about class separability of high-dimensional data in low-dimensional space. For this reason, we use manifold learning also as one of the data preparation steps.

2.4 Feature selection

Feature selection aims to select a subset of features from the dataset that performs better or at least as good as the whole set of features. From methodological point of view, feature selection has three types: filter approach, wrapper approach, and embedded approach. In filter approach, a pre-defined importance criterion is used to rank the features. In wrapper approach, machine learning models are trained on candidate subsets of features and the subset that offers best performance is selected. In embedded approach, feature selection is integrated into the machine learning algorithm. Jenul et. al. [20] provide a concise summary of different feature selection approaches.

We use Exhaustive Feature Selection (EFS) from `mlxtend`[1] library for our data. EFS belongs to the wrapper approach of feature selection and a supervised learning model is trained for every possible subset of features from the dataset. Combined with cross-validation, it somewhat guarantees that the best performing subset is selected. However, EFS can be computationally expensive if the number of features is high. This is not a critical concern after we divide our data into blocks (details in Chapter 3) and hence, we choose EFS.

2.5 Synthetic Minority Over-sampling Technique

The data being used in this thesis is imbalanced and one of the ways to tackle this is through resampling, i.e., by either under-sampling the majority class, over-sampling the minority class, or a combination of both. We address the class imbalance in our data by using Synthetic Minority Over-sampling Technique (SMOTE) [38]. This is a data augmentation technique and is used to oversample the minority class by generating synthetic samples of data. We use a variant of this technique called Borderline SMOTE [39] and augment the data in each block with synthetic data points to balance the classes.

In Borderline-SMOTE, a classification model is trained, and focus is placed on samples of minority class that are misclassified i.e., samples that are near the *borderline* between two classes. Synthetic samples are then generated that are similar to the misclassified samples, thereby increasing the population of minority class in the region where it is required.

It is reported that feature selection before oversampling using SMOTE offers better results [40]. Therefore, we also follow this approach and apply SMOTE after feature selection in our data iterations.

2.6 Dynamic ensemble selection

Different classification algorithms produce different errors on different data samples. Hence, combining different classifiers and creating an ensemble to be used on different subsets of data can offer better prediction performance [41], [42]. Bagging and Boosting [43], [44] are such techniques that use ensemble learning for classification. In addition, multiple classifier systems have also been explored at length to improve classification accuracy [45]. Dynamic ensemble selection (DES) is also an ensemble learning technique that uses multiple classifiers. In DES, multiple ML models are trained on training data and a dynamic selection is made from ensemble members to make predictions on new data. When new data comes in, its similarity with data points in the training set is determined using KNN; the classification model that performs best on nearest neighbours of the new data point is chosen to make predictions [46]. `DESlib`[2] library provides Python implementation of DES.

2.7 Performance metric

Considerable amount of literature is available on different performance metrics and what they mean. Hence, we choose not to describe them all. Like the previous work [20], we use Matthews correlation coefficient (MCC) as our primary performance metric and briefly describe it here. In addition to MCC, we report five other performance metrics as well, to get a more detailed impression of model performance. These five metrics are: (i) accuracy, (ii) area under the receiver operating characteristic curve (ROC AUC), (iii) F1 score, (iv) precision, and (v) recall.

Matthews correlation coefficient

Matthews correlation coefficient (MCC) is widely regarded as a reliable performance metric [47], even more so in case of imbalanced classes. MCC is given by equation 2.4, where TP, FP, TN, FN mean true positive, false positive, true negative, false negative, respectively. Value of MCC ranges from -1 to +1, where +1 is perfect classification and MCC of 0 is considered as good as random guessing.

$$\mathrm{MCC} = \frac{(\mathrm{TP} \cdot \mathrm{TN}) - (\mathrm{FP} \cdot \mathrm{FN})}{\sqrt{(\mathrm{TP} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TN} + \mathrm{FN})}} \tag{2.4}$$

2.8 Model validation

Validation of model performance is the key to building robust machine learning models that can generalize and perform well on unseen data. There are several options for the choice of model validation protocol and they mostly depend on the amount of available data. If sufficient data is available, the most straightforward *hold-out validation* is done in which data is divided into training and test sets e.g., in 70:30 (train:test) proportion. Model is trained on the training set, validated on the test set, and then hyper-parameters are tuned for best performance. If there is less data available, *K-fold cross validation* is a reasonable choice. The data is divided into equal chunks (K number of chunks), and one chunk is chosen as validation set while model is trained on the remaining data. The training/validation step is then repeated by selecting a different chunk of data as the validation set. The process continues until all chunks of data are used as validation sets; this is illustrated in figure 2.5. Because multiple models are trained during K-fold cross validation, the reported performance metric is usually the average performance across all models.

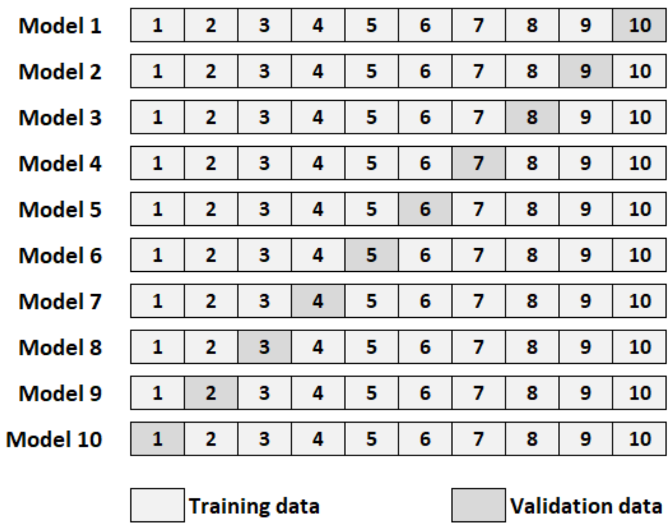


Figure 2.5: Example of *K-fold* cross validation, with K=10. The data is divided into ten chunks represented by numbers 1-10. Each chunk represents 10% of available data.

An extension of K-fold is iterated K-fold or repeated K-fold. In this protocol, repetitions of K-fold cross validation are done and the data is shuffled in each repetition so the chunks of data are not the same across repetitions. A total of $K \times m$ models are trained in repeated K-fold cross validation, where m represents the number of repetitions/iterations. We use repeated K-fold cross valiation for evaluating and validating the model performance in our work.

Notes

[1] `http://rasbt.github.io/mlxtend/api_subpackages/mlxtend.feature_selection/`

[2] `https://github.com/scikit-learn-contrib/DESlib`

3

Materials and Methods

3.1 Data blocks

In previous work [20], the data was divided into five blocks by grouping together the similar features. We use the same data blocks:

Block A: Patient group

Block B: Physical health and family history

Block C: Results of cognitive tests

Block D: Lesion and white-matter hypersensitivity

Block E: Results from MRI of subcortical brain structures

This is a binary classification (0 or 1) problem, and the objective is to predict the presence or absence of biomarkers of Alzheimer's disease. For the target column, 1 means that there is evidence of the presence of biomarkers of AD in a patient, and 0 means otherwise.

Data in Block A only represents the subject group to which patients belong e.g., cognitive symptoms control group, family history control group etc. Block A does not provide any other information about risk factors or health history. This is further confirmed by the results achieved previously [20], where performance of the models trained on features of Block A was poor. Hence, we exclude Block A from our analysis.

Furthermore, we perform basic data cleaning and feature engineering slightly differently than how it was done previously [20]. Hence, we end up with different size of data for Block B and the test set. This is explained further in Chapter 4.

3.2 Computational resource

We perform all of the computations on Windows 10 Enterprise 20H2 (OS build 19042.1826) running on 11th Gen Intel® Core$^{\text{TM}}$ i7-1165G7 CPU @ 2.80 GHz with

15.7 GB of usable random-access memory. Storage media is a solid-state drive.

We use Jupyter Notebook as Python 3.9 IDE for coding work. In addition to the most used Python libraries for data pre-processing, visualization, and machine learning such as `NumPy`, `pandas`, `matplotlib`, `seaborn`, `scikit-learn`; we also use additional Python libraries that provide implementations of methods described in Chapter 2. These libraries and their brief descriptions are provided in table 3.1. We use GitHub Desktop version 2.9.12 (x64) as Git client to manage the repository (M30-DV Master Thesis) on GitLab.

Table 3.1: Python libraries used in this thesis.

Python library	Brief description
`SciPy`	Scientific computing including statistical functions
`hoggorm`	Explorative multivariate statistics
`pyOD`	Outlier/anomaly detection in multivariate data
`sklearn.manifold`	Manifold learning/non-linear dimensionality reduction
`mlxtend`	Useful library for common data science tasks
`imbalanced-learn`	Tools for dealing with imbalanced classes
`DESlib`	Modules for dynamic ensemble selection algorithms

3.3 Data-centric approach

As discussed in Chapter 2, we follow a data-centric approach and the data preparation steps are applied iteratively while the machine learning model is kept fixed. Following each iteration of data preparation, we use dynamic ensemble selection which serves as our classification model. The performance of ML models for all the data iteration are compared to determine which combination of data preparation steps delivers the best performance.

3.3.1 Data preparation iterations

We plan a set of 17 iterations on the data (1 baseline + 16 different combinations of data preparation steps) and feed the data to ML model after each iteration. The data preparation steps and their purpose is briefly described in table 3.2; we also assign IDs to these steps for easy reference. The underlying theory behind these steps has been explained in Chapter 2. Steps and their order of application that constitutes each data iteration is given in table 3.3. We apply these 17 data iterations one-by-one, on each of the four data blocks separately, and a separate machine learning model (discussed in the next section) is trained in each case. Note that the data iterations DI-10 to DI-16 do not have step S2, so these iterations can be regarded as the ones that do not use data transformation to change the distribution of data. On the contrary, data iterations DI-02 to DI-09 involve the step S2 and so the data distribution is modified in these iterations.

Table 3.2: Data preparation steps to be used in data iterations (S0 is described briefly in figure 3.1).

ID	Data preparation	Purpose
S0	Data cleaning	Data blocks with only basic cleaning
S1	Standardization	Bring data on the same scale
S2	Power/Log-transform	Change data to normal distribution
S3	Exhaustive feature selection	Select subset of features that work best
S4	Resampling using SMOTE	Fix class imbalance
S5	Manifold learning	Separate classes in low-dim embedding

3.3.2 ML Model: Dynamic Ensemble Selection

We now discuss the machine learning model that we use after each data iteration to make predictions. We are using dynamic ensemble selection (DES). The algorithm we use for DES is k-Nearest Neighbour Oracle (KNORA) whose implementation is provided in `DESlib` library. Different classification algorithms can be selected to create a pool of classifiers that serve as the ensemble to be used by KNORA. We use a variant of KNORA called KNORA-Eliminate, or KNORA-E for short. For a data point to be classified, KNORA-E selects all classifiers in the ensemble that give perfect predictions on the nearest neighbours of that given data point. In case no ensemble member achieves perfect accuracy, the process is repeated with reduced size of the neighbourhood until model(s) with perfect performance are selected and are used to make predictions on the data point.

Pool of classifiers for KNORA

By default, KNORA implementation in `DESlib` uses bagging (bootstrap aggregation) ensemble decision trees as the pool of classifiers. Remember that we are following a data-centric approach in which model code is kept fixed. Therefore, we use the default implementation of KNORA as our classification model.

3.4 Workflow

The 17 data iterations listed in table 3.3 can be divided into two categories: DI-02 to DI-09 are with transformation of data distribution (step S2), and DI-00, DI-01, and DI-10 to DI-16 are without transformation of data distribution. For sake of clarity, we visualize the workflow of both categories of data iterations.

Figure 3.1 shows the schematic of data iterations that include step S2. Figure 3.2 shows the schematic of data iterations that do not have step S2. Note that DI-00 and DI-01 are pre-requisites for DI-02, hence these iterations are shown in both workflows. Both workflows are repeated for four blocks of data (Block B, Block C, Block D, and Block E), resulting in separate models for every combination of data iteration and data block. The results are then compared to determine which of these models perform the best.

Table 3.3: Data preparation iterations for data-centric approach in this thesis. The ML model (dynamic ensemble selection) is trained after each iteration, for each of the four data blocks.

Data Iteration ID	Steps and their order
DI-00	S0 (to be used as baseline)
DI-01	S0 → S1
DI-02	S0 → S1 → S2
DI-03	S0 → S1 → S2 → S3
DI-04	S0 → S1 → S2 → S3 → S4
DI-05	S0 → S1 → S2 → S3 → S4 → S5
DI-06	S0 → S1 → S2 → S3 → S5
DI-07	S0 → S1 → S2 → S4
DI-08	S0 → S1 → S2 → S4 → S5
DI-09	S0 → S1 → S2 → S5
DI-10	S0 → S1 → S3
DI-11	S0 → S1 → S3 → S4
DI-12	S0 → S1 → S3 → S4 → S5
DI-13	S0 → S1 → S3 → S5
DI-14	S0 → S1 → S4
DI-15	S0 → S1 → S4 → S5
DI-16	S0 → S1 → S5

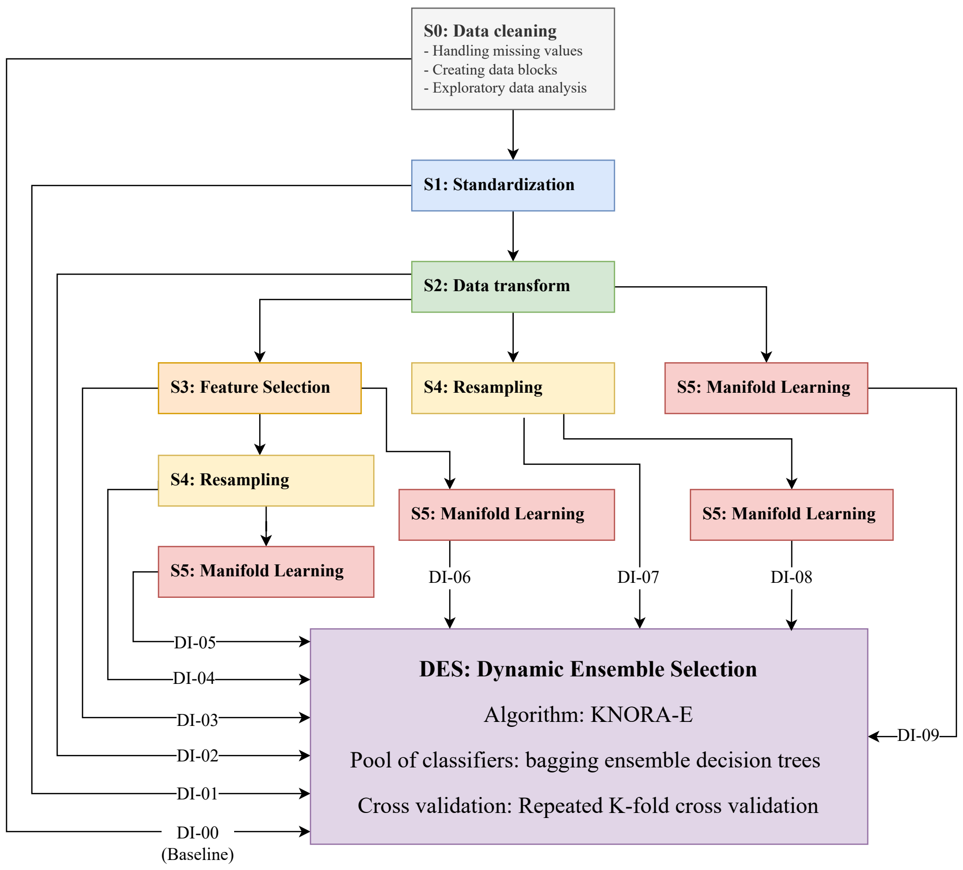


Figure 3.1: Schematic of data-centric approach to our work: data iterations with data transformation step S2.

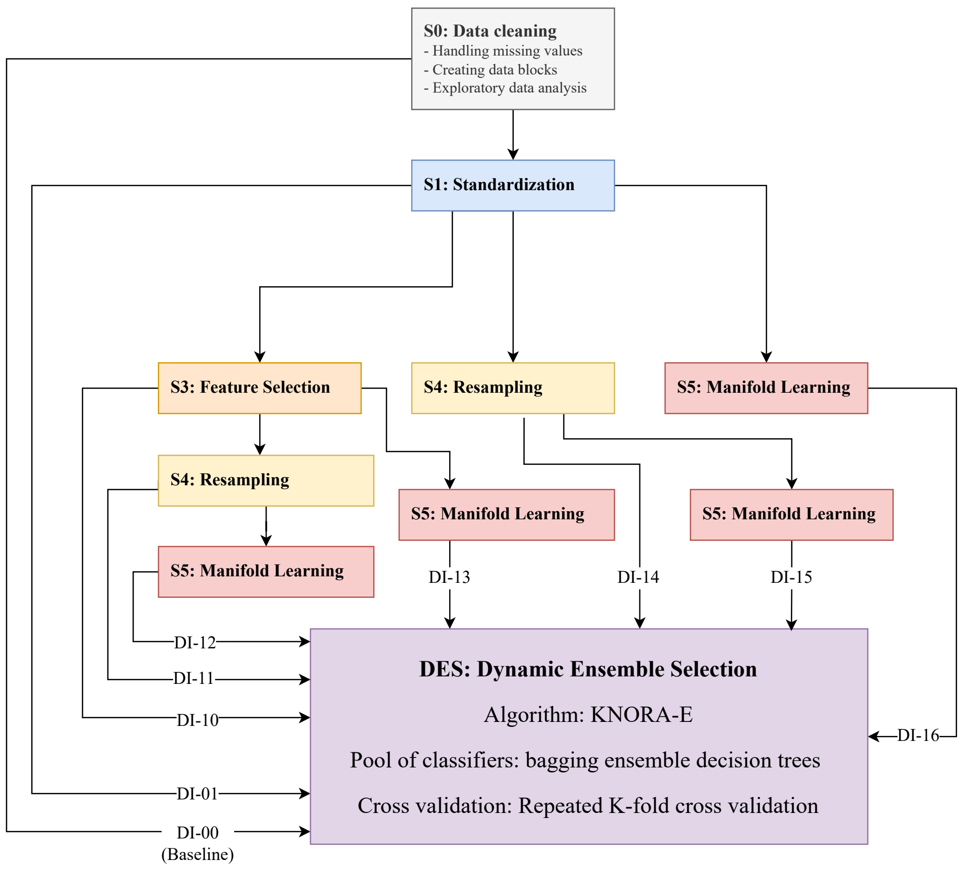


Figure 3.2: Schematic of data-centric approach to our work: data iterations without data transformation.

4

Results and Discussion

4.1 Data preparation

4.1.1 Data cleaning and standardization

The raw data has information about 789 patients who were investigated for Alzheimer's disease. There are 1505 observations and 1733 columns including the target column. 480 observations have missing target, so we drop those and are left with 1025 observations of 660 unique patients. Taking cues from the work done previously [20], data is divided into five blocks (see Chapter 3) based on the available patient information.

Basic exploration and cleaning

Data blocks are created mostly in the same way as in the previous thesis [20]. However, Block B is processed differently. More specifically, the indicator for cardiovascular disease (`cvd_risk`) in Block B was created as the sum of all those features that could be potential risk factors (all categorical, 0 or 1). If there's even a single risk factor present, it was treated as the patient in susceptible to heart disease and the `cvd_risk` for that patient is assigned a value of 1 (or 0 in the absence of no risk factors). However, in some cases, the data for most or all of risk factors was simply missing. When summation operation is applied over the dataframe to create the `cvd_risk`, it was assigned a value of 0 even in cases where risk factors had all `NaN`. This is clearly wrong, so we fix this and `cvd_risk` is assigned `NaN` where data about risk factors was missing. Furthermore, information about `headtrauma` (traumatic brain injury) and `cns_infec` (infection of the central nervous system) is also included in Block B. Finally, we have slightly less data in Block B than what was reported by Olofsson [20]. We also perform one-hot encoding of the nominal categories in our data. The data points in which information about every feature in all the blocks was available is kept as the test set. Table 4.1 shows the shape of each data block after basic pre-processing steps. Note that we exclude Block A from our analysis because of lack of information in the data. List of features and their types in each data block are given tables A.1 – A.4 (Appendix A1).

Table 4.1: Size of the data blocks (no. of rows, no. of features) after basic exploration and cleaning. Sum of the number of features across all blocks equals the number of features in the test set.

Data block	Size
Block B	(254, 17)
Block C	(753, 7)
Block D	(121, 6)
Block E	(439, 10)
Test set	(164, 40)

Exploratory data analysis

A very comprehensive exploratory data analysis (EDA) of this data has already been done previously [20]. This exploration includes analysis and visualization of missing values, correlation coefficients within blocks and between data blocks, and principal component analysis. Instead of re-inventing the wheel, we focus our attention regarding EDA on: (i) checking the distribution of data to determine whether to apply transformations, (ii) checking for correlated features, and (iii) exploring the possibility of linear dimensionality reduction through PCA.

Test of normality of data Instead of relying on the appearance (histogram, KDE plot etc.), we perform the Shapiro-Wilk test for normality on all the non-categorical features in our data to see whether they are normally distributed and to decide whether to apply transformations or not. According to the Shapiro-Wilk test performed using the `SciPy` library in Python, there are a total of 21 numerical features in all the blocks that do not have a normal distribution. Examples of two such features from each data block are visualized using normal probability plots in figures A.1–A.8 (Appendix A2).

Correlated features There are features in the data that appear to be correlated. Some features in some blocks have very high correlation among each other, while some blocks have only some of the features that are correlated. For example, features in Block C do not seem to correlate much, but features in Block D have high correlation coefficients. We check for correlation among the features to see whether dimensionality reduction or feature selection is warranted. Looking at the heatmaps of Pearson's correlation coefficients for the numerical features, and phi coefficients for one-hot encoded features in figures A.9–A.13 (Appendix A3), we conclude that we must explore dimensionality reduction and feature selection to address the issue of correlation among the features in our data.

PCA We did PCA on all data blocks and observe that if we use the principal components, we do not get a lot of benefit in terms of dimensionality reduction. Because to capture enough variation in data, the number of principal components required are not significantly lower than the number of original features. For example, with Block B which has 17 features, we need 14 principal components to get validated cumulative explained variance of over 90%. Figures A.14–A.17 (Appendix A4) show the number of principal components vs. variance explained for each data block. And table 4.2 gives the number of principal components required in each block to get validated cumulative explained variance of over 90%.

Table 4.2: Number of principal components required to capture more than 90% validated cumulative explained variance.

Data block	Explained variance (%)	No. of PCs required	No. of features
Block B	93.6	14	17
Block C	92.4	6	7
Block D	95.5	4	6
Block E	92.1	7	10

Therefore, we conclude that PCA does not offer much in terms of dimensionality reduction for our data. We also stand to lose the direct connection between original features and the target after features are transformed in PCA. Hence, to address the issue of correlated features, we look into feature selection in later part of our work. We also explore manifold learning which is considered to be non-linear dimensionality reduction. The data cleaned until this step is denoted as S0, and it is used in the first data iteration (DI-00). Results from the classification model using this data serve as the baseline for the rest of our work.

Standardization

Following data cleaning and exploration, we perform the simple step of standardizing the data as explained in Chapter 2.

4.1.2 Data transformation

Out of total 27 numerical features (across all the data blocks), there are 21 features that are not normally distributed (checked using Shapiro-Wilk test). Although some features fail the normality test, their normal probability plots (Q–Q plots against normal distribution) show that their distribution is reasonably close to Gaussian/normal distribution. We use Yeo-Johnson transformation on some features and some features go through log transformation. As mentioned in Chapter 2, we prefer log transformation due to its easy interpretability, but it only works for data that has a log-normal distribution. So we apply log transformation where we can and apply Yeo-Johnson transformation in other cases. After trying different options of transforming the data, we settle with the following:

Features with normal distribution:

`cowat_tscore, ANT_HPC, POST_HPC, ERC, Br36, Meninges`

Feature close to normal distribution/no transformation applied:

`age, edu_years, edu_level, mmse_total, clock_score,`

`vosp_tscore, cerad_recall, MISC, Br35`

Features that are power transformed (Yeo-Johnson transformation):

`systolic_bp, PSMD, Meninges_PHC, PHC, ColSul`

Features that are log **transformed:**

`tmta_sec, tmtb_sec, LesF, LesO, LesP, LesT, WMHo_rV`

4.1.3 Visualizing high-dimensional data and outliers

To visualize high-dimensional data, we decide to use a manifold learning algorithm: Uniform Manifold Approximation and Projection (UMAP). We also explored different outlier detection algorithms and use ECOD class from pyOD, which uses empirical cumulative distribution function for unsupervised outlier detection. We look at the plots from manifold learning and results from outlier detection to decide how to identify outliers and what to do with them. Visual inspection of UMAP projections for each data block do not show any data points that are away from rest of the data. Using an outlier detection algorithm where we manually set a threshold can provide different results and they can be different from our interpretation of the UMAP projections. For example, there are 439 points in block E and setting a 1% fraction (called contamination in pyOD implementation of ECOD) identifies 5 data points as outliers. Screenshots of ECOD results are shown in figures A.18–A.21 (Appendix A5) and plots obtained from UMAP using two components are shown in A.22–A.25 (Appendix A6).

ECOD is highly dependent on setting a manual threshold, so we perform a qualitative assessment and rely on the visual inspection of UMAP projections for identifying clusters of data points and potential outliers. Looking at the plots, it appears that there are no points that are away from rest of the data, even if we tune the parameters of UMAP[1]. Hence, we decide not to drop any data points or treat any of them as outliers.

Note: In this section, we have only talked about using manifold learning for visualizing high-dimensional data, and for identifying outliers. We are not yet using manifold learning to transform our data for training the classification model. Later (in section 4.1.6), we discuss using UMAP to create low-dimensional embedding of data in some of the data iterations in our work.

4.1.4 Feature selection

We use brute-force evaluation of feature subsets using exhaustive feature selector (EFS) with 5-fold cross validation from `mlxtend` library. We use k-nearest neighbours (KNN) classification with EFS to evaluate the best performing subset. The results of EFS (with k=5 in KNN) including the best performing feature subsets and the corresponding MCC scores are given in tables 4.3 (feature selection after step S2) and 4.4 (feature selection without prior step S2).

Table 4.3: Best performing subset of features determined by EFS (with step S2).

Data block	Selected features after EFS	MCC
Block B	`age, edu_years, cvd_risk, APOE_E2/E3, APOE_E3/E3, cns_infec`	0.598
Block C	`mmse_total, clock_score, cerad_recall`	0.351
Block D	`LesF, LesP, WMHo_rV`	0.420
Block E	`POST_HPC, MISC, ERC, PHC`	0.398

Table 4.4: Best performing subset of features determined by EFS (without step S2).

Data block	Selected features after EFS	MCC
Block B	`age, edu_years, cvd_risk, APOE_E2/E3, APOE_E3/E3, cns_infec`	0.598
Block C	`mmse_total, clock_score, tmta_sec, cerad_recall`	0.367
Block D	`LesF, LesP, LesT, WMHo_rV`	0.367
Block E	`POST_HPC, MISC, ERC, PHC`	0.399

The MCC scores achieved after feature selection are comparable to that of previous work [20] which used RENT for feature selection. For ready reference, the MCC scores reported previously [20] after using RENT are 0.5519 (Block B), 0.4048 (Block C), 0.3442 (Block D), and 0.3712 (Block E).

4.1.5 Handling class imbalance

Our data has imbalanced class distribution, and we choose to balance the classes by resampling the data. Class distribution in each block is given in table 4.5. We use a variant of Synthetic Minority Over-sampling Technique (SMOTE) called Borderline SMOTE to oversample the minority class (class 1) by generating synthetic samples of data. Class distribution after resampling using Borderline SMOTE is given in table 4.6.

Table 4.5: Class distribution in each data block.

Data block	Class distribution (**0:1**)	Class distribution in % (**0:1**)
Block B	186:68	73::27
Block C	515:238	68:32
Block D	90:31	74:26
Block E	299:140	68:32
Test set	117:47	71:29

Table 4.6: Class distribution in each data block after applying Borderline SMOTE; note that test data is not resampled.

Data block	Class distribution (**0:1**)	Class distribution in % (**0:1**)
Block B	186:186	50:50
Block C	515:515	50:50
Block D	90:90	50:50
Block E	299:299	50:50
Test set	117:47	71:29

4.1.6 Manifold learning for dimensionality reduction

Earlier in this chapter, we discussed manifold learning for visualizing high dimensional data and outliers. However, the application of manifold learning methods extend beyond just data visualization. They can be used as an indicator of the separability of classes in lower dimensions. We explore two different manifold learning methods, viz. t-distributed Stochastic Neighbour Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP). The superiority of UMAP over t-SNE has been strongly argued by Oskolkov [48]. Some of those arguments are that UMAP can embed in more than 3 dimensions while t-SNE is restricted to maximum of 3 dimensions; this may not be a problem from visualization point of view, but it is a limitation if manifold learning is used for dimensionality reduction. Also, t-SNE does not preserve global structures, i.e., t-SNE can identify different clusters but its results do not give much information about how different the clusters are from one another, e.g., if two clusters of data points in two-dimensional t-SNE projection are far from each other, it does not guarantee that these clusters are very different. t-SNE also requires too much memory and scales poorly as sample size increases. Furthermore, documentation of the Python library `umap` points to interesting scientific papers [49] that successfully used UMAP in a variety of subjects for visualization of high-dimensional data and dimensionality reduction. Hence, we perform of manifold learning using the library `umap` which provides Python implementation of the UMAP algorithm. In essence, manifold learning is an unsupervised approach and we use it to produce a low-dimensional embedding of our data for use in the classification model.

4.2 Classification model and performance

We use different combinations of data preparation steps for the data iterations (table 3.3) and then use dynamic ensemble selection (DES) model for classification (figures 3.1 and 3.2). This is repeated for four blocks of data viz. Block B, Block C, Block D, and Block E. During model training, we implement repeated stratified K-fold cross validation (5-folds with 2 repetitions) in DES for model validation. Models are also tested on test data to evaluate predictive performance. The cross validated performance and predictive performance of all the models are reported using six performance metrics including Matthews correlation coefficient (MCC), accuracy, area under the ROC curve (ROC AUC), F1 Score, precision, and recall.

The performance metrics achieved during cross validation and on test data for all data iterations and for all data blocks are given in tables A.5–A.10 in Appendix A7. It is obvious that most of the models are overfitting by a large margin (predictive performance metrics are lower than performance metrics on training data), with some classifiers performing even worse than random guessing (MCC lower than zero on test data). Table 4.7 summarizes the data iterations that give the highest metrics for each data block for both training and test data. The table shows that for training data, best scores (after repeated k-fold cross validation) are achieved with data iterations that include data transformation (S2), feature selection (S3), and resampling (S4) steps. However, with regard to the performance on test data, it is unclear which combination of pre-processing steps would give the highest score for every data block; it seems that best performance is achieved by different pre-processing for different data. Hence, we cannot conclude with certainty that a given sequence of data pre-processing gives the best performance. We believe that further research on more datasets is needed to establish more knowledge about what is best in each situation/data block.

Table 4.7: Data iterations that give the best performance metrics for each data block with training data (with repeated k-fold cross validation) and test data. Corresponding performance metric recorded is given in parentheses.

Metric	Block B		Block C		Block D		Block E	
	Train	*Test*	*Train*	*Test*	*Train*	*Test*	*Train*	*Test*
MCC	DI-07 (0.656)	DI-12 (0.350)	DI-07 (0.502)	DI-15 (0.255)	DI-04 (0.555)	DI-03 (0.227)	DI-07 (0.551)	DI-00 (0.343)
Accuracy	DI-07 (0.827)	DI-16 (0.744)	DI-07 (0.751)	DI-03 (0.732)	DI-03 (0.773)	DI-03 (0.732)	DI-07 (0.774)	DI-00 (0.744)
ROC AUC	DI-07 (0.853)	DI-12 (0.689)	DI-07 (0.777)	DI-15 (0.625)	DI-04 (0.778)	DI-14 (0.585)	DI-07 (0.785)	DI-00 (0.661)
F1 score	DI-05 (0.826)	DI-12 (0.559)	DI-07 (0.752)	DI-15 (0.462)	DI-04 (0.787)	DI-04 (0.404)	DI-07 (0.776)	DI-00 (0.512)
Precision	DI-07 (0.836)	DI-16 (0.581)	DI-07 (0.751)	DI-03 (0.565)	DI-04 (0.760)	DI-06 (0.600)	DI-07 (0.773)	DI-00 (0.564)
Recall	DI-05 (0.839)	DI-12 (0.660)	DI-07 (0.753)	DI-05 (0.511)	DI-04 (0.828)	DI-08 (0.426)	DI-07 (0.783)	DI-14 (0.553)

Notes

[1]`https://umap-learn.readthedocs.io/en/latest/parameters.html`

5 Conclusion

In this thesis, we adopted a data-centric approach for building a classification model that could predict the presence of biomarkers of Alzheimer's disease. We started with four data blocks that were created by grouping together the similar set of features such as health history, results of cognitive tests, and results of medical imaging. We iterated over different combinations of data pre-processing steps for each of the four data blocks and trained dynamic ensemble selection with repeated k-fold cross validation on each data iteration. We find that for training data, best performing data iterations generally have three steps in common for all data blocks: (i) transformation of data into a normal distribution, (ii) feature selection, (iii) oversampling of minority class. However, looking at the predictive performance, we see that the models are highly data dependent and it is not clear which is the best sequence of data pre-processing that would work for all the data blocks. We also find that most of the models we trained in this study were overfitting and had significantly lower performance on test set than on the training set. Therefore, we cannot claim to have conclusive proof that a given set of data pre-processing steps gives the best performance in predicting the presence of biomarkers of Alzheimer's disease. More research with more data, and perhaps with more combinations of data pre-processing steps should be explored to further develop knowledge about the approach that can provide satisfactory and reliable performance.

Bibliography

[1] World Health Organization (WHO) - Dementia Fact Sheet updated on 2nd September 2021 https://www.who.int/news-room/fact-sheets/detail/dementia retrieved on 27 February 2022

[2] Eratne, D., Loi, S. M., Farrand, S., Kelso, W., Velakoulis, D., and Looi, J. C. (2018). Alzheimer's disease: clinical update on epidemiology, pathophysiology and diagnosis. Australasian Psychiatry, 26(4), 347-357

[3] Breijyeh, Z., and Karaman, R. (2020). Comprehensive review on Alzheimer's disease: Causes and treatment. Molecules, 25(24), 5789.

[4] Armstrong, R. A. (2019). Risk factors for Alzheimer's disease. Folia neuropathologica, 57(2), 87-105.

[5] Knopman, D.S., Amieva, H., Petersen, R.C., Chételat, G., Holtzman, D.M., Hyman, B.T., Nixon, R.A. and Jones, D.T., (2021). Alzheimer disease. Nature reviews Disease primers, 7(1), pp.1-21.

[6] Chen, G. F., Xu, T. H., Yan, Y., Zhou, Y. R., Jiang, Y., Melcher, K., and Xu, H. E. (2017). Amyloid beta: structure, biology and structure-based therapeutic development. Acta Pharmacologica Sinica, 38(9), 1205-1235.

[7] Metaxas, A., and Kempf, S. J. (2016). Neurofibrillary tangles in Alzheimer's disease: elucidation of the molecular mechanism by immunohistochemistry and tau protein phospho-proteomics. Neural regeneration research, 11(10), 1579.

[8] Jatoi, S., Hafeez, A., Riaz, S. U., Ali, A., Ghauri, M. I., and Zehra, M. (2020). Low Vitamin B12 levels: An underestimated cause of minimal cognitive impairment and dementia. Cureus, 12(2).

[9] Dubois, B., Hampel, H., Feldman, H.H., Scheltens, P., Aisen, P., Andrieu, S., Bakardjian, H., Benali, H., Bertram, L., Blennow, K. and Broich, K. (2016). Preclinical Alzheimer's disease: definition, natural history, and diagnostic criteria. Alzheimer's and Dementia, 12(3), pp.292-323.

[10] Gonneaud, J., Baria, A.T., Pichet Binette, A., Gordon, B.A., Chhatwal, J.P., Cruchaga, C., Jucker, M., Levin, J., Salloway, S., Farlow, M. and Gauthier, S.

(2021). Accelerated functional brain aging in pre-clinical familial Alzheimer's disease. Nature communications, 12(1), pp.1-17.

[11] King-Robson, J., Wilson, H., Politis, M., and Alzheimer's Disease Neuroimaging Initiative. (2021). Associations between Amyloid and Tau pathology, and Connectome Alterations, in Alzheimer's Disease and Mild Cognitive Impairment. Journal of Alzheimer's Disease, 82(2), 541-560.

[12] Sharma, S., and Mandal, P. K. (2022). A comprehensive report on Machine Learning-based early detection of Alzheimer's disease using multi-modal neuroimaging Data. ACM Computing Surveys (CSUR), 55(2), 1-44.

[13] Mezrar, S., and Bendella, F. (2022). Machine learning and Serious Game for the Early Diagnosis of Alzheimer's Disease. Simulation and Gaming, 53(4), 369–387.

[14] Ghazal, T. M., Abbas, S., Munir, S., Khan, M. A., Ahmad, M., Issa, G. F., Zahra, S. B., Khan, M. A., Hasan, M. K. (2022). Alzheimer disease detection empowered with transfer learning. Computers, Materials and Continua, 70(3), pp. 5005–5019.

[15] Pirrone, D., Weitschek, E., Di Paolo, P., De Salvo, S., and De Cola, M. C. (2022). EEG Signal Processing and Supervised Machine Learning to Early Diagnose Alzheimer's Disease. Applied Sciences, 12(11), 5413.

[16] Mofrad, S. A., Lundervold, A., Lundervold, A. S., and Alzheimer's Disease Neuroimaging Initiative. (2021). A predictive framework based on brain volume trajectories enabling early detection of Alzheimer's disease. Computerized Medical Imaging and Graphics, 90, 101910.

[17] Revathi, A., Kaladevi, R., Ramana, K., Jhaveri, R. H., Rudra Kumar, M., and Sankara Prasanna Kumar, M. (2022). Early detection of cognitive decline using machine learning algorithm and cognitive ability test. Security and Communication Networks, 2022.

[18] Mohammed, B. A., Senan, E. M., Rassem, T. H., Makbol, N. M., Alanazi, A. A., Al-Mekhlafi, Z. G., Almurayziq, T. S., and Ghaleb, F. A. (2021). Multi-method analysis of medical records and MRI images for early diagnosis of dementia and Alzheimer's disease based on deep learning and hybrid methods. Electronics, 10(22), 2860.

[19] Mirzaei, G., and Adeli, H. (2022). Machine learning techniques for diagnosis of alzheimer disease, mild cognitive disorder, and other types of dementia. Biomedical Signal Processing and Control, 72, 103293.

[20] Olofsson, C. (2021). Using Machine Learning and Repeated Elastic Net Technique for Identification of Biomarkers of Early Alzheimer's Disease.

[21] Jenul, A., Schrunner, S., Huynh, B. N., and Tomic, O. (2021). RENT: A Python Package for Repeated Elastic Net Feature Selection. Journal of Open Source Software, 6(63), 3323, https://doi.org/10.21105/joss.03323

[22] Ng, A. (2021, Mar 24) MLOPs: From Model-centric to Data-centric AI. DeepLearningAI https://www.deeplearning.ai/wp-content/uploads/2021/06/MLOps-From-Model-centric-to-Data-centric-AI.pdf

[23] Kurlansik, R. and ML SMEs at Databricks (2021, Jun 23). Need for Data-centric ML Platforms. Databricks. https://databricks.com/blog/2021/06/23/need-for-data-centric-ml-platforms.html

[24] Patel, H. (updated on 2022, Jul 22) Data-Centric Approach vs Model-Centric Approach in Machine Learning. Neptune.ai Blog. https://neptune.ai/blog/data-centric-vs-model-centric-machine-learning

[25] Clemente, F. (2021, Mar 29) From model-centric to data-centric. Towards Data Science. https://towardsdatascience.com/from-model-centric-to-data-centric-4beb8ef50475

[26] Muaz, U. (2021, May 9) From Model-centric to Data-centric Artificial Intelligence. Towards Data Science. https://towardsdatascience.com/from-model-centric-to-data-centric-artificial-intelligence-77e423f3f593

[27] Singh, A. (2021, Jul 22) Moving from Model-centric to Data-centric approach. Medium. https://medium.com/analytics-vidhya/moving-from-model-centric-to-data-centric-approach-1468fb5dbafb

[28] Radečić, D. (2021, Aug 13). Data-centric vs. Model-centric AI? The Answer is Clear. Towards Data Science. https://towardsdatascience.com/data-centric-vs-model-centric-ai-the-answer-is-clear-4b607c58af67

[29] Nouri, S. (2021, Sep 30). Data-centric approach vs model-centric approach. LinkedIn. https://www.linkedin.com/pulse/data-centric-approach-vs-model-centric-steve-nouri/

[30] Ghasemi, A., and Zahediasl, S. (2012). Normality tests for statistical analysis: a guide for non-statisticians. International journal of endocrinology and metabolism, 10(2), 486.

[31] Box, G. E., and Cox, D. R. (1964). An analysis of transformations. Journal of the Royal Statistical Society: Series B (Methodological), 26(2), 211-243.

[32] Yeo, I. K., and Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. Biometrika, 87(4), 954-959.

[33] Li, Z., Zhao, Y., Hu, X., Botta, N., Ionescu, C., and Chen, G. (2022). Ecod: Un-

supervised outlier detection using empirical cumulative distribution functions. IEEE Transactions on Knowledge and Data Engineering.

[34] Cayton, L. (2005). Algorithms for manifold learning. Univ. of California at San Diego Tech. Rep, 12(1-17), 1.

[35] Van der Maaten, L., and Hinton, G. (2008). Visualizing data using t-SNE. Journal of machine learning research, 9(11).

[36] McInnes, L., Healy, J., and Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426.

[37] Melas-Kyriazi, L. (2020). The mathematical foundations of manifold learning. arXiv preprint arXiv:2011.01307.

[38] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[39] Han, H., Wang, W. Y., and Mao, B. H. (2005, August). Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In International conference on intelligent computing (pp. 878-887). Springer, Berlin, Heidelberg.

[40] Blagus, R., and Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. BMC bioinformatics, 14(1), 1-16.

[41] Brown, G., Wyatt, J., Harris, R., and Yao, X. (2005). Diversity creation methods: a survey and categorisation. Information fusion, 6(1), 5-20.

[42] Kittler, J., Hatef, M., Duin, R. P., and Matas, J. (1998). On combining classifiers. IEEE transactions on pattern analysis and machine intelligence, 20(3), 226-239.

[43] Bartlett, P., Freund, Y., Lee, W. S., and Schapire, R. E. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. The annals of statistics, 26(5), 1651-1686.

[44] Kuncheva, L. I., Skurichina, M., and Duin, R. P. (2002). An experimental study on diversity for bagging and boosting with linear classifiers. Information fusion, 3(4), 245-258.

[45] Cruz, R. M., Sabourin, R., and Cavalcanti, G. D. (2018). Dynamic classifier selection: Recent advances and perspectives. Information Fusion, 41, 195-216.

[46] Ko, A. H., Sabourin, R., and Britto Jr, A. S. (2008). From dynamic classifier selection to dynamic ensemble selection. Pattern recognition, 41(5), 1718-1731.

[47] Chicco, D., and Jurman, G. (2020). The advantages of the Matthews correlation

coefficient (MCC) over F1 score and accuracy in binary classification evaluation. BMC genomics, 21(1), 1-13.

[48] Oskolkov, N. (2019, Nov 2). How Exactly UMAP Works. GitHub. https://github.com/NikolayOskolkov/HowUMAPWorks

[49] McInnes , L. (2018). Scientific papers - umap 0.5 documentation. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. https://umap-learn.readthedocs.io/en/latest/scientific_papers.html

A

Appendix

A1: Features in each data block

Table A.1: List of features in Block B and their types.

Feature	Type
age	numerical
edu_years	numerical
edu_level	numerical
female	categorical
male	categorical
systolic_bp	numerical
smok	categorical
cvd_risk	categorical
APOE_E2/E2	categorical
APOE_E2/E3	categorical
APOE_E2/E4	categorical
APOE_E3/E3	categorical
APOE_E3/E4	categorical
APOE_E4/E4	categorical
dementia_history	categorical
headtrauma	categorical
cns_infec	categorical

Table A.2: List of features in Block C and their types.

Feature	Type
mmse_total	numerical
clock_score	numerical
tmta_sec	numerical
tmtb_sec	numerical
vosp_tscore	numerical
cowat_tscore	numerical
cerad_recall	numerical

Table A.3: List of features in Block D and their types.

Feature	Type
LesF	numerical
LesO	numerical
LesP	numerical
LesT	numerical
PSMD	numerical
WHo_rV	numerical

Table A.4: List of features in Block E and their types.

Feature	Type
ANT_PHC	numerical
Post_HPC	numerical
MISC	numerical
Meninges_PHC	numerical
ERC	numerical
Br35	numerical
Br36	numerical
PHC	numerical
ColSul	numerical
Meninges	numerical

A2: Normal probability plots of continuous features

Note: Only two examples of not-normal (failed normality test) features from each data block are shown here. See Jupyter notebook in code repository for more details.

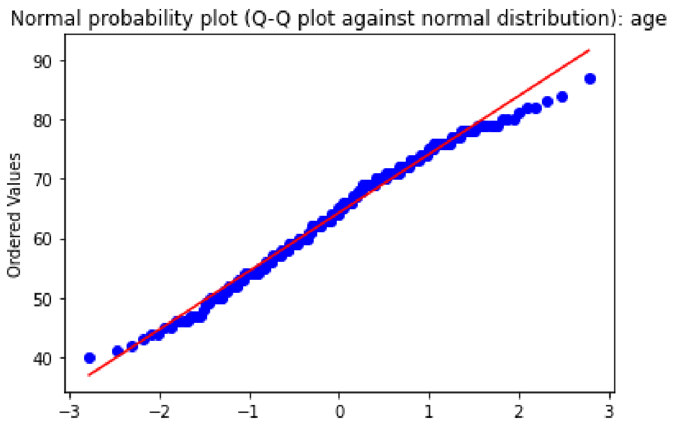


Figure A.1: Normal probability plot of the feature *age*: example from Block B.

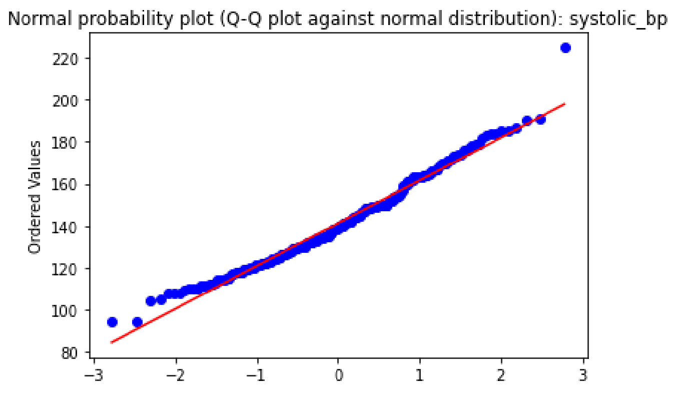


Figure A.2: Normal probability plot of the feature *systolic_bp*: example from Block B.

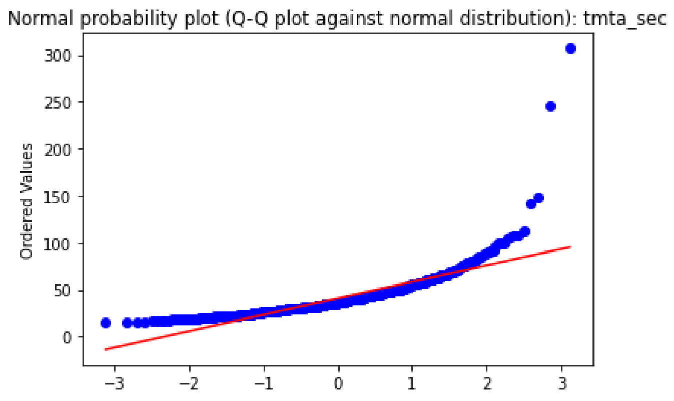


Figure A.3: Normal probability plot of the feature *tmta_sec*: example from Block C.

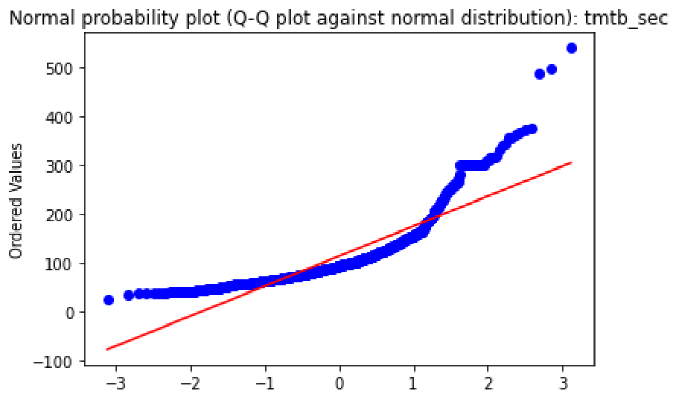


Figure A.4: Normal probability plot of the feature *tmtb_sec*: example from Block C.

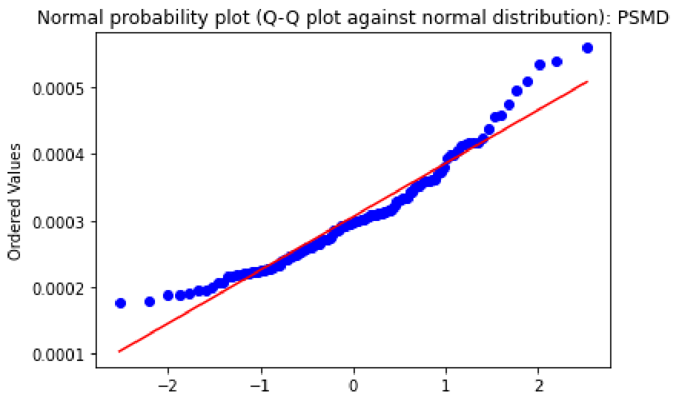


Figure A.5: Normal probability plot of the feature *PSMD*: example from Block D.

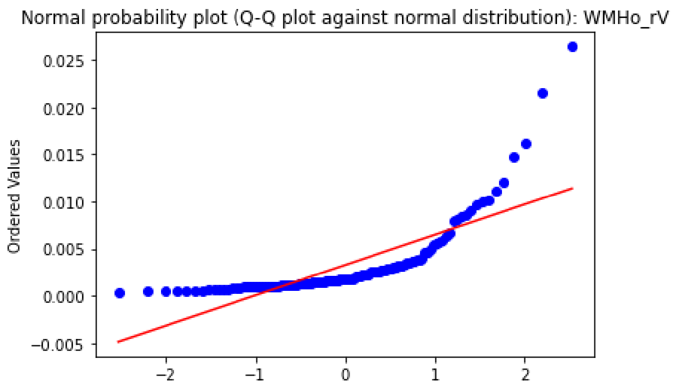


Figure A.6: Normal probability plot of the feature *WMHo_rV*: example from Block D.

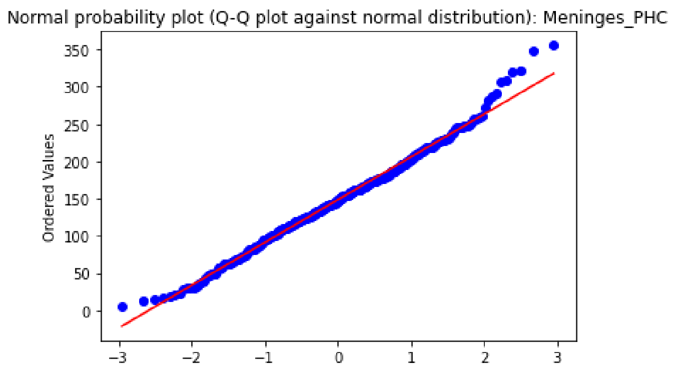


Figure A.7: Normal probability plot of the feature *Meninges_PHC*: example from Block E.

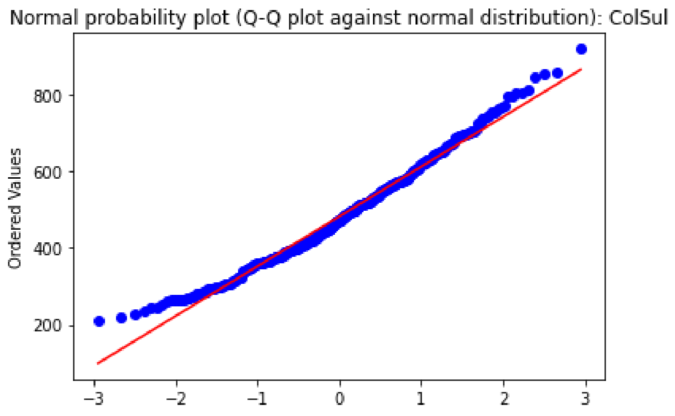
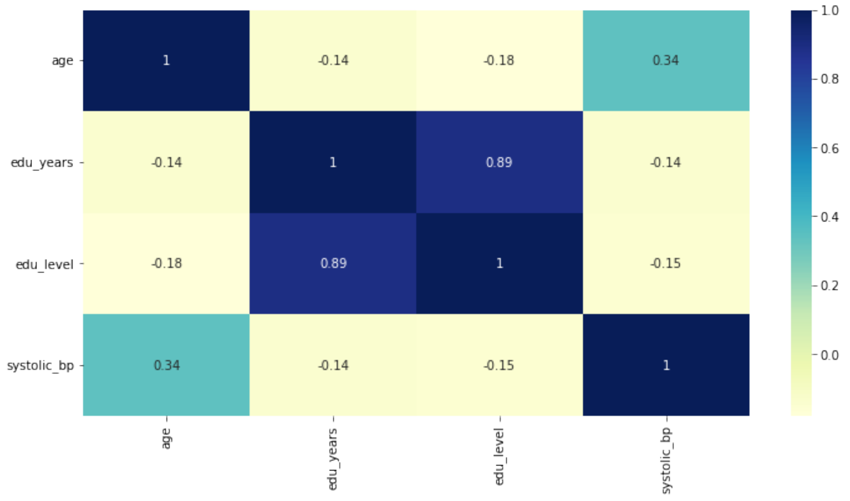


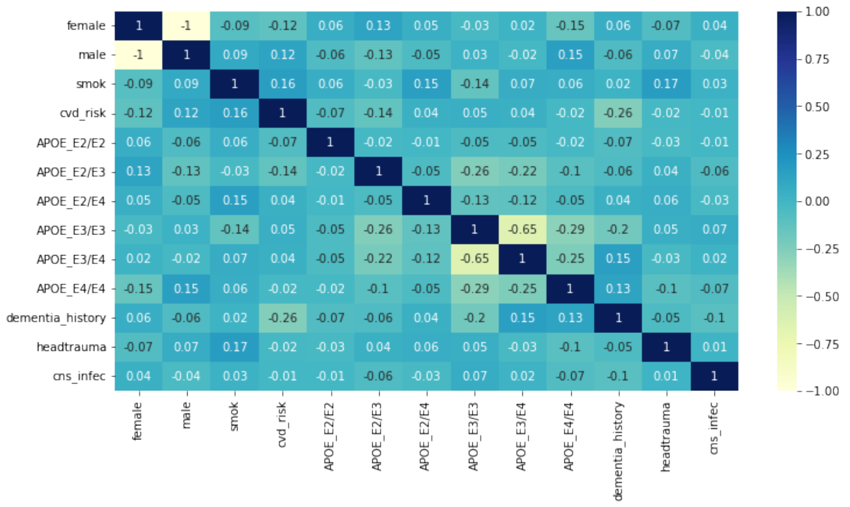
Figure A.8: Normal probability plot of the feature *ColSul*: example from Block E.

A3: Correlation of features



	age	edu_years	edu_level	systolic_bp
age	1	-0.14	-0.18	0.34
edu_years	-0.14	1	0.89	-0.14
edu_level	-0.18	0.89	1	-0.15
systolic_bp	0.34	-0.14	-0.15	1

Figure A.9: Pearson's correlation coefficients for continuous features in block B.



	female	male	smok	cvd_risk	APOE_E2/E2	APOE_E2/E3	APOE_E2/E4	APOE_E3/E3	APOE_E3/E4	APOE_E4/E4	dementia_history	headtrauma	cns_infec
female	1	-1	-0.09	-0.12	0.06	0.13	0.05	-0.03	0.02	-0.15	0.06	-0.07	0.04
male	-1	1	0.09	0.12	-0.06	-0.13	-0.05	0.03	-0.02	0.15	-0.06	0.07	-0.04
smok	-0.09	0.09	1	0.16	0.06	-0.03	0.15	-0.14	0.07	0.06	0.02	0.17	0.03
cvd_risk	-0.12	0.12	0.16	1	-0.07	-0.14	0.04	0.05	0.04	-0.02	-0.26	-0.02	-0.01
APOE_E2/E2	0.06	-0.06	0.06	-0.07	1	-0.02	-0.01	-0.05	-0.05	-0.02	-0.07	-0.03	-0.01
APOE_E2/E3	0.13	-0.13	-0.03	-0.14	-0.02	1	-0.05	-0.26	-0.22	-0.1	-0.06	0.04	-0.06
APOE_E2/E4	0.05	-0.05	0.15	0.04	-0.01	-0.05	1	-0.13	-0.12	-0.05	0.04	0.06	-0.03
APOE_E3/E3	-0.03	0.03	-0.14	0.05	-0.05	-0.26	-0.13	1	-0.65	-0.29	-0.2	0.05	0.07
APOE_E3/E4	0.02	-0.02	0.07	0.04	-0.05	-0.22	-0.12	-0.65	1	-0.25	0.15	-0.03	0.02
APOE_E4/E4	-0.15	0.15	0.06	-0.02	-0.02	-0.1	-0.05	-0.29	-0.25	1	0.13	-0.1	-0.07
dementia_history	0.06	-0.06	0.02	-0.26	-0.07	-0.06	0.04	-0.2	0.15	0.13	1	-0.05	-0.1
headtrauma	-0.07	0.07	0.17	-0.02	-0.03	0.04	0.06	0.05	-0.03	-0.1	-0.05	1	0.01
cns_infec	0.04	-0.04	0.03	-0.01	-0.01	-0.06	-0.03	0.07	0.02	-0.07	-0.1	0.01	1

Figure A.10: Phi coefficients for one-hot encoded features in block B.

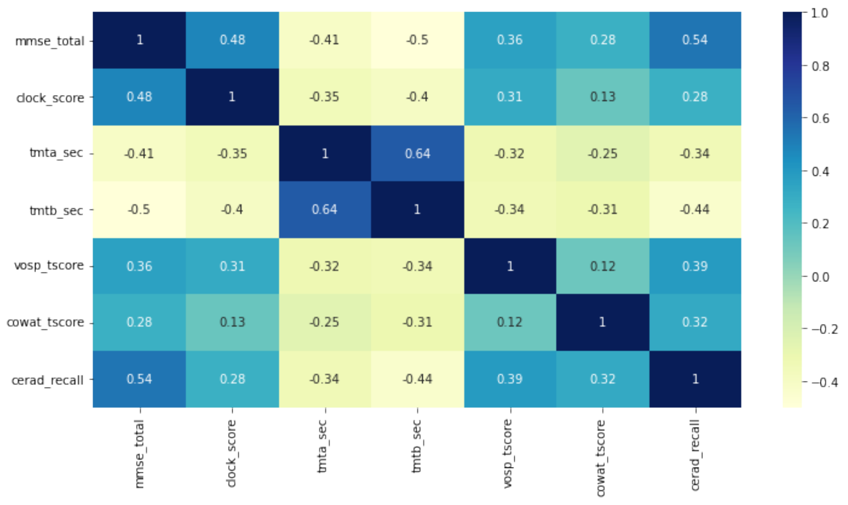


Figure A.11: Pearson's correlation coefficients for features in block C.

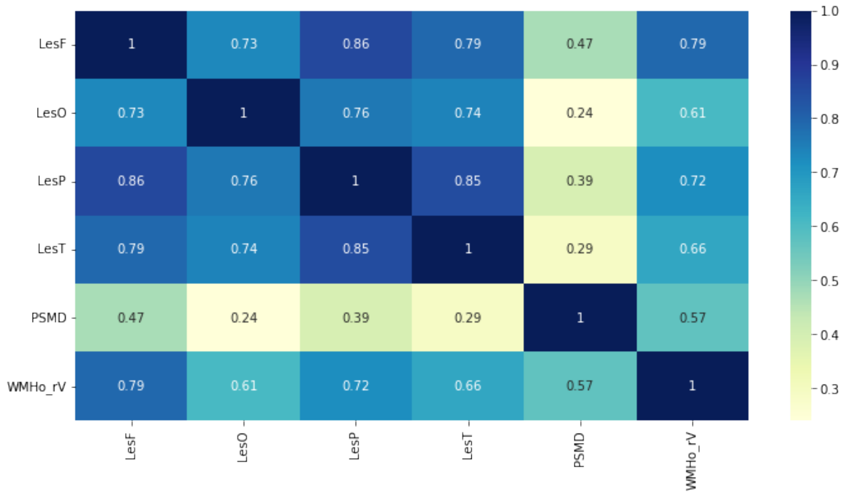


Figure A.12: Pearson's correlation coefficients for features in block D.

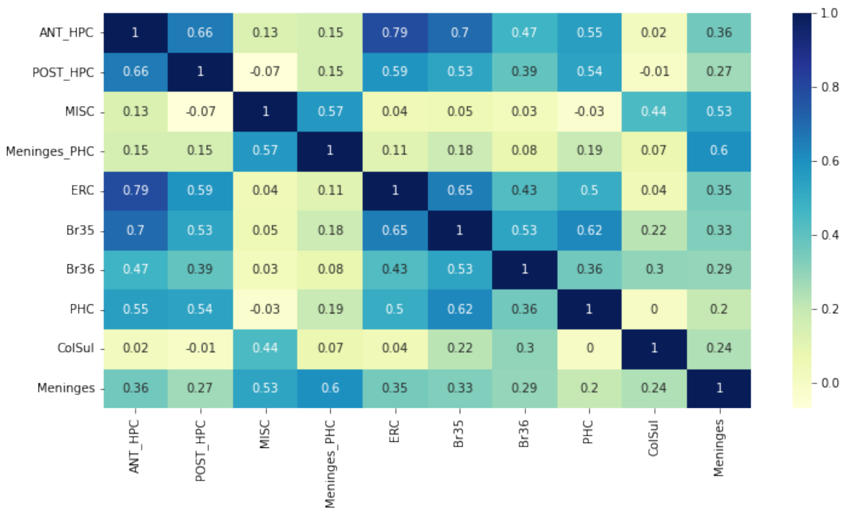


Figure A.13: Pearson's correlation coefficients for features in block E.

A4: Principal Component Analysis

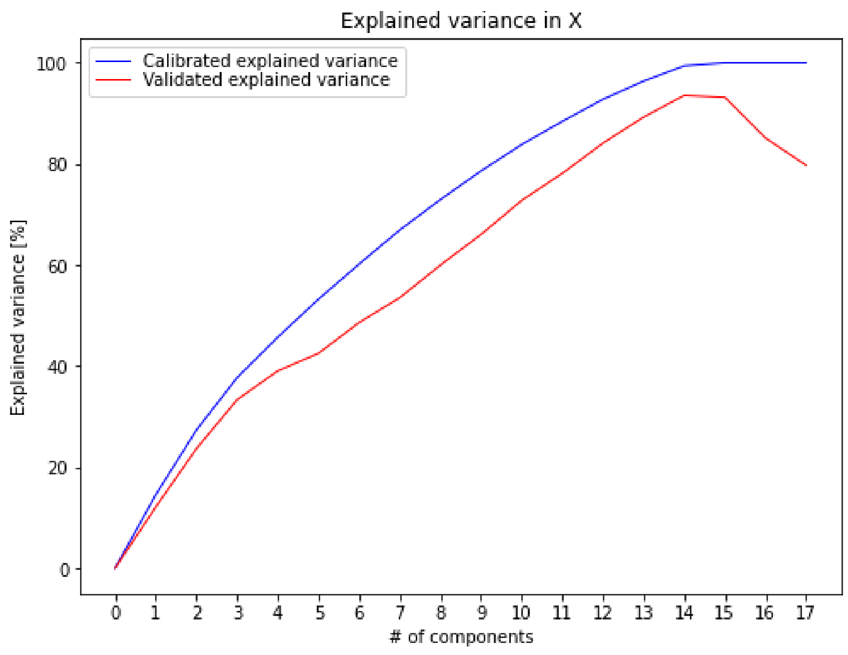


Figure A.14: Number of principal components vs. % variance explained: Block B.

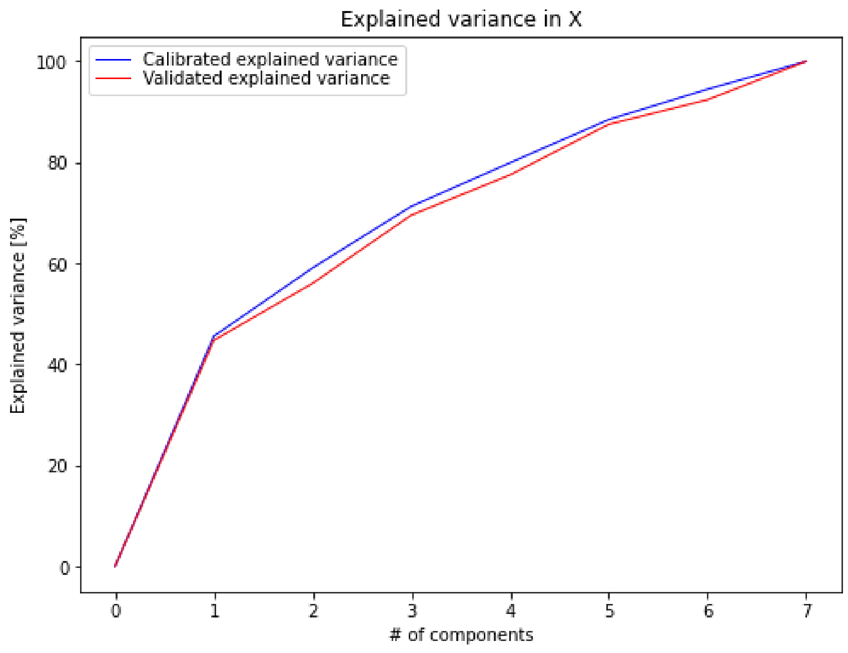


Figure A.15: Number of principal components vs. % variance explained: Block C.

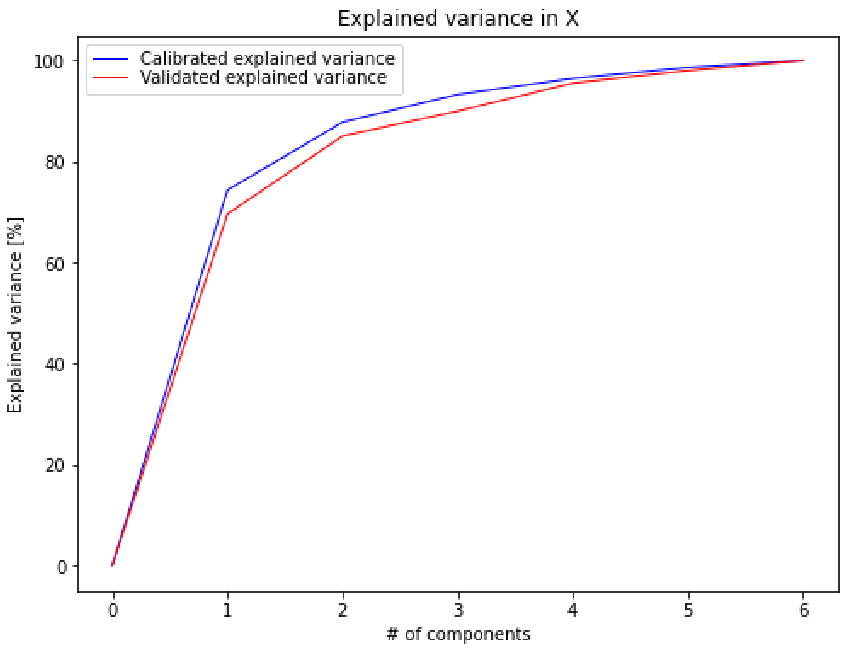


Figure A.16: Number of principal components vs. % variance explained: Block D.

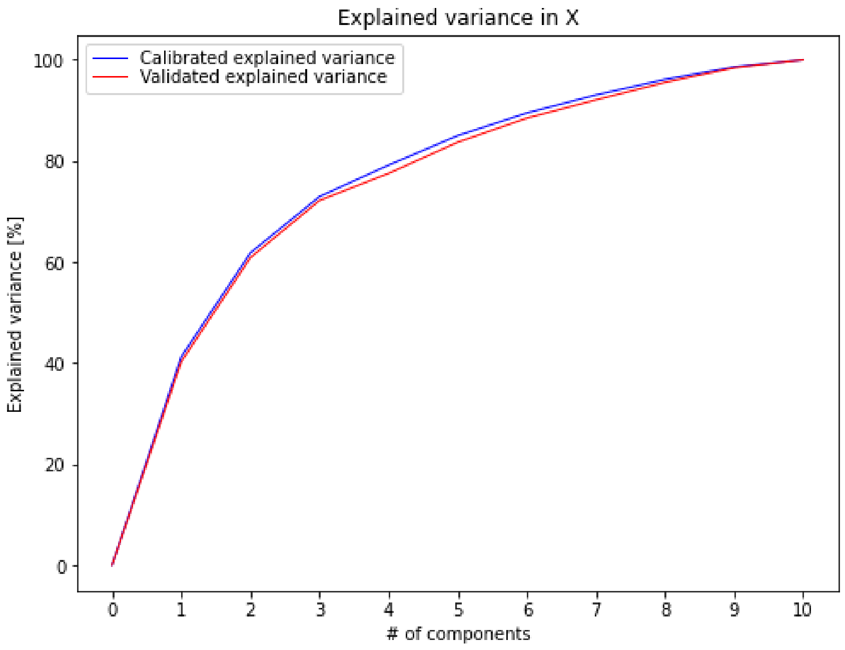
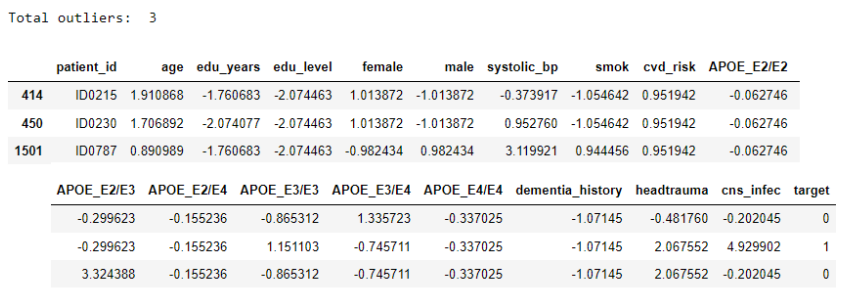


Figure A.17: Number of principal components vs. % variance explained: Block E.

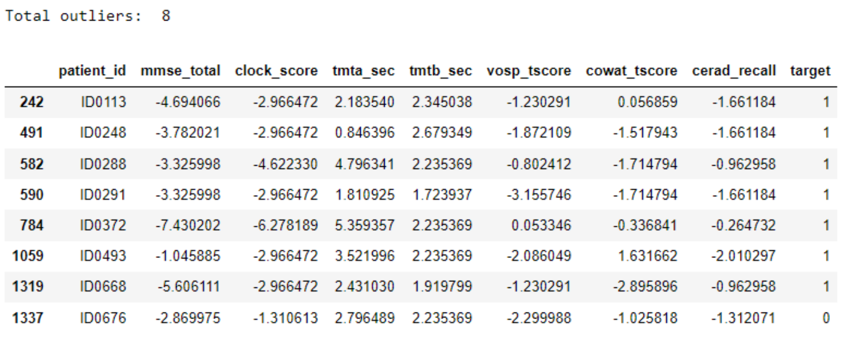
A5: Outlier detection using ECOD



```
Total outliers:  3
```

	patient_id	age	edu_years	edu_level	female	male	systolic_bp	smok	cvd_risk	APOE_E2/E2	APOE_E2/E3	APOE_E2/E4	APOE_E3/E3	APOE_E3/E4	APOE_E4/E4	dementia_history	headtrauma	cns_infec	target
414	ID0215	1.910868	-1.760683	-2.074463	1.013872	-1.013872	-0.373917	-1.054642	0.951942	-0.062746	-0.299623	-0.155236	-0.865312	1.335723	-0.337025	-1.07145	-0.481760	-0.202045	0
450	ID0230	1.706892	-2.074077	-2.074463	1.013872	-1.013872	0.952760	-1.054642	0.951942	-0.062746	-0.299623	-0.155236	1.151103	-0.745711	-0.337025	-1.07145	2.067552	4.929902	1
1501	ID0787	0.890989	-1.760683	-2.074463	-0.982434	0.982434	3.119921	0.944456	0.951942	-0.062746	3.324388	-0.155236	-0.865312	-0.745711	-0.337025	-1.07145	2.067552	-0.202045	0

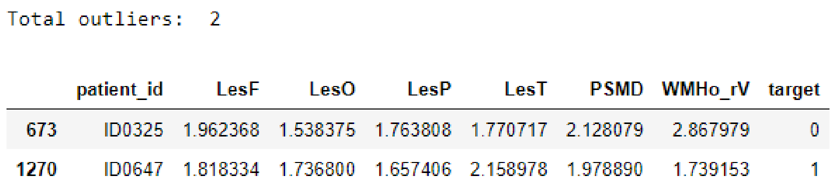
Figure A.18: Results of outlier detection using ECOD (1% contamination) for Block B.



```
Total outliers:  8
```

	patient_id	mmse_total	clock_score	tmta_sec	tmtb_sec	vosp_tscore	cowat_tscore	cerad_recall	target
242	ID0113	-4.694066	-2.966472	2.183540	2.345038	-1.230291	0.056859	-1.661184	1
491	ID0248	-3.782021	-2.966472	0.846396	2.679349	-1.872109	-1.517943	-1.661184	1
582	ID0288	-3.325998	-4.622330	4.796341	2.235369	-0.802412	-1.714794	-0.962958	1
590	ID0291	-3.325998	-2.966472	1.810925	1.723937	-3.155746	-1.714794	-1.661184	1
784	ID0372	-7.430202	-6.278189	5.359357	2.235369	0.053346	-0.336841	-0.264732	1
1059	ID0493	-1.045885	-2.966472	3.521996	2.235369	-2.086049	1.631662	-2.010297	1
1319	ID0668	-5.606111	-2.966472	2.431030	1.919799	-1.230291	-2.895896	-0.962958	1
1337	ID0676	-2.869975	-1.310613	2.796489	2.235369	-2.299988	-1.025818	-1.312071	0

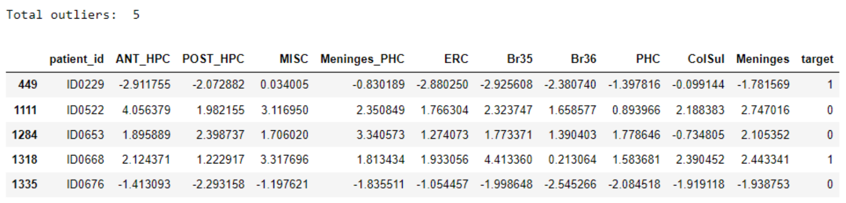
Figure A.19: Results of outlier detection using ECOD (1% contamination) for Block C.



```
Total outliers:  2
```

	patient_id	LesF	LesO	LesP	LesT	PSMD	WMHo_rV	target
673	ID0325	1.962368	1.538375	1.763808	1.770717	2.128079	2.867979	0
1270	ID0647	1.818334	1.736800	1.657406	2.158978	1.978890	1.739153	1

Figure A.20: Results of outlier detection using ECOD (1% contamination) for Block D.



```
Total outliers:  5
```

	patient_id	ANT_HPC	POST_HPC	MISC	Meninges_PHC	ERC	Br35	Br36	PHC	ColSul	Meninges	target
449	ID0229	-2.911755	-2.072882	0.034005	-0.830189	-2.880250	-2.925608	-2.380740	-1.397816	-0.099144	-1.781569	1
1111	ID0522	4.056379	1.982155	3.116950	2.350849	1.766304	2.323747	1.658577	0.893966	2.188383	2.747016	0
1284	ID0653	1.895889	2.398737	1.706020	3.340573	1.274073	1.773371	1.390403	1.778646	-0.734805	2.105352	0
1318	ID0668	2.124371	1.222917	3.317696	1.813434	1.933056	4.413360	0.213064	1.583681	2.390452	2.443341	1
1335	ID0676	-1.413093	-2.293158	-1.197621	-1.835511	-1.054457	-1.998648	-2.545266	-2.084518	-1.919118	-1.938753	0

Figure A.21: Results of outlier detection using ECOD (1% contamination) for Block E.

A6: Visualizing high-dimensional data

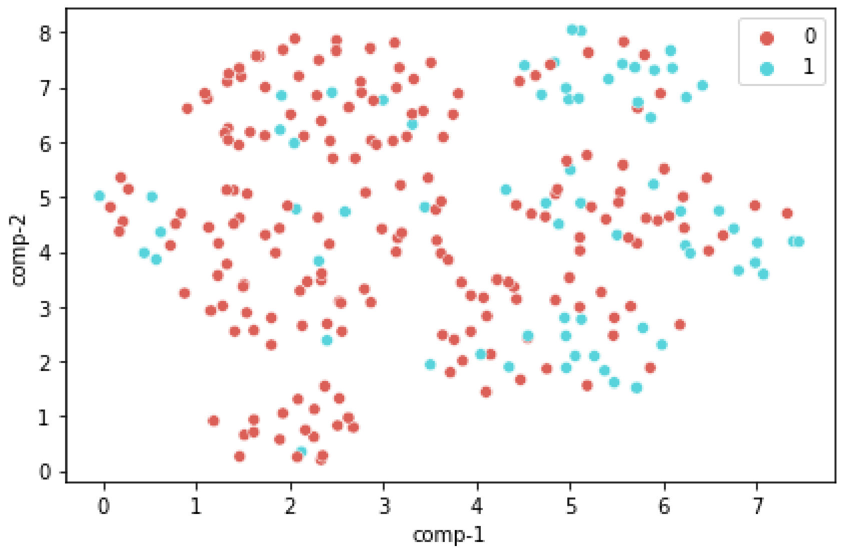


Figure A.22: UMAP embedding with 2 components for Block B.

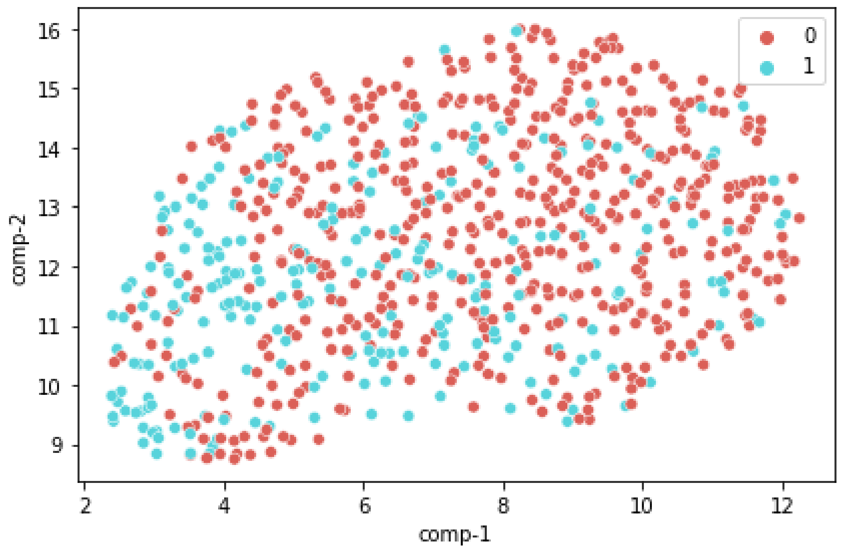


Figure A.23: UMAP embedding with 2 components for Block C.

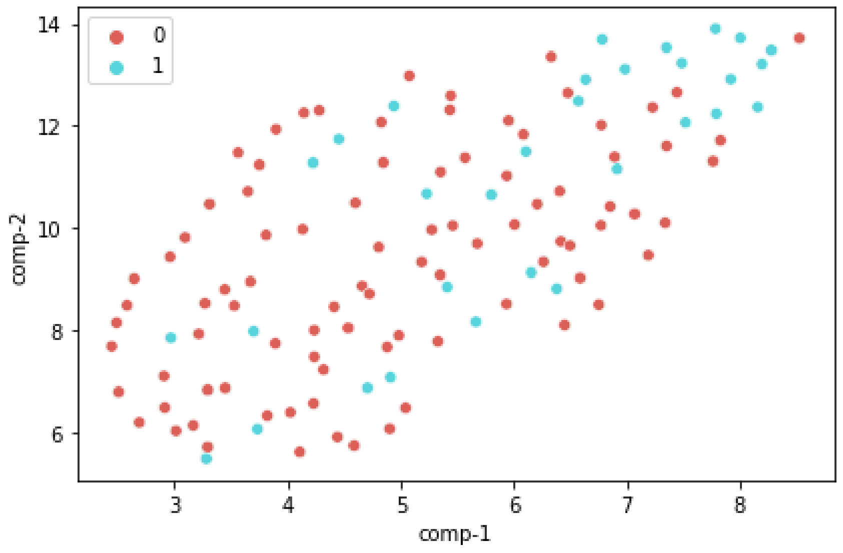


Figure A.24: UMAP embedding with 2 components for Block D.

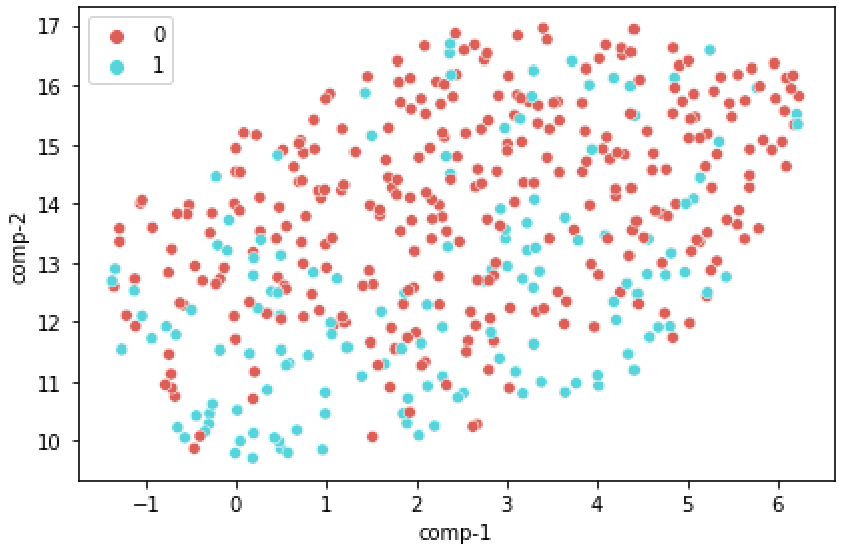


Figure A.25: UMAP embedding with 2 components for Block E.

A7: Model performance

Table A.5: MCC scores achieved after cross validation, for each data iteration and all data blocks. Corresponding scores achieved on test data are given in parentheses.

Data Iteration ID	Block B	Block C	Block D	Block E
DI-00	0.478 (0.219)	0.201 (0.174)	0.292 (0.089)	0.259 (0.343)
DI-01	0.413 (0.254)	0.231 (0.227)	0.164 (0.142)	0.238 (0.280)
DI-02	0.431 (0.144)	0.209 (0.110)	0.155 (0.153)	0.240 (-0.013)
DI-03	0.439 (0.264)	0.316 (0.249)	0.369 (0.227)	0.333 (0.000)
DI-04	0.650 (0.145)	0.332 (0.142)	0.555 (0.165)	0.473 (0.000)
DI-05	0.648 (0.132)	0.251 (0.099)	0.410 (-0.01)	0.442 (0.000)
DI-06	0.375 (0.277)	0.259 (0.111)	0.317 (0.220)	0.302 (0.000)
DI-07	0.656 (0.245)	0.502 (0.124)	0.451 (0.122)	0.551 (-0.058)
DI-08	0.516 (0.000)	0.384 (0.186)	0.444 (0.095)	0.350 (-0.13)
DI-09	0.334 (-0.141)	0.246 (0.248)	0.242 (0.142)	0.270 (0.000)
DI-10	0.439 (0.264)	0.301 (0.205)	0.302 (0.087)	0.331 (0.316)
DI-11	0.568 (0.177)	0.449 (0.082)	0.438 (0.099)	0.506 (0.293)
DI-12	0.553 (0.350)	0.339 (0.128)	0.276 (-0.028)	0.443 (0.118)
DI-13	0.375 (0.277)	0.314 (0.236)	0.184 (0.057)	0.354 (0.205)
DI-14	0.628 (0.198)	0.500 (0.174)	0.509 (0.201)	0.486 (0.220)
DI-15	0.418 (0.193)	0.354 (0.255)	0.348 (0.110)	0.346 (0.095)
DI-16	0.304 (0.314)	0.244 (0.250)	0.270 (-0.03)	0.315 (0.174)

Table A.6: Accuracy scores achieved after cross validation, for each data iteration and all data blocks. Corresponding scores achieved on test data are given in parentheses.

Data Iteration ID	Block B	Block C	Block D	Block E
DI-00	0.807 (0.720)	0.671 (0.677)	0.744 (0.646)	0.686 (0.744)
DI-01	0.780 (0.720)	0.679 (0.701)	0.711 (0.671)	0.672 (0.732)
DI-02	0.788 (0.713)	0.670 (0.659)	0.711 (0.671)	0.673 (0.701)
DI-03	0.791 (0.689)	0.720 (0.732)	0.773 (0.732)	0.715 (0.713)
DI-04	0.824 (0.652)	0.665 (0.677)	0.772 (0.659)	0.736 (0.713)
DI-05	0.823 (0.598)	0.625 (0.573)	0.703 (0.579)	0.720 (0.713)
DI-06	0.760 (0.713)	0.697 (0.652)	0.757 (0.732)	0.707 (0.713)
DI-07	0.827 (0.671)	0.751 (0.646)	0.722 (0.652)	0.774 (0.543)
DI-08	0.755 (0.713)	0.692 (0.659)	0.717 (0.604)	0.674 (0.476)
DI-09	0.754 (0.421)	0.685 (0.707)	0.740 (0.689)	0.693 (0.713)
DI-10	0.791 (0.689)	0.708 (0.677)	0.740 (0.652)	0.714 (0.726)
DI-11	0.782 (0.652)	0.724 (0.634)	0.714 (0.652)	0.752 (0.707)
DI-12	0.776 (0.701)	0.669 (0.634)	0.636 (0.604)	0.721 (0.604)
DI-13	0.760 (0.713)	0.716 (0.720)	0.710 (0.665)	0.728 (0.683)
DI-14	0.812 (0.652)	0.750 (0.683)	0.750 (0.707)	0.743 (0.646)
DI-15	0.708 (0.634)	0.677 (0.701)	0.672 (0.622)	0.671 (0.561)
DI-16	0.736 (0.744)	0.684 (0.726)	0.756 (0.622)	0.708 (0.683)

Table A.7: ROC AUC scores achieved after cross validation, for each data iteration and all data blocks. Corresponding scores achieved on test data are given in parentheses.

Data Iteration ID	Block B	Block C	Block D	Block E
DI-00	0.752 (0.587)	0.619 (0.583)	0.623 (0.542)	0.637 (0.661)
DI-01	0.686 (0.612)	0.620 (0.606)	0.567 (0.566)	0.636 (0.621)
DI-02	0.696 (0.545)	0.618 (0.551)	0.582 (0.572)	0.639 (0.498)
DI-03	0.718 (0.636)	0.658 (0.596)	0.641 (0.576)	0.680 (0.500)
DI-04	0.838 (0.572)	0.714 (0.563)	0.778 (0.582)	0.754 (0.500)
DI-05	0.833 (0.572)	0.636 (0.554)	0.716 (0.495)	0.724 (0.500)
DI-06	0.694 (0.634)	0.628 (0.553)	0.637 (0.570)	0.657 (0.500)
DI-07	0.853 (0.629)	0.777 (0.561)	0.746 (0.559)	0.785 (0.469)
DI-08	0.769 (0.500)	0.708 (0.595)	0.731 (0.550)	0.691 (0.429)
DI-09	0.650 (0.422)	0.630 (0.617)	0.589 (0.559)	0.631 (0.500)
DI-10	0.718 (0.636)	0.656 (0.602)	0.685 (0.540)	0.685 (0.655)
DI-11	0.799 (0.591)	0.741 (0.540)	0.731 (0.546)	0.775 (0.648)
DI-12	0.781 (0.689)	0.686 (0.565)	0.644 (0.487)	0.734 (0.563)
DI-13	0.694 (0.634)	0.650 (0.600)	0.574 (0.523)	0.681 (0.600)
DI-14	0.827 (0.604)	0.769 (0.580)	0.757 (0.585)	0.770 (0.618)
DI-15	0.729 (0.604)	0.697 (0.625)	0.680 (0.557)	0.689 (0.552)
DI-16	0.662 (0.636)	0.629 (0.604)	0.595 (0.487)	0.669 (0.580)

Table A.8: F1 scores achieved after cross validation, for each data iteration and all data blocks. Corresponding scores achieved on test data are given in parentheses.

Data Iteration ID	Block B	Block C	Block D	Block E
DI-00	0.579 (0.361)	0.426 (0.391)	0.435 (0.326)	0.482 (0.512)
DI-01	0.545 (0.425)	0.454 (0.424)	0.316 (0.357)	0.473 (0.436)
DI-02	0.558 (0.230)	0.439 (0.333)	0.320 (0.372)	0.474 (0.039)
DI-03	0.568 (0.485)	0.495 (0.371)	0.455 (0.312)	0.534 (0.000)
DI-04	0.821 (0.387)	0.658 (0.346)	0.787 (0.404)	0.740 (0.000)
DI-05	0.826 (0.421)	0.628 (0.407)	0.702 (0.289)	0.723 (0.000)
DI-06	0.527 (0.472)	0.460 (0.345)	0.467 (0.290)	0.497 (0.000)
DI-07	0.823 (0.481)	0.752 (0.370)	0.738 (0.360)	0.776 (0.272)
DI-08	0.751 (0.000)	0.694 (0.429)	0.727 (0.381)	0.681 (0.259)
DI-09	0.480 (0.296)	0.465 (0.442)	0.391 (0.320)	0.484 (0.000)
DI-10	0.568 (0.485)	0.502 (0.430)	0.455 (0.313)	0.535 (0.505)
DI-11	0.775 (0.424)	0.716 (0.333)	0.727 (0.329)	0.756 (0.500)
DI-12	0.778 (0.559)	0.669 (0.388)	0.649 (0.235)	0.720 (0.404)
DI-13	0.527 (0.472)	0.503 (0.395)	0.350 (0.247)	0.543 (0.422)
DI-14	0.809 (0.447)	0.749 (0.381)	0.763 (0.368)	0.749 (0.473)
DI-15	0.710 (0.455)	0.675 (0.462)	0.688 (0.380)	0.681 (0.410)
DI-16	0.468 (0.462)	0.464 (0.400)	0.388 (0.205)	0.522 (0.381)

Table A.9: Precision scores achieved after cross validation, for each data iteration and all data blocks. Corresponding scores achieved on test data are given in parentheses.

Data Iteration ID	Block B	Block C	Block D	Block E
DI-00	0.713 (0.520)	0.473 (0.425)	0.514 (0.359)	0.511 (0.564)
DI-01	0.619 (0.515)	0.489 (0.474)	0.411 (0.405)	0.492 (0.548)
DI-02	0.638 (0.500)	0.471 (0.378)	0.372 (0.410)	0.493 (0.250)
DI-03	0.637 (0.462)	0.584 (0.565)	0.657 (0.588)	0.565 (0.000)
DI-04	0.833 (0.391)	0.676 (0.412)	0.760 (0.404)	0.730 (0.000)
DI-05	0.816 (0.358)	0.624 (0.338)	0.712 (0.280)	0.719 (0.000)
DI-06	0.566 (0.500)	0.525 (0.375)	0.515 (0.600)	0.561 (0.000)
DI-07	0.836 (0.439)	0.751 (0.378)	0.709 (0.381)	0.773 (0.250)
DI-08	0.766 (0.000)	0.689 (0.412)	0.704 (0.345)	0.666 (0.217)
DI-09	0.551 (0.227)	0.503 (0.487)	0.477 (0.429)	0.529 (0.000)
DI-10	0.637 (0.462)	0.543 (0.435)	0.520 (0.361)	0.561 (0.523)
DI-11	0.802 (0.404)	0.736 (0.349)	0.694 (0.368)	0.744 (0.490)
DI-12	0.771 (0.484)	0.669 (0.373)	0.628 (0.263)	0.723 (0.355)
DI-13	0.566 (0.500)	0.570 (0.517)	0.412 (0.346)	0.585 (0.442)
DI-14	0.825 (0.411)	0.750 (0.432)	0.727 (0.483)	0.732 (0.413)
DI-15	0.709 (0.397)	0.678 (0.477)	0.659 (0.358)	0.664 (0.333)
DI-16	0.518 (0.581)	0.501 (0.536)	0.542 (0.258)	0.550 (0.432)

Table A.10: Recall scores achieved after cross validation, for each data iteration and all data blocks. Corresponding scores achieved on test data are given in parentheses.

Data Iteration ID	Block B	Block C	Block D	Block E
DI-00	0.502 (0.277)	0.389 (0.362)	0.414 (0.298)	0.457 (0.468)
DI-01	0.501 (0.362)	0.426 (0.383)	0.300 (0.319)	0.461 (0.362)
DI-02	0.508 (0.149)	0.414 (0.298)	0.302 (0.340)	0.461 (0.021)
DI-03	0.516 (0.511)	0.435 (0.277)	0.402 (0.213)	0.511 (0.000)
DI-04	0.812 (0.383)	0.645 (0.298)	0.828 (0.404)	0.751 (0.000)
DI-05	0.839 (0.511)	0.633 (0.511)	0.700 (0.298)	0.731 (0.000)
DI-06	0.508 (0.447)	0.411 (0.319)	0.436 (0.191)	0.457 (0.000)
DI-07	0.815 (0.532)	0.753 (0.362)	0.778 (0.340)	0.783 (0.298)
DI-08	0.744 (0.000)	0.700 (0.447)	0.767 (0.426)	0.699 (0.319)
DI-09	0.442 (0.426)	0.435 (0.404)	0.338 (0.255)	0.450 (0.000)
DI-10	0.516 (0.511)	0.470 (0.426)	0.433 (0.277)	0.514 (0.489)
DI-11	0.755 (0.447)	0.699 (0.319)	0.772 (0.298)	0.773 (0.511)
DI-12	0.788 (0.660)	0.672 (0.404)	0.678 (0.213)	0.719 (0.468)
DI-13	0.508 (0.447)	0.454 (0.319)	0.333 (0.191)	0.511 (0.404)
DI-14	0.798 (0.489)	0.750 (0.340)	0.811 (0.298)	0.768 (0.553)
DI-15	0.715 (0.532)	0.675 (0.447)	0.722 (0.404)	0.704 (0.532)
DI-16	0.445 (0.383)	0.435 (0.319)	0.307 (0.170)	0.500 (0.340)