

Norwegian University
of Life Sciences

Master's Thesis 2021 30 ECTS
Faculty of Science and Technology

Using Machine learning and Repeated Elastic Net Technique for identification of biomarkers of early Alzheimer's disease

Charlott Olofsson
Master of Science in Data Science

Preface

This thesis is written at the Faculty of Science and Technology at the Norwegian University of Life Sciences (NMBU) in 2021 and marks the end of my master's degree in Data Science. The data used in this thesis has been provided by Computational Radiology and Artificial Intelligence (CRAI), in Oslo University Hospital.

First of all, I would like to thank my supervisor Associate Professor Oliver Tomic for giving me the opportunity to work with this relevant issue in my master's thesis, and for his guidance, engagement and feedback. I would also like to thank my co-supervisors Associate Professors Cecilia Marie Futsæther and Kristian Hovde Liland, for suggestions and feedback.

Furthermore, I would like to thank Per Selnes, Lene Pålhaugen and Jonas Alexander Jarholm from Akershus Universitetssykehus, for their collaboration and guidance when working with medical data.

Thanks to my fellow students at NMBU for the collaboration throughout this study, and to my family, for the support and encouragement along the way. A special thanks to my boyfriend for his patience through my time as a student, and to Sia and Indy for keeping me company in the hours spent writing this thesis.

Ås, 14th December, 2021

Charlott Kjærre Olofsson

Abstract

Alzheimer's disease is a neurodegenerative brain disease that damages neurons in the part of the brain involved in cognitive function, and early diagnosis is crucial for treatment that could slow down the progression of the disease. In the preclinical stage, the accumulation of a protein fragment called amyloid-beta outside the neurons can be associated with the early onset of Alzheimer's disease.

The aim of the study was to identify biomarkers (features) for early detection of Alzheimer's disease using data from patients known to have an accumulation of amyloid-beta in their brains. 44 features from different sources were used and divided into 5 blocks of similar measurements. A baseline analysis was done with all the features combined, consisting of 172 patient assessments with complete measurements, where 49 had presence of amyloid-beta. The same patient assessments were used as the test data for block-wise analysis. This study includes exploratory analysis of the data using correlation, principal components analysis (PCA) and partial least squares regression (PLSR). The performance outcomes of five different classifiers were compared when trying to separate the two classes. Repeated Elastic Net Technique (RENT) was used for feature selection, in combination with repeated stratified k-fold validation for the acquisition of robust results.

Using the selected features from the RENT analysis, the best performing classifier for the individual blocks were identified through repeated stratified k-fold validation. A final prediction of the class was computed from the prediction of each block using a performance-based weighted average. The final score based on this weighted average did not exceed the score of the baseline study.

The block consisting of factors related to environment and heritage provided the highest predictive performance. In the baseline analysis with RENT, the factors related to heritage came out as important for the classification task, along with features related to cognitive tests. From the features containing information from MR-images of the brain, white matter hyperintensity and lesion measured in the occipital lobe can be considered as important for both the baseline analysis and the block-wise analysis.

Sammendrag

Alzheimers sykdom er en nevrodegenerativ hjernesykdom som skader nevroner i den delen av hjernen som er involvert i kognitiv funksjon, og tidlig diagnose er avgjørende for behandling som kan bremse utviklingen av sykdommen. I det prekliniske stadiet kan akkumulering av et proteinfragment kalt amyloid-beta utenfor nevronene assosieres med begynnelsen av Alzheimers sykdom.

Målet med dette studiet var å identifisere biomarkører (variabler) for tidlig oppdagelse av Alzheimers sykdom ved å bruke data fra pasienter som er kjent for å ha en akkumulering av amyloid-beta i hjernen. 44 variabler fra forskjellige kilder ble brukt og delt inn i 5 blokker med lignende målinger. En baseline-analyse ble gjort med alle variablene kombinert, bestående av 172 pasientvurderinger med komplette målinger, der 49 hadde tilstedeværelse av amyloid-beta. De samme pasientvurderingene ble brukt som testdata for blokkvis analyse. Dette studiet inkluderer utforskende analyse av dataene ved bruk av korrelasjon, hovedkomponentanalyse (PCA) og partiell minste kvadraters regresjon (PLSR). Ytelsen til fem forskjellige modeller for klassifisering ble sammenlignet for å skille mellom de to klassene. Repeated Elastic Net Technique (RENT) ble brukt for variabel seleksjon, i kombinasjon med gjentatt stratifisert k-fold validering for å oppnå robuste resultater.

Ved å bruke de selekterte variablene fra RENT-analysen, ble den modellen med best ytelse for de individuelle blokkene identifisert gjennom gjentatt stratifisert k-fold validering. En endelig prediksjon av klassen ble beregnet fra prediksjonen for hver blokk ved å bruke et ytelsesbasert vektet gjennomsnitt. Den endelige poengsummen basert på dette vektete gjennomsnittet oversteg ikke resultatet i baseline-analysen.

Blokken bestående av faktorer knyttet til miljø og arv ga den høyeste prediktive ytelsen. I baseline-analysen med RENT kom variablene knyttet til arv ut som viktige for klassifiseringsoppgaven, sammen med variabler knyttet til kognitive tester. Fra variablene som inneholder informasjon fra MR-bilder av hjernen, kom hyperintensitet og lesjon i hvit substans målt i occipitallappen ut som viktige både for baseline-analysen og den blokkvise analysen.

Contents

1	Introduction	15
1.1	Background	15
1.2	Previous work	18
1.3	Problem statement	20
1.4	Structure of thesis	20
2	Theory	21
2.1	Preprocessing	23
2.1.1	Data transformation	23
2.1.2	Data reduction	24
2.1.3	One-hot-encoding	25
2.2	Exploratory analysis	25
2.2.1	Correlation	26
2.2.2	RV matrix correlation	26
2.2.3	Principal Component Analysis	27
2.2.4	Partial Least Squares Regression	30
2.3	Models for classification	32
2.3.1	Logistic Regression	35
2.3.2	Support Vector Machine	36
2.3.3	Decision Tree	37
2.3.4	Random Forest	38
2.3.5	K-Nearest Neighbors	39
2.3.6	Passive Aggressive Classifier	40
2.4	Model Evaluation	41
2.4.1	Regularisation	42
2.4.2	Splitting the data	44
2.4.3	Hyperparameters	44
2.4.4	Cross-validation	44
2.4.5	Scoring metrics	45
2.5	Repeated Elastic Net Technique	47
2.6	Ensemble learning	49

3	Materials	53
3.1	Data collection	53
3.2	Block structure	54
3.2.1	Block A: Background information	54
3.2.2	Block B: Environment and heritage	54
3.2.3	Block C: Cognitive tests	56
3.2.4	Block D: White matter hyperintensity load by region	56
3.2.5	Block E: MR images of subcortical brain structures	58
4	Methods	59
4.1	Software	59
4.2	Workflow	60
4.3	Data preprocessing	61
4.3.1	Target	61
4.3.2	One Hot Encoding	61
4.3.3	Data transformation	62
4.4	Explorative analysis	62
4.4.1	Correlation between features and blocks	62
4.4.2	PCA	63
4.4.3	PLSR	63
4.5	Data splitting	63
4.6	Classification	65
4.6.1	Baseline model	65
4.6.2	Block-wise models	65
4.6.3	Tuning	66
4.6.4	Performance metrics	67
4.6.5	Evaluation	67
4.7	Feature selection	67
4.8	Ensemble learning	68

5	Results	69
5.1	Data preprocessing	70
5.1.1	Power transformation	70
5.1.2	Missing values	71
5.2	Explorative analysis	73
5.2.1	Correlation	73
5.2.2	PCA	77
5.2.3	PLSR	80
5.2.4	Outlier detection	83
5.3	Classification	83
5.3.1	Baseline model	83
5.3.2	Block-wise models	87
5.4	RENT	88
5.4.1	Baseline	88
5.4.2	Block-wise	92
5.5	Ensemble	101
6	Discussion	105
6.1	The data	105
6.2	Data preprocessing	106
6.3	Explorative analysis	107
6.4	Classification	110
6.5	RENT	113
6.6	Weighted Average	119
6.7	Further work	119
7	Conclusions	121
	Bibliography	i
	Appendix	vii

List of Tables

3.1	Block A - Background information	54
3.2	Block B - Environment and heritage	55
3.3	Block C - Cognitive tests	56
3.4	Block D - White matter hyperintensity	57
3.5	Block D - Combined white matter hyperintensity	57
3.6	Block E - Subcortical brain structure images	58
4.1	Patient assessments with complete data	61
4.2	Block dimensions for training- and test data	65
4.3	Hyperparameter testing	66
5.1	Hyperparameters results	86
5.2	Baseline scores	87
5.3	Block scores	88
5.4	Baseline selected features	91
5.5	Baseline RENT scores	92
5.6	Parameters RENT	92
5.7	Block B - Selected features	98
5.8	Block C - Selected features	99
5.9	Block D - Selected features	99
5.10	Block E - Selected features	99
5.11	Baseline RENT scores	100
5.12	Final features	101
5.13	Final models	102
6.8	Comparing baseline scores	115

List of Figures

1.1	Neuropathology AD	16
1.2	PET scan	17
2.1	Data science term	22
2.2	One-hot-encoding	25
2.3	Scores plot	29
2.4	Loadings plot	29
2.5	Regression	30
2.6	Perceptron	32
2.7	Gradient descent	34
2.8	Sigmoid function	35
2.9	Support Vector Machine	36
2.10	Decision tree	37
2.11	Decision Forest	39
2.12	K-Nearest Neighbors	40
2.13	Bias-variance	41
2.14	Optimum balance	42
2.15	Regularisation	43
2.16	K-fold cross-validation	45
2.17	Confusion matrix	46
2.18	Repeated Elastic Net Teqnique	49
2.19	Bagging ensemble	50
2.20	Boosting ensemble	51
2.21	Stacking ensemble	52
4.1	Workflow	60
4.2	Data splitting	64
5.1	Distribution	70
5.2	Distribution	70
5.3	Block A - Missing	71
5.4	Block B - Missing	71
5.5	Block C - Missing	72

5.6	Block D - Missing	72
5.7	Block E - Missing	73
5.8	Matrix Correlation Coefficient	73
5.9	Block A - Correlation Coefficient	74
5.10	Block B - Correlation Coefficient	75
5.11	Block C - Correlation Coefficient	75
5.12	Block D - Correlation Coefficient	76
5.13	Block E - Correlation Coefficient	77
5.14	PCA scores	78
5.15	PCA loadings	78
5.16	PCA Correlation loadings	79
5.17	PCA Explained variance	79
5.18	PLSR scores	80
5.19	PLSR loading	81
5.20	PLSR correlation loadings	81
5.21	PLSR explained variance in x	82
5.22	PLSR explained variance in y	82
5.23	Outlier detection	83
5.24	Baseline Logistic Regression	84
5.25	Baseline Passive Aggressive Classifier	84
5.26	Baseline Random Forest	85
5.27	Baseline KNN	85
5.28	Baseline SVM	86
5.29	Baseline heatmaps RENT	89
5.30	Baseline selection frequency RENT	89
5.31	Baseline elementary models RENT	90
5.32	Baseline τ scores	91
5.33	Block A - heatmaps RENT	93
5.34	Block B: Tau	94
5.35	Block C: Tau	95
5.36	Block D: Tau	96
5.37	Block E: Tau	97
5.38	Boxplot scores	100
5.39	Predictions heatmap	102

Chapter 1

Introduction

1.1 Background

Dementia is a general term for brain diseases that causes loss of memory, language problems, and other thinking abilities that are serious enough to interfere with daily life. One of the most common causes of Dementia is Alzheimer's Disease (AD), which is responsible for 60 – 80% of Dementia cases [1].

AD is a neurodegenerative brain disease that damages or destroys neurons in the part of the brain involved in cognitive functions. The developing process of the disease consists of five stages. The first stage is preclinical AD, then mild cognitive impairment (MCI) due to AD, and the last three stages consist of Dementia in the mild, moderate, and severe categories. In the preclinical stage, the patient does not notice any symptoms, but changes in the brain are measurable. Brain changes associated with AD are mainly accumulation of a protein fragment called amyloid-beta outside the neurons and accumulation of an unusual form of the protein tau inside the neurons, called tau tangles. Fig. 1.1 gives an illustration of how the two proteins affect a diseased neuron. The illustration includes a healthy neuron for comparison. Other forms of brain changes associated with the disease include inflammation and atrophy [2]. Atrophy related to AD is the loss of neurons and the connection between the neurons. This reduces brain tissue and causes the brain to shrink.

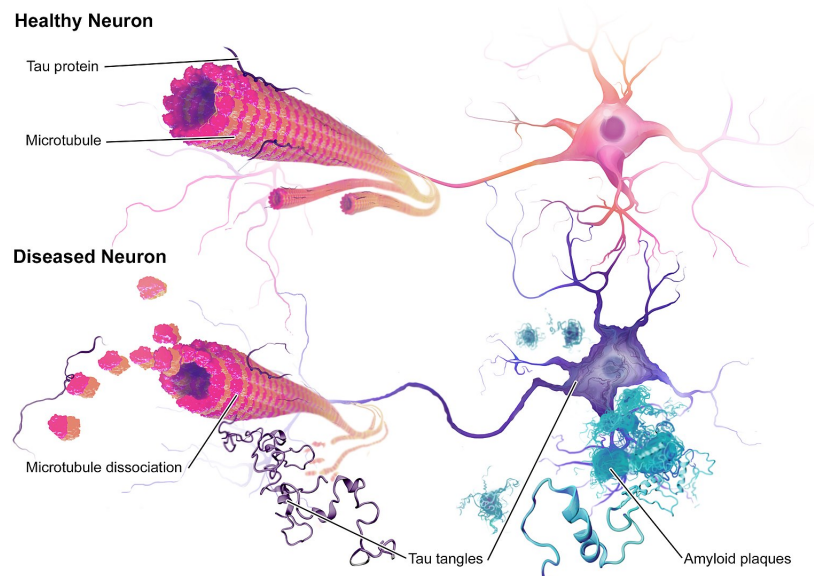


Figure 1.1: Illustration of the neuropathology of AD. Comparison of a healthy neuron and a diseased neuron. Figure adapted from [3]

A positron emission tomography (PET) scan and analysis of cerebrospinal fluid (CSF) can show abnormal amyloid-beta levels. Not all individuals with abnormal levels of beta-amyloid go on to develop dementia/symptoms of MCI [1]. Figure 1.2 shows PET scans that detect amyloid-beta plaques, comparing scans from a normal brain with a brain with AD. Another early indication for AD is the decreased metabolism of glucose, which is also visible in PET scans [2].

Mild symptoms begin to appear during the stage of MCI due to AD, but they do not interfere with daily activities. More severe symptoms develop during the last three stages. AD symptoms include memory loss, cognitive deficits, problems with recognition, spatial awareness, speaking, reading, or writing. Other symptoms related to the disease are personality and behaviour changes [1]. As the disease progresses, it will affect neurons in different brain parts, and ultimately lead to premature death [2].

The disease progresses gradually, starting with minor changes in the brain. Researchers believe it can take 20 years or more for symptoms like memory loss and language problems to develop [2]. A study showed that through analysis of CSF, a mutation in a gene suggests that AD has a pre-symptomatic period of 40-50 years [5]. An early diagnosis opens up treatment opportunities that may slow down the progression of the disease [6]. There exist medications that could temporarily improve cognitive symptoms by either increasing neurotransmitters in the brain

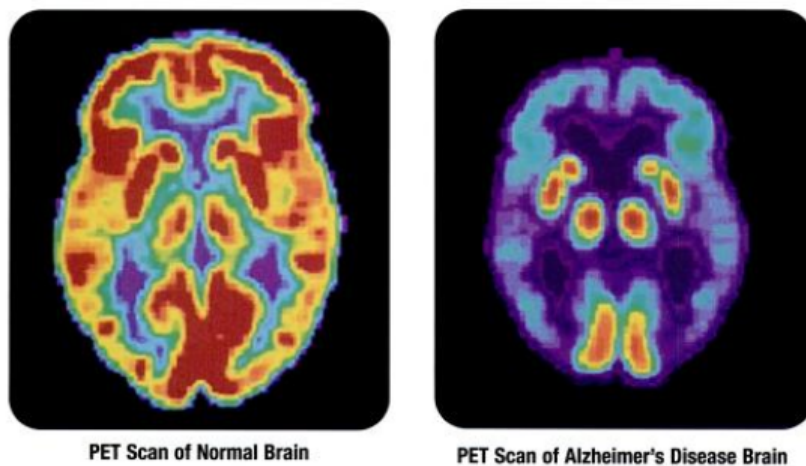


Figure 1.2: PET scans for detecting amyloid-beta plaques. The figure on the left shows the scan of a healthy brain, while the figure on the right is of a brain with AD. Figure adapted from [4].

or blocking certain receptors in the brain from excess stimulation that can damage nerve cells. The medication does not stop or reverse the damage of brain cells. Treatment without medication is available, including memory training, aerobic and nonaerobic exercise, special lighting to lessen sleep disorders, and music-related treatment [2]. There exist no treatments that would reverse the death of brain cells, so early diagnosis is crucial [1].

Several tests can be performed to determine if a person has AD. The tests include cognitive and memory tests, neurological function tests, blood and urine tests, PET-CT- or MRI-scan of the brain, and genetic testing [1]. To get an accurate diagnosis, doctors may also talk to friends and family about a patient's behaviour and take tests to rule out other impairment causes [7]. Detecting amyloid-beta abnormality is standard for diagnosing AD and is done by analyzing the cerebrospinal fluid (CSF). PET and MRI scans can show structural changes in the brain, amyloid-beta density, and neural tissue metabolism activity [5].

Risk factors for developing Alzheimer's Dementia are mainly related to ageing, genetics, and family history. Although the disease is not a natural part of ageing, the percentage of people with the disease increases with age [2]. Research has shown an increase in the risk of developing AD related to the apolipoprotein-e4 (APOE-e4) gene. The APOE gene exists in three forms (alleles) - e2, e3, and e4. Everyone has a pair of these forms, one inherited from each parent and resulting in six different combinations of the APOE pairs, e2/e2, e2/e3, e2/e4, e3/e3, e3/e4 and e4/e4. Researchers have found that having the e4 form increases the risk of developing

AD compared to having the e3 form and that the e2 form may decrease the risk of developing AD [2]. People with a first-degree relative with AD have a higher risk of developing AD. Although genetics is a part of this, other lifestyle habits within a family, such as diet, could also impact this [2]. Other risk factors include individual lifestyle factors, traumatic brain injuries, and exposition to environmental contaminants as toxic metals, pesticides, or industrial chemicals [1]. A connection of developing AD can also be found with cardiovascular diseases. Many factors that increase the risk of cardiovascular disease can be related to a greater risk of developing Dementia. A healthy heart provides the brain with enough blood, and healthy blood vessels ensure that the blood is oxygen- and nutrient-rich [2].

In 2020, over 50 million people worldwide lived with Dementia, and the number is expected to increase due to better healthcare and people getting older [8]. The economic cost of Dementia impacts the family around the person who develops it and the healthcare system. The existing cost is related to the diagnostics and informal care and the social and medical care needed. In 2015 the cost of Dementia worldwide was estimated to be 818 billion US dollars a year. The annual cost is now above 1 trillion dollars [8].

1.2 Previous work

Because early diagnosis can dramatically improve a patient's life, many studies on Alzheimer's disease using machine learning methods exist. This includes classification of people with AD and machine learning methods used to search for biomarkers for AD.

Machine learning methods can explore unknown patterns using medical images[9]. As mentioned, CSF, PET- and MRI-scan can detect amyloid-beta abnormality and brain changes. As these methods are comprehensive and expensive, cheaper alternatives are in demand. One such alternative is a study that uses retinal images that investigates the retinal vasculature for extracting potential biomarkers [5]. Their study focus on using machine learning methods for identifying links between the retinal vasculature and AD, and showed that using machine learning methods could be an efficient solution for cheaper AD screening [5]. Another study uses functional magnetic resonance imaging (fMRI) combined with a convolutional neural network (CNN) to differentiate fMRI signals from healthy people, people with MCI, and AD [10]. The authors of this paper hope to investigate further the possibilities of developing fMRI-based biomarkers for AD and MCI [10].

A study using machine learning models for differentiating between AD and vascular dementia (VD) showed that machine learning combined with volumetric measurements derived from structural MRI was a useful approach [9].

There also exist studies that extend beyond classification problems. In the lack of tools predicting individual progression to Dementia, a study uses a trajectory modelling approach to determine predictive and interpretable markers of individual variability in progression to Dementia due to AD [11]. Although not all individuals with MCI develop Dementia, the disease can also remain stable, and the study aims to create a model that discriminates between stable and progressive MCI. When predicting an individualized rate of future cognitive decline, their model showed better performance when trained on biological data than cognitive data [11]. The biological data contain measurements of amyloid-beta burden, grey matter density and APOE 4, while the cognitive data contain information about memory, executive function and affective measurements. The study also showed that the grey matter score was a highly predictive feature for classifying stable MCI vs. progressive MCI [11]. The paper refers to grey matter density from MRI scans, beta-amyloid burden from PET scans, and APOE-4 status as well studied biomarkers of AD. Another study uses an unsupervised machine learning algorithm for simulating detailed patient trajectories to train a model for individual forecasting of disease progression [12].

Although there exist different models and purposes behind the analysis of AD, the studies all face some of the same challenges because of clinical data. For example, clinical data often contain data from multiple sources derived from different measurement instruments and may have missing values [12]. Thus, for a broader implementation of machine learning methods in precision medicine, methods that can overcome these challenges must be developed.

In this thesis, block-wise analysis will be used to see if more information can be extracted from the data to improve model performance locally for the blocks. The separation of features into groups could result in an increase of patients with no missing values. The models will be based on classification between people with the presence of amyloid-beta, and people without. As mentioned above, the presence of amyloid-beta does not necessarily imply that the person will develop Alzheimer's dementia, but they are candidates that are more likely to develop AD than people without.

1.3 Problem statement

Although there are several risk factors related to the development of Alzheimer's disease, scientists don't fully understand what causes the disease in most people [13]. This promotes the need for a better understanding of risk factors related to the disease. Even though there exists no treatment against the disease, early diagnosis can be beneficial [7]. As mentioned, there exist methods that could slow down the decline in memory and other cognitive skills, which will be most beneficial in the early stages of the development of the disease. The problem statement for this thesis will be to analyse patient data to look for risk factors related to Alzheimer's Disease, and factors that could work as biomarkers for early detection of the disease.

1.4 Structure of thesis

The thesis will start with theory regarding machine learning and the different techniques used, which is described in Ch. 2. The data is described in Ch. 3, while Ch. 4 gives a description of how the data is preprocessed and how the methods are applied to the data. Results from the analysis are presented in Ch. 5 and then further discussed in Ch. 6. A final conclusion of the analysis and this thesis is given in Ch. 7.

Chapter 2

Theory

Data analysis is the process of using different methods to look for information and patterns in the data. The data analysis process includes collecting the data, cleaning the data, modelling the data, interpretation, and visualisation of results. There exist different types of data analysis methods, and the primary methods are text analysis, statistical analysis, diagnostic analysis, predictive analysis, and prescriptive analysis. **Machine learning** is a type of data analysis where algorithms can identify patterns and make decisions by learning from the data. Machine learning is based on understanding how the human brain works to create Artificial Intelligence (AI) that is deployed to solve real-world problems. The human brain consists of connected nerve cells, neurons that process and transmit electrical signals. The first machine learning algorithm, the Perceptron, is a simplified model of such a neuron and is based on how a neuron fires a signal dependent on if an accumulated value goes over a threshold value. The first concept of this algorithm was published in 1957 by Warren McCulloch and Walter Pitts [14]. Machine learning is a subfield of AI, and Deep Learning is a subfield of Machine learning. Fig. 2.1 gives an illustration of how the terms are connected.

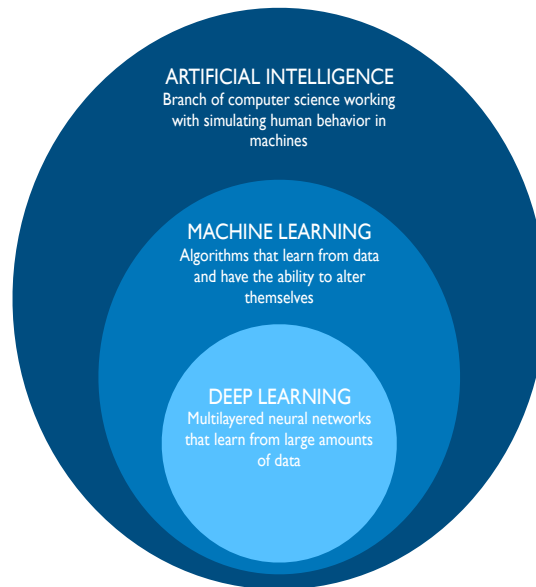


Figure 2.1: Illustration of how the terms used in the field of data science are connected.

Through an iterative process, machine learning algorithms gradually improve their performance and can be applied to analyse large amounts of data. In machine learning the variables in the dataset are called **features** and the response variables are called **targets**. The different types of Machine learning are listed below:

- Supervised learning: The model can train itself with already known answers, labelled data, and then be used on new data for prediction. The models receive direct feedback in supervised learning.
- Unsupervised Learning: There are no labels, and the model tries to find hidden structures in the data without direct feedback. Subfields of unsupervised learning include clustering of data and dimensionality reduction of the data.
- Reinforced Learning: Training an agent (system) that learns from interactions with its environment. A reward is given based on the interaction, and the agent tries to maximise the reward.

In this thesis supervised learning will be used, and it consists of the subcategories **classification** and **regression**. Classification is used for predicting categorical class labels and can be used for binary classification or multiclass classification. A commonly used example for binary classification is the task of separating email by spam or non-spam. Regression is used for predicting a target that has a continuous value [14]. Some classification models can produce continuous values in the form of probability that a sample belongs to one of the classes [15].

Chapter structure

The theory chapter will start with describing the steps involved in preprocessing the data, in Sec. 2.1. Sec. 2.2 describes methods for exploratory analysis of the data, including correlation, Principal Component Analysis (PCA) and Partial Least Squares Regression (PLSR). Sec. 2.3 describes the models used for classification, and Sec. 2.4 goes into how to evaluate a model. Feature selection with Repeated Elastic Net Technique (RENT) will be described in Sec. 2.5. Sec. 2.6 will go into the details about creating ensembles of several models.

2.1 Preprocessing

Preparing the data for analysis consists of several steps. Cleaning the data is essential to ensure good quality of the data and includes removing data that is incomplete, duplicate values, or values that are irrelevant for the analysis. Different techniques can be applied to modify the data instead of removing it [16]. The dataset may include missing values, and there are different ways of handling this. Columns or samples can be removed if they contain several missing values, or they can be handled with data imputation. This means replacing the missing value with, e.g., the mean, median, or most frequent value of the given feature. The mentioned examples are simple ways of imputing missing values, and more advanced techniques exist.

2.1.1 Data transformation

Data transformation is another step in preprocessing the data, which is the process of transforming the data into an applicable form. Transforming the data can improve the performance of a model [17]. One way of transforming the data is through scaling the data to limit the value range. Normalisation is one way of scaling the data and limits the value to be in the range from 0 to 1. This is done using the equation:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (2.1)$$

Where X' is the new value for the sample, X_{min} is the minimum value of that feature and X_{max} is the maximum value. Another way of scaling the data is through standardisation. With standardisation, the values are scaled so that they are centred around the mean with a unit standard deviation. The values are not restricted to a particular range.

This is achieved by subtracting the mean value, μ , and dividing by the standard deviation, σ as shown in equation:

$$X' = \frac{X - \mu}{\sigma} \quad (2.2)$$

Whether to normalise or standardise depends on the problem and the data, and it could be useful to test both. Both techniques are useful when working with measurements that differ in range. Standardisation does not give a limited range for the values, and the operation will not affect outliers. Another type of data transformation is power transformations, which are techniques that use a power function to make the probability distribution of a feature more Gaussian-like [17]. This removes distributional skewness in a feature. One method of power transforming the data is Box-Cox transformation [18], which is defined by the equation:

$$y^\lambda = \begin{cases} (y^\lambda - 1)/\lambda, & \text{if } \lambda \neq 0. \\ \log(y), & \text{if } \lambda = 0. \end{cases} \quad (2.3)$$

where y is a list of strictly positive numbers and λ is a hyperparameter used to control the nature of the transformation [17]. Another method of power transformation is Yeo-Johnson transformation [18] and can be used without the restriction of positive numbers. The Yeo-Johnson transformation is defined with the equation:

$$\psi(\lambda, y) = \begin{cases} ((y + 1)^\lambda - 1)/\lambda, & \text{if } \lambda \neq 0, y \geq 0. \\ \log(y + 1), & \text{if } \lambda = 0, y \geq 0. \\ -[(-y + 1)^{2-\lambda} - 1]/(2 - \lambda), & \text{if } \lambda \neq 2, y < 0. \\ -\log(-y + 1), & \text{if } \lambda = 2, y < 0. \end{cases} \quad (2.4)$$

Where ψ is the new value for the sample and a function of λ and y . λ is a hyperparameter, chosen to give the best approximation of a normal distribution. If the values of y are strictly positive, the transformation becomes the BoxCox transformation.

2.1.2 Data reduction

When dealing with a large dataset, a reduction of the data can be useful. Data reduction preserves the information in the data set while reducing the volume. Three basic methods used for reducing the data are dimensionality reduction, numerosity reduction, and data compression. Dimensionality reduction reduces the number of features in the data set, and different techniques can be used. One of the techniques is **Principal Component Analysis** (PCA), which is discussed more in detail in Sec. 2.2.3.

2.1.3 One-hot-encoding

Features in a dataset can consist of different types of data. Numerical features can be both discrete and continuous values. A feature can also consist of categorical features, grouping information with similar characteristics, and containing a finite number of groups. An example of a categorical feature could be gender. If the dataset contains categorical features, these should be converted to numerical values. Most algorithms require input values to be numerical. A method of converting data is One Hot Encoding, which gives each categorical value a new column. It then assigns binary values of 0 and 1 to these columns. Figure 2.2 gives an illustration with the example of a feature: gender, with categories male and female. On the left is the original dataset, and on the right is the dataset after One-hot-encoding. One-hot-encoding is useful in situations where the categorical features don't have any way of being ordered in numerical ranking [19].

<u>Original</u>	<u>One-hot-encoded</u>																												
<table border="1"><thead><tr><th>Gender</th></tr></thead><tbody><tr><td>Female</td></tr><tr><td>Male</td></tr><tr><td>Male</td></tr><tr><td>Female</td></tr><tr><td>Female</td></tr><tr><td>Male</td></tr></tbody></table>	Gender	Female	Male	Male	Female	Female	Male	<table border="1"><thead><tr><th>Gender</th><th>Female</th><th>Male</th></tr></thead><tbody><tr><td>Female</td><td>1</td><td>0</td></tr><tr><td>Male</td><td>0</td><td>1</td></tr><tr><td>Male</td><td>0</td><td>1</td></tr><tr><td>Female</td><td>1</td><td>0</td></tr><tr><td>Female</td><td>1</td><td>0</td></tr><tr><td>Male</td><td>0</td><td>1</td></tr></tbody></table>	Gender	Female	Male	Female	1	0	Male	0	1	Male	0	1	Female	1	0	Female	1	0	Male	0	1
Gender																													
Female																													
Male																													
Male																													
Female																													
Female																													
Male																													
Gender	Female	Male																											
Female	1	0																											
Male	0	1																											
Male	0	1																											
Female	1	0																											
Female	1	0																											
Male	0	1																											




Figure 2.2: Illustration of how One-hot-encoding changes the categorical feature into numerical values.

2.2 Exploratory analysis

Through descriptive statistics and visualisation of the distribution of values in a feature, it is possible to perform an initial analysis of the data. Exploratory analysis of the data can consist of several techniques to analyse the data for patterns or anomalies.

Descriptive statistics gives information about the values in the data. For features containing continuous data, the distribution can be numerically described with different statistical measures. The mean and median for the different features will give information about the centres of the data, the central tendency. Plotting the distribution in a histogram will provide information about how the values are spread [20].

2.2.1 Correlation

With multiple features in a dataset, it could be of interest to obtain information about the relationship between the features. The covariance between two features describes the direction of the linear relationship between the two features and can indicate how these features change together. The covariance σ_{jk} between two feature vectors, x_j and x_k , can be calculated with the equation [14]:

$$\sigma_{jk} = \frac{1}{n-1} \sum_{i=1}^n (x_j^i - \mu_j)(x_k^i - \mu_k) \quad (2.5)$$

where μ_j and μ_k are the sample means of features j and k , and n is the number of samples. x_j^i is the value of sample i for feature j and x_k^i is the value of sample i for feature k . The values for the covariance are not standardised. This can make it difficult to determine the strength of the relationship between the two features. By dividing the covariance with the standard deviations of the features, the value for the correlation coefficient can be computed. Correlation measures whether the two variables systematically vary together, and therefore describes the strength of the relationship between two features. This is shown mathematically with the equation:

$$CORR(x_j, x_k) = \frac{\sigma_{jk}}{\sigma_{x_j} \sigma_{x_k}} \quad (2.6)$$

Where σ_{x_j} and σ_{x_k} are the standard deviations of the features x_j and x_k . The values for the correlation coefficient range from -1 to 1 . The closer the value is to either -1 or 1 , the more closely the two features are related. When the coefficient is positive, an increase in one feature means an increase in the other feature. When the coefficient is negative, an increase in one feature means a decrease in the other feature, meaning the changes in the two features are inversely related. The correlation coefficient will be 0 if there is no relationship between the two features [21]

The value for the correlation coefficient between the features in a dataset can be plotted in a heat map. This will give visual information about which features are the most correlated or anti-correlated. This method is used in this thesis, and the heat maps are presented in Sec. 5.2.1.

2.2.2 RV matrix correlation

When working with data with many features or if the data comes from different sources, it can be helpful to divide the data into groups of similar measurements.

It could be of interest to look into the overlapping information between the blocks. As with correlation between features, it is possible to calculate the correlation between blocks. This is done by calculating the RV matrix correlation coefficients between pairs of arrays and the coefficient represents a measure of common information. Given two matrices \mathbf{X} and \mathbf{Y} , one expression for the RV matrix is [22]:

$$RV(\mathbf{X}, \mathbf{Y}) = \frac{Vec(\mathbf{X}\mathbf{X}^T)^T Vec(\mathbf{Y}\mathbf{Y}^T)}{\sqrt{Vec(\mathbf{X}\mathbf{X}^T)^T Vec(\mathbf{X}\mathbf{X}^T) \times Vec(\mathbf{Y}\mathbf{Y}^T)^T Vec(\mathbf{Y}\mathbf{Y}^T)}} \quad (2.7)$$

Where $Vec(\mathbf{X})$ is the vectorised version of the matrix \mathbf{X} . The value for the correlation coefficients is between 0 and 1, where 1 is perfect correlation. For matrix correlations for high-dimensional data, the modified RV-coefficient can be used. The modified RV-coefficient is also called RV_2 -coefficient and ignores the diagonal matrices. The same formula can be used if replacing: $\mathbf{X}\mathbf{X}^T$ with $\mathbf{X}\mathbf{X}^T - diag(\mathbf{X}\mathbf{X}^T)$. The matrix $diag(\mathbf{X}\mathbf{X}^T)$ contains only diagonal elements, the diagonal elements of $\mathbf{X}\mathbf{X}^T$, and zero's elsewhere [22]. For the modified coefficient, the values can range from -1 to 1 , which means it also can measure negative correlation.

2.2.3 Principal Component Analysis

Principal components analysis (PCA) is an unsupervised, non-parametric technique for exploratory data analysis. PCA can be used to describe the variability in a dataset and reduce the dimensions of a dataset while preserving systematic information by the use of principal components [23]. Reducing the dimensions is done by **feature extraction**, where the extracted features are the principal components and are formed by linear combinations of the original features. PCA projects the data onto a new subspace by looking for the directions with the most variance in the data. The subspace can have equal or fewer dimensions than the original subspace of the data. Finding the most variance is done by finding patterns based on the correlation between features [14].

PCA works as follows: for a d -dimensional feature vector, \vec{x} , given as:

$$\vec{x} = [x_1, x_2, \dots, x_d] \quad (2.8)$$

the first step is to standardize the data using Eq. (2.2), because of the sensitivity to scaling for the direction of the principal components. The next step is to construct the covariance matrix, which is a square $d \times d$ matrix, where d is the dimension of the feature vector. The covariance between two features is discussed above in Sec. 2.2.1 and is calculated using Eq. 2.5. The covariance matrix can then be

constructed by calculating the covariance between the different features. If $d = 3$, the covariance matrix would become:

$$\Sigma = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix} \quad (2.9)$$

Where σ_{12} is the covariance for $j = 1$ and $k = 2$. The covariance matrix is then decomposed into its eigenvectors and eigenvalues using eigenvalue decomposition [24]. The eigenvectors indicate the direction of maximum variance, which corresponds to the principal components. The elements of an eigenvector are the weight coefficients for each of the features from the original data. The weight coefficients are also known as loadings. The corresponding eigenvalues represent the magnitude of the eigenvector. For an eigenvector, \vec{v} , with the eigenvalue, λ , the following condition is satisfied [14]:

$$\Sigma \vec{v} = \lambda \vec{v} \quad (2.10)$$

For dimensionality reduction, the subset of eigenvectors that contributes to most of the variance should be selected and the eigenvalues can be used to calculate the explained variance. The explained variance ratio for a given eigenvalue λ_j is shown in the equation:

$$\text{Explained variance ratio} = \frac{\lambda_j}{\sum_{j=1}^d \lambda_j} \quad (2.11)$$

which is the fraction of the given eigenvalue divided by the total sum of all the eigenvalues. The eigenvalues give the magnitude of the eigenvectors and can be sorted in decreasing order. The eigenvectors are sorted in corresponding order as the eigenvalues. From this, the k top corresponding eigenvectors can be chosen to represent the new feature subspace, where k is the dimension of the new feature subspace. For dimensionality reduction, k must be smaller than d . To find an appropriate value for k , it is possible to plot the explained variance ratio as a function of the number of principal components. It is ideal to select the subset of eigenvectors that contains most of the information in the data, which is the same as choosing the number of principal components that most of the variation. The next step is to construct a transformation matrix, \mathbf{W} , which consists of the top k eigenvectors. Then the original data set can be transformed using this matrix to get the new feature subspace, as shown in the following equation:

$$\mathbf{X}' = \mathbf{X}\mathbf{W} \quad (2.12)$$

2.2.4 Partial Least Squares Regression

Regression is a statistical method for analyzing the strength and character of the relationship between features. With simple regression analysis, this is the relationship between a single explanatory feature and a response, usually denoted with x and y , respectively. The task of regression analysis is to produce an estimate of the features involved in the analysis based on previous information [25]. For simple linear regression, this means finding the model that best describes the relationship between the features. The regression model can be described with the equation:

$$y = a + bx \quad (2.13)$$

where a is the intercept of the best-fitted line and b is the slope of the line. The individual sample's deviation from the line is called the residuals, and finding the best-fitted line to the measurements is done by finding a line that minimizes the sum of the squared residuals. Figure 2.5 shows an example of a regression problem for illustration. The red line is the regression line, the black dots are the samples, and the black lines show the residuals.

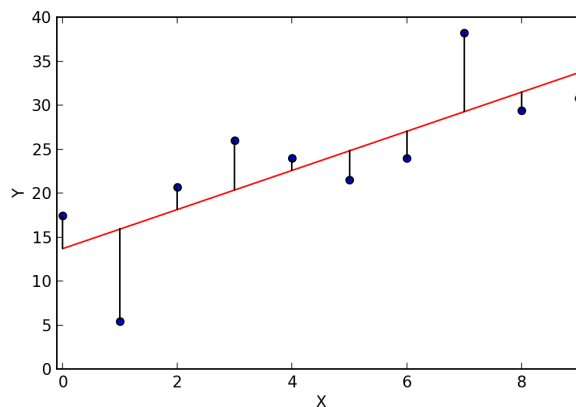


Figure 2.5: Example of a regression problem. The red line is the regression line. The samples are shown as black dots and the residuals as black lines. Figure adapted from [26].

The line that minimizes the sum of the squared residuals is called the least-squares line, and the slope is calculated with the equation:

$$b = \frac{SS_{xy}}{SS_{xx}} \quad (2.14)$$

Where SS_{XY} is the sum of the cross products and SS_{XX} is the sum of the squares for variable x . The intercepts can be found by rearranging Eq. 2.13 and using the average values for y and x .

With multiple regression analysis, we are dealing with several explanatory features, $x_1 \dots x_n$, to describe the target feature. The explanatory features are independent features, and the response is a dependent feature. When dealing with two explanatory features, the task is no longer to find the best-suited line but the best-suited plane. The best-suited plan can be found by minimizing the distance between the value of y and the estimated plane. This can be extended to the general concept that when dealing with n explanatory features, the task is to find a n dimensional best-suited plane [25].

Partial Least Squares Regression (PLSR) is a supervised technique for exploratory data analysis. Similar to PCA, it projects the data onto a new subspace, but it differs from PCA because it projects both the X and Y data. The following equations can describe a PLSR model [27]:

$$\begin{aligned} \mathbf{X} &= \mathbf{TP}^T + \mathbf{E} \\ \mathbf{Y} &= \mathbf{UQ}^T + \mathbf{F} \end{aligned} \quad (2.15)$$

\mathbf{X} is the data matrix with dimension $n * m$ and \mathbf{Y} is the response matrix with dimension $n * p$. T and U are matrices with dimensions $n * l$ and are the scores matrices for \mathbf{X} and \mathbf{Y} . \mathbf{P} and \mathbf{Q} are the loadings matrices for \mathbf{X} and \mathbf{Y} and have dimensions $m * l$ and $p * l$. \mathbf{E} and \mathbf{F} are residual error matrices.

A PLSR model (requires) solving an optimization problem under the restrictions: $\vec{t} = \mathbf{X}\vec{w}$ and $\vec{u} = \vec{t}\vec{c} = \mathbf{X}\vec{w}\vec{c}$. The models optimisation task is to maximize $(\vec{t}^T \vec{u})$ subjected to $\vec{t}^T \vec{t} = \vec{w}^T \vec{w} = \mathbf{I}$, which means maximizing the covariance of the vectors \vec{t} and \vec{u} . The regression coefficient matrix can be calculated with equation:

$$\mathbf{B}^{PLS} = \mathbf{W}(\mathbf{P}^T \mathbf{W})^{-1} \mathbf{CQ}^T \quad (2.16)$$

The predicted value can then be found with equation:

$$\hat{\mathbf{Y}} = \mathbf{TCQ}^T = \mathbf{XB}^{PLS} \quad (2.17)$$

2.3 Models for classification

As the problem studied in this thesis is a binary classification problem, this section describes models used for classification. Classification is a type of supervised learning, where the model learns from labelled data.

As mentioned at the beginning of this chapter, machine learning is inspired by how the biological brain works and how the signals are transmitted through neurons in the brain. The first machine learning algorithm, the Perceptron, is a model with a single neuron and is used for binary classification [14]. Figure 2.6 illustrates the concept behind the Perceptron. The network receives input values x_1 to x_n . The input values are then linearly combined with the weights, w_{1j} to w_{nj} , to form the net input z . The net input is referred to as net_j in the figure.

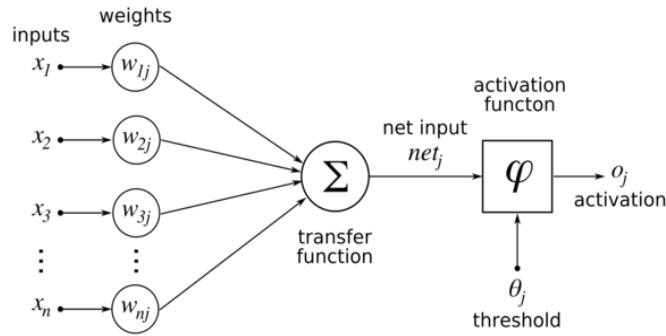


Figure 2.6: Illustration of the concept behind the single-neuron Perceptron algorithm. Figure adapted from [28].

The net input is used as input to the activation. The net input can be calculated with equation [14]:

$$z = \vec{w}^T \vec{x} = w_0x_0 + w_1x_1 + \dots + w_nx_n \quad (2.18)$$

w_0 and x_0 represents the bias. Which class the sample belongs to is decided by the value of the net input and a threshold value, θ . The activation function for the Perceptron is called a unit step function. Defining the bias unit as $w_0 = -\theta$ and $x_0 = 1$, the function can be described with equation:

$$\phi(z) = \begin{cases} 1, & \text{if } z \geq 0, \\ -1, & \text{otherwise.} \end{cases} \quad (2.19)$$

The network learns by updating the weights, which happens after computing the output value, \hat{y} , denoted with o_j in the figure. The weights are initially zero or small random numbers and each weight, w_j , in weight vector \vec{w} is updated simultaneously:

$$w_j := w_j + \delta w_j \quad (2.20)$$

Where δw_j is calculated with the equation:

$$\delta w_j = \eta(y^{(i)} - \hat{y}^{(i)})x_j^{(i)} \quad (2.21)$$

where i indicates the i th samples, η is the learning rate, y is the actual class label, and \hat{y} is the predicted class label. The Perceptron model will only converge if the classes are linearly separable [14]. Another single-layer neural network is the Adaptive Linear Neuron (Adaline), which lays the foundation for understanding more complex networks and classification models. The Adaline model illustrates the key concept of defining and minimizing continuous cost functions [14]. The model uses a linear activation function for updating the weights, which is described with the equation:

$$\phi(z) = z \quad (2.22)$$

After the activation function, a threshold function is used to make the final prediction. While the Perceptron compares the actual class label to the predicted class label to update the weights, the Adaline compares the actual class label with the continuous output from the linear activation function [14]. To learn the best-suited parameters, a cost function can be used, where the goal for the model is to minimize this cost function. In Adaline, the cost function is defined as:

$$J(\vec{w}) = \frac{1}{2} \sum_i (y^{(i)} - \phi(z^{(i)}))^2 \quad (2.23)$$

Showing that the Adaline algorithm learns the weights as the sum of squared errors (SSE) between the actual class label and the calculated outcome. An algorithm called gradient descent is used to find the weights that minimize the cost function. The weights can be updated by taking a step in the opposite direction of the gradient to the cost function for a given \vec{w} . The change in weights is calculated by taking the negative gradient multiplied by the learning rate, and the gradient of the cost function is calculated using the partial derivative of the cost function with respect to each weight in \vec{w} [14].

The update of weight can be described mathematically with the equation:

$$\Delta w_j = -\eta \frac{\partial J}{\partial w_j} = \eta \sum_i (y^{(i)} - \phi(z^{(i)})) x_j^i \quad (2.24)$$

Figure 2.7 illustrates how the gradient descent algorithm works by using an example where the data is fitted to a straight line. The figure on the left shows the straight line as black x's and the fitted lines with different values for the weight. The initial weight is zero and the orange line is the start. By gradually increasing the weights, the line gets closer to the black line. The figure on the right shows the cost function. The orange dot represent the start and using the gradient the dots move closer towards the global minimum, the minimum cost.

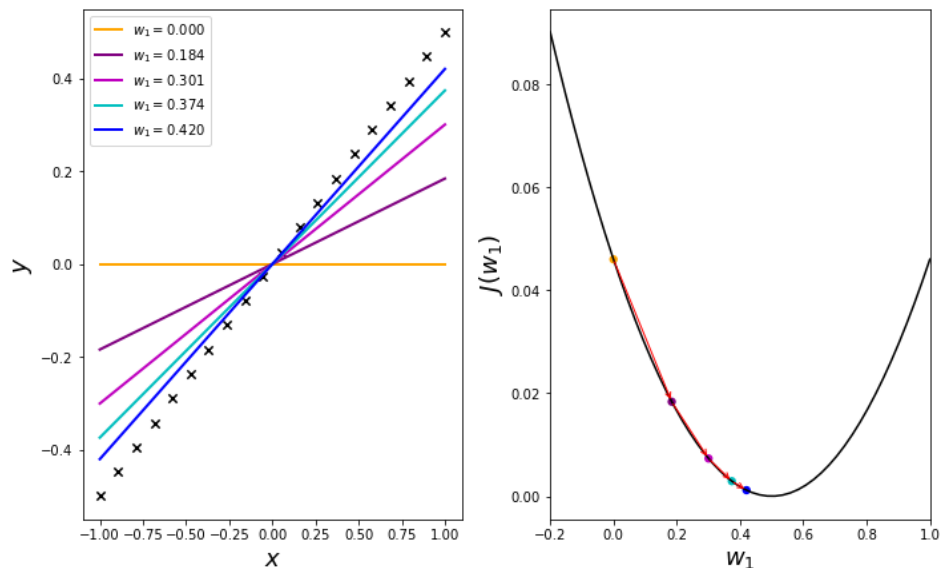


Figure 2.7: Illustration of how the cost can be minimized using gradient descent. The figure on the left shows the data and the fitted lines, and the cost function is shown in the figure on the right.

Building on these fundamental concepts of machine learning, the different classifiers used in this thesis will be described in the following sections.

2.3.1 Logistic Regression

Logistic Regression is a linear model used for binary classification. The model performs well for linearly separable classes and is a widely used algorithm for classification [14]. The main difference between Logistic Regression and the previously described Adaline algorithm is the activation function, ϕ . Logistic Regression uses an activation function called logistic sigmoid function, which is defined as:

$$\phi(z) = \frac{1}{1 + e^{-z}} \quad (2.25)$$

Where z is the net input to the activation function described with Eq. 2.18. Figure 2.8 shows the sigmoid activation function. It maps the net input to values in the range of $[0, 1]$, describing the sample's probability of belonging to one of the two classes.

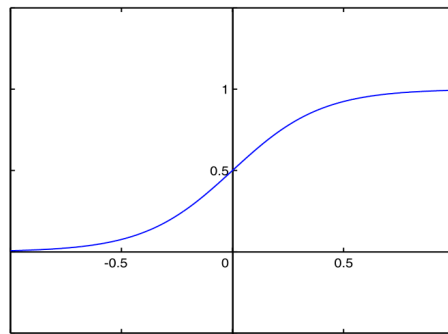


Figure 2.8: Graphic representation of the sigmoid activation function used in Logistic Regression. Figure adapted from [29].

A threshold function is used to turn the probability into binary values. For Logistic Regression this is mathematically described:

$$z = \begin{cases} 1, & \text{if } \phi(z) \geq 0.5. \\ 0, & \text{otherwise.} \end{cases} \quad (2.26)$$

Logistic Regression is a probabilistic algorithm, and the cost function for learning the weights is based on the log-likelihood function. The cost function can be written as:

$$J(\vec{w}) = - \sum_i y^{(i)} \log(\phi(z^{(i)})) + (1 - y^{(i)}) \log(1 - \phi(z^{(i)})) \quad (2.27)$$

The gradient descent algorithm can be used for optimizing the cost function. The negative sign comes from the fact that we want to maximize the probability by minimizing the loss. If the cost decreases, the maximum likelihood assuming that samples are drawn from an identically independent distribution will increase [?].

2.3.2 Support Vector Machine

The Support Vector Machine (SVM) algorithm can be thought of as an extension of the Perceptron and can be used for regression and classification tasks, but is widely used for classification problems [30]. When looking at a classification problem, the algorithm works to find a hyperplane that distinctly separates the two classes, also called the decision boundary. Given a decision boundary that separates positive and negative samples, we then have a positive- and negative hyperplane. The distance between the positive- and negative hyperplane is known as the margin, and the SVM algorithm works to maximize this margin, finding the hyperplane with the greatest margin. With a large margin, the models tend to have lower generalization errors [14].

Figure 2.9 illustrates the concept behind the algorithm. The optimal hyperplane is illustrated as a red line, with the margin at each side as dotted lines. In this case, we are trying to separate the blue and the green samples, which we could call the blue- and green hyperplane. The support vectors are the samples closest to the decision boundary.

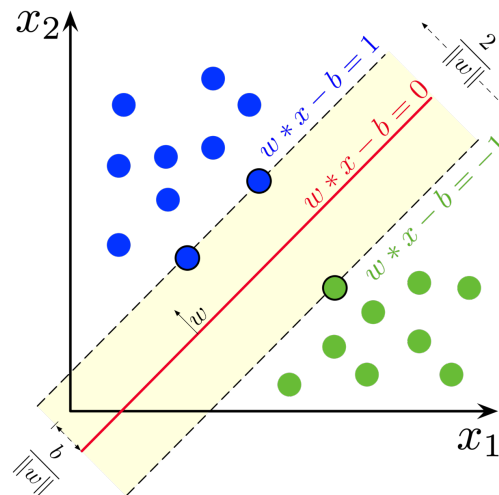


Figure 2.9: Illustration of the optimal hyperplane, support vectors, and the margin used in the SVM algorithm. Figure adapted from [31].

As shown in the figure, the positive- and negative hyperplanes are parallel to the optimal hyperplane. The objective function of the SVM becomes the maximization of the distance between the positive- and negative hyperplane under the constraint the samples are correctly classified [14]. Wanting all the positive samples to fall on the side of the positive hyperplane and the negative samples to fall on the side of the negative hyperplane can be described mathematically as:

$$y^{(i)}(w_0 + \vec{w}^T \vec{x}^{(i)}) \geq 1 \quad (2.28)$$

To handle nonlinearly separable cases, a slack variable can be added to the equation above. Another possibility is mapping nonlinear combinations of the features to a higher-dimensional space using kernel methods [14].

2.3.3 Decision Tree

Decision Tree classifiers are models that make decisions based on a series of questions to break down the data [14]. Figure 2.10 shows the structure of a Decision Tree. Starting with the root node, it splits into internal nodes where decisions are made, also called decision nodes. The final leaf nodes are called the terminal nodes since it is a final node, and no more splitting can be done. When all the samples at each node belong to the same class, the leaf node can be called a pure node [32].

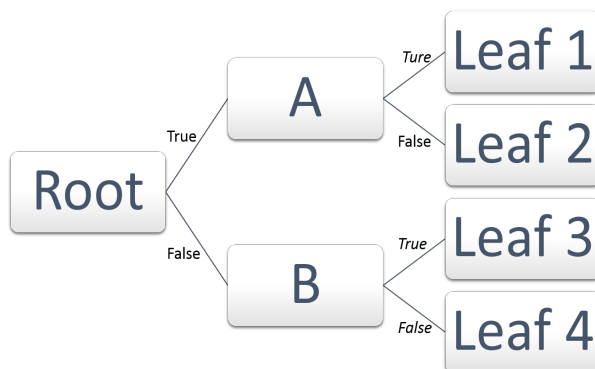


Figure 2.10: An example of a Decision tree. Starting at the root node and branches out to decision nodes. The final nodes are the leaf nodes. Figure adapted from [33].

The number of nodes at each level and the number of levels, the depth, will depend on the problem and how each split is performed. The data is split in a way that maximizes the information gain (IG), which gives information about how impor-

tant a feature is. The objective function for the Decision Tree algorithm can be described with the equation:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^m \frac{N_j}{N_p} I(D_j) \quad (2.29)$$

Where D_p and D_j are the dataset of the parent node and j th child node, f is the given feature, N_p and N_j are the number of samples at the parent node and the j th child node and I is the impurity measure. Information gain is the difference between the child node impurities added together and the impurity of the parent node [32]. A binary decision tree splits each node into two leaves and the common impurity measures, also called splitting criteria, are Gini impurity (I_G), entropy (I_H), and classification error (I_E).

2.3.4 Random Forest

Random Forest is an ensemble technique that combines multiple decision trees to form one model. By combining multiple trees, it is possible to build a model that generalizes better than one tree alone. Ensuring that there are enough trees will prevent that the model is overfitted on the training data. The algorithm for random forest can be summarized in four steps [14]:

1. Choose n samples from the training data with replacement.
2. Grow a decision tree from the n samples. At each node:
 - Select d features without replacement.
 - Split the node using the feature that maximizes the information gain.
3. Repeat steps 1-2 a given number of times.
4. Aggregate the prediction by each tree and use majority voting to assign the class label.

Figure (2.11) shows an illustration of a Random Forest model consisting of three trees, where the sample is referred to as an instance. The decision of which class a sample belongs to is based on majority voting from the predictions that each tree makes.

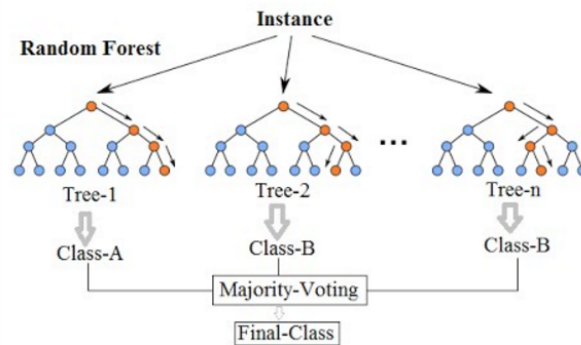


Figure 2.11: An example of how a forest can be constructed. Consisting of three decision trees. Figure adapted from [34]

By averaging or combining results from different trees, the model overcomes the issue of overfitting the data [35]. Compared to a single decision tree, Random Forest has less variance. These advantages come at the cost of more computational resources needed to implement the Random Forest model, and that prediction can be time-consuming.

2.3.5 K-Nearest Neighbors

The K-Nearest Neighbors algorithm (KNN) is different from the other algorithms discussed as it uses the data directly for classification and needs to keep all the training data. This could be an advantage because the classifier adapts as new data is collected, but this also means that the efficiency for classifying new samples decreases with the increase of data. With more data the memory computations increases. The algorithm can be summarized in three steps [14]:

1. Choose a number, k , representing the number of neighbours and a distance metric.
2. Find the k -nearest neighbours of the data to be classified.
3. Assign class label by majority voting.

The algorithm uses a chosen metric to calculate the distance between points and works with the assumption that similar things exist close to each other. There are several metrics that can be used for calculating the distance, but the Euclidean distance (straight line) is commonly used. The number of neighbours, k , is the algorithm's core deciding factor [36].

Figure 2.12 illustrates how the algorithm works. The new sample to classify is marked as a green circle, and the distance from this sample is marked with a solid and a dashed circle. k is equal to five for the outer dashed circle and 3 for the inner solid circle. For $k = 3$ the new sample, the green circle, will be predicted as the class of red triangles. For $k = 5$ it will be predicted as the class of blue squares.

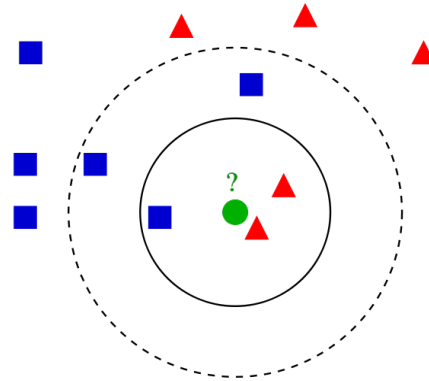


Figure 2.12: Illustration of the concept behind KNN. The circles represent the neighbourhood of the sample that is to be classified. Figure adapted from [37].

2.3.6 Passive Aggressive Classifier

There exist additional categories of Machine Learning in addition to the main categories supervised learning, unsupervised learning and reinforced learning, which was listed at the beginning of this chapter. One of them is **Online learning**, which is the category that the Passive Aggressive Classifier belongs to. With online learning, the algorithm is fitted with the training data in increments and continues to learn when new data is added [38]. Adding the data sequentially can be useful when working with a large amount of data and when used in systems that continue to receive data. The algorithm classifies labels as -1 and 1 . 0 is the classification boundary, and between -1 and 1 is a margin. The algorithm computes a loss function based on where the prediction falls. If a label is predicted correctly, the loss is zero, and the algorithm stays passive. The error becomes bigger the further away the prediction is for the correct value, and the algorithm is said to move aggressively to classify the labels [38].

2.4 Model Evaluation

When building a machine learning model, the performance must be evaluated to know if the predictions can be trusted. This will give information about how well the model generalizes on unseen data.

When a model is too simple, it will not capture the relationship between the features and target and miss important trends. This results in poor prediction performance. The model is then under-performing, also called under-fitting. High bias and low variance is a good indication of underfitting [39]. Bias error describes how much the predictions differ from the actual value, while variance gives information about how the prediction on the same sample differs from each other when the model is trained on different subsets of the data. Figure 2.13 illustrates bias and variance. The white circle middle is the actual value, and the black dots are the predictions. The figure illustrates how a model can have low bias and low variance (a), high bias (b), low variance (c) or high bias and high variance (d). An accurate model has low bias and low variance.

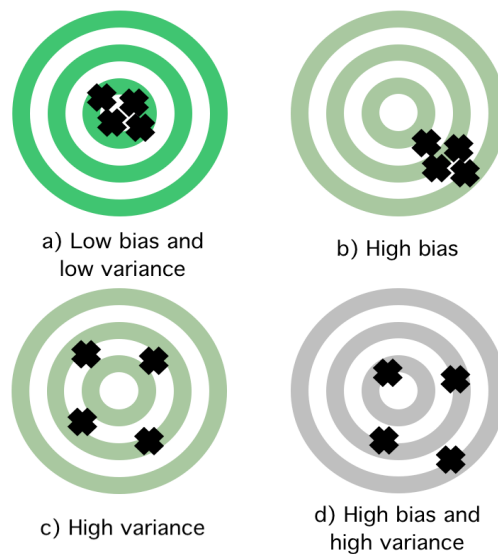


Figure 2.13: Illustration of the terms bias and variance. The white circle in the middle can be considered as the target. Figure adapted from [40].

More complex models lead to less bias and lower bias in the model will cause a reduction in error. But a model can also become too complex, overfitting the training data, which in turn leads to high variance. This will give good prediction performance on the training data used for learning but poor prediction performance on unseen data since the model does not generalize well [39]. Finding a good balance between the errors bias and variance is necessary for making a good model, and the

model needs to be evaluated to find this balance [14]. Figure 2.14 shows an illustration of a regression model that is underfitted (left), optimal (middle) and overfitted (right), where the task is to find the best-suited curve to the blue dots.

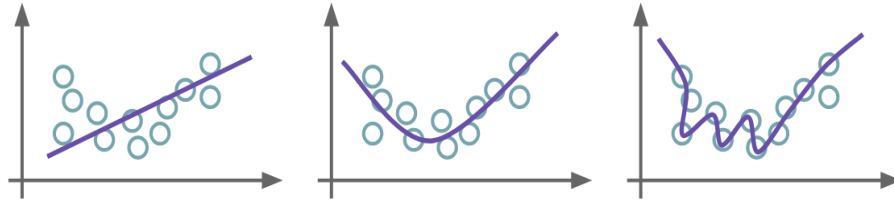


Figure 2.14: Shows the cases of a model that is underfitting, balanced and overfitting. Figure adapted from [41].

There exist different techniques for accessing the performance of a model. There are also different scoring metrics to use, and it is important to choose the right metric for the given problem and data to measure the model's performance correctly.

2.4.1 Regularisation

To handle underfitting and overfitting, finding a good bias-variance tradeoff, the model complexity can be increased or decreased respectively. A method for this is regularisation, which works to reduce the variance in a model without a considerable increase in bias [42]. With this method, a regularisation term is added to the cost function to shrink the weights during training. The regularisation term includes a regularisation parameter, λ , that decides how much we want to penalize. If it is set to zero, the regularisation term will have no effect. An increase in λ will cause a higher penalty and a reduction of the weights [42]. As the value for λ approaches infinity, the weights will approach zero. When the weights decrease, the variance will decrease, but it is essential to find the correct value [42]. There are mainly two types of regularisation techniques. One of them is **Ridge Regression**, which uses an L2 penalty function as shown in Eq. 2.30. To improve long-term performance this method introduces a small amount of bias, known as Ridge regression penalty. The technique minimizes the weights, but they don't become zero. The regularisation term for this technique is defined with the equation:

$$\lambda \sum_{j=1}^m w_j^2 \quad (2.30)$$

A constraint function can be defined when defining a constant s for each value of λ . The constraint is shown in Fig. 2.15 in turquoise colour and is a circle for Ridge regression. The Ridge regression technique can be thought of as finding a solution to an equation where the summation of the squares of the weights is less or equal to s , which becomes the constraint function [42]. The weights will have the smallest residual sum of squares (RSS) for the samples that lie within the circle given by this constraint. The other technique is called **Lasso**, and it uses an L1 penalty function. The regularisation term is defined with the equation:

$$\lambda \sum_{j=1}^m |w_j| \quad (2.31)$$

This technique takes the absolute weights instead of a square of weights, and the constraint function gets the shape of a diamond. With this technique, the weights can become zero, and it can be used for feature selection. Figure 2.15 gives a geometric interpretation of the two regularisation methods. The symbol β is used for the weights. The Lasso technique is presented on the left and the Ridge regression technique on the right, with the constraint function as the green areas. The example in the figure uses the sum of squared errors (SSE) cost function, which is the cost function used in the Adaline algorithm, described in Sec. 2.3. The same concept applies to other cost functions [14]. The contours of RSS are shown as red ellipses. In the figure, the Lasso and Ridge regression weight estimates are given by the first point where the ellipse meets the constraint region.

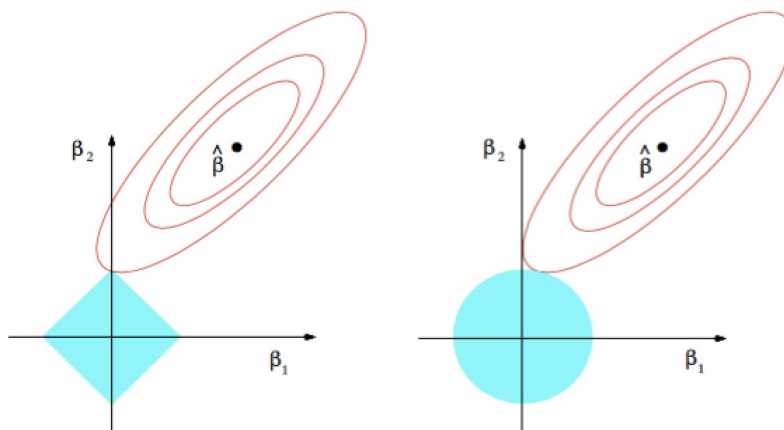


Figure 2.15: Lasso on the left and Ridge regression on the right with the constraint functions in turquoise. Figure adapted from [43].

A third regularisation method is the **Elastic Net**, which combines the L1 and the L2 penalty functions [44]. It uses an L1 penalty to produce sparsity and an L2 penalty for selecting features [14]. The regularisation term is the sum of the two previous terms, but with different values for the parameter λ :

$$\lambda_1 \sum_{j=1}^p \beta_j^2 + \lambda_2 \sum_{j=1}^p |\beta_j| \quad (2.32)$$

2.4.2 Splitting the data

Before training the model the data is split into **training** data and **test** data. The training data is used to fit the model and learn the weights. The test set is used to evaluate the already fitted model to see how it performs on unseen data. A portion of the training data can be used as **validation** data. Validation data can be used for evaluating the model while tuning the hyperparameters. Hyperparameters are discussed in the following section. The final evaluation of the model is done on the test data. The ratio of the data to use in each set will depend on the size of the dataset and the model used [14].

2.4.3 Hyperparameters

In Sec. 2.3 a theoretical description of the different models for classification was given. In practice, these models have hyperparameters that can be adjusted. The hyperparameters are not the parameters that are derived through training of the model, the weights. Instead, the hyperparameters control the learning process. An example of a hyperparameter is k , the number of neighbours used in the KNN algorithm. To find the best-suited hyperparameter, the model must be evaluated for different values to see which gives the highest value for the selected scoring metric.

2.4.4 Cross-validation

Cross-validation is a way of evaluating a model by training the model on part of the data and evaluating the performance on unseen data. Cross-validation can be used for finding the best suitable hyperparameters, optimizing the hyperparameters. It can also be used for comparing several different models. Cross-validation solves the problem of the reduction of data due to the previously mentioned splitting. The test data is left out for the final evaluation, but the training data will be divided into smaller partitions that will be used to evaluate the model. There exist different types of cross-validation, but a detailed description will only be given for the method used in this thesis.

K-fold cross-validation

K-fold cross-validation is a commonly used cross-validation technique where the training data is iterated through k times [14]. The training data is split into k parts in each iteration. $k - 1$ parts are used for training and the last for validation. Figure 2.16 shows an illustration of the process with $k = 4$. The samples are coloured in red and green, given with class it belongs to. The samples used for validation in each iteration is highlighted by a square, black rectangle. The validation samples are referred to as test data in the figure. $k = 4$ models are created and the average performance is based on these 4 models. This method can be computationally expensive but utilizes the available data to make a robust model evaluation. If there is an imbalance in the distribution of the classes, stratified k-fold cross-validation is an option to preserve the proportions of the classes in each fold, which can give better bias and variance estimates [14].

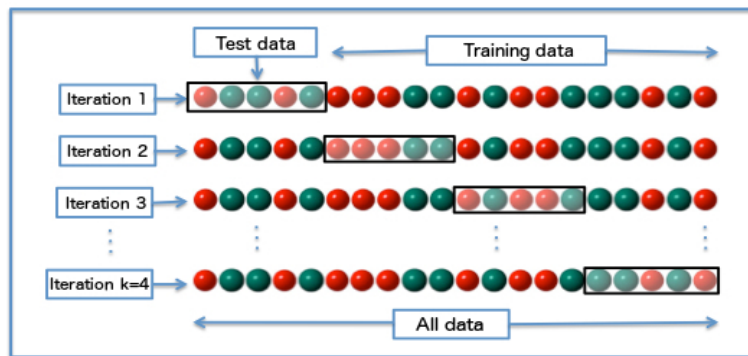


Figure 2.16: Illustration of the process behind k -fold cross-validation.
Figure adapted from [45].

2.4.5 Scoring metrics

In binary classification, the models' predictions fall into one of four categories [46]:

- True positive (TP): Actual positive that was predicted positive.
- False positive (FP): Actual negative that was predicted positive.
- True negative (TN): Actual negative that was predicted negative.
- False negative (FN): Actual positive that was predicted negative.

These categories can be visualised through a confusion matrix as shown in Fig. 2.17. The actual values are along the vertical axis, and the predicted values are along the horizontal axis.

Actual	Positive	TP	FN
	Negative	FP	TN
		Positive	Negative

Predicted

Figure 2.17: Confusion matrix illustrating the four categories of prediction.

There exists different types of metrics for evaluating the performance of a model, and it is important to choose an appropriate metric for a given problem. A frequently used metric for scoring a classifier is accuracy, which is the fraction of the correctly predicted samples of all the samples. It can have a value between 0 and 1 and is described mathematically with the equation:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.33)$$

Accuracy is useful for evaluating a model's performance if the distribution of the class labels is balanced. When dealing with cases where the class distribution is unbalanced, other metrics should be considered for evaluating performance. Precision is a class-specific metric, and defined with the equation:

$$PRE = \frac{TP}{TP + FP} \quad (2.34)$$

Precision gives the fraction between the correctly predicted positive samples, and all the samples predicted as positive. Recall is another used metric and is the fraction between the correctly predicted positive samples, and all the positive samples. It can be described with the equation:

$$REC = \frac{TP}{TP + FN} \quad (2.35)$$

Another metric is the F1-score, which is the harmonic mean of precision and recall, combining the two metrics.

$$F1 = \frac{2 (PRE) (REC)}{PRE + REC} \quad (2.36)$$

The mentioned metrics have values between 0 and 1. The last metric to be described is Matthews correlation coefficient (MCC), which is unaffected by an imbalance in the distribution of the classes. The value for MCC is in the range from -1 to 1 , where 1 is perfect classification, -1 is perfect misclassification, and 0 can be thought of as random guessing. It can be calculated with the equation [46]:

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) (TP + FN) (TN + FP) (TN + FN)}} \quad (2.37)$$

2.5 Repeated Elastic Net Technique

When working with datasets with many features, it could be beneficial to reduce the dimensions of the data. There exist different techniques for dimensionality reduction. As mentioned in Sec. 2.2.3, PCA can be used to reduce the features through feature extraction. Another way of reducing the dimensions is by **feature selection**, reducing the number of features. Reducing the number of features makes the modelling computational more efficient and reduces the chance of overfitting the data. A dataset can contain features that don't contribute to any useful information for the prediction of the target, and feature selection methods look for the features that are most useful for that purpose [47].

There are many different techniques used for feature selection and can be categorized into filter-, wrapper-, and embedded methods [48]. The method used in this thesis is Repeated Elastic Net Technique (RENT), which is an embedded approach for feature selection. The method is based on an ensemble of elastic net regularized models that are trained on separate subsets of the data [48]. The regularisation method Elastic net is described in Sec. 2.4. The separate subsets are created by sampling the original training data with replacement, creating unique subsets for each model in the ensemble. Using several models gives a more robust estimation of the importance of features by identifying how often a feature is selected among many models.

The number of models in the ensemble, K , can be adjusted by the user. Elastic Net determines which features are selected in each model. The weights for the features that are not selected are zero and non-zero for the features selected. Each trained model has a vector of feature weights, β_n , which is combined to a weight matrix, \mathbf{B} . For a N dimensional features space, the weights matrix \mathbf{B} has dimensions $(K \times N)$.

In order for a feature to be selected, RENT uses three criteria that all need to be satisfied, τ_1 , τ_2 and τ_3 . The cutoff value for each criteria can be adjusted but has to be in the interval between 0 and 1 because τ_1 , τ_2 and τ_3 can take values in this interval. τ_1 tells how often a feature is selected and is found by using the equation:

$$\tau_1(\beta_n) = c(\beta_n) \quad (2.38)$$

where $c(\beta_n)$ measures the importance of a feature through average frequency:

$$c(\beta_n) = \frac{1}{K} \sum_{k=1}^K 1, \quad [\beta_{k,n} \neq 0] \quad (2.39)$$

Finding the percentage of non-zero parameters estimates for each feature, f_n by counting how often this feature was selected on average across the models. τ_2 measures the percentage of same polarity (either positive or negative sign) among the weights that are not zero. The value for τ_2 can maximum become as large as the value for τ_1 , which only happens when all the non-zero weights have the same polarity. How often the weights have the same polarity is defined with the equation:

$$\tau_2(\beta_n) = \frac{1}{K} \left| \sum_{k=1}^K \text{sign}(\beta_{k,n}) \right| \quad (2.40)$$

τ_3 is a criteria that identifies whether the weights are consistently larger than zero. τ_3 represents the cumulative distribution function of Student's t-distribution with $K - 1$ degrees of freedom. Using the specific mean, μ , and variance σ of the feature vectors, it can be calculated with equation:

$$\tau_3(\beta_n) = t_{K-1} \left(\frac{|\mu(\beta_n)|}{\sqrt{\frac{\sigma^2(\beta_n)}{K}}} \right) \quad (2.41)$$

Figure 2.18 illustrates the procedure of feature selection of RENT, represented by the blue frame. The input training data, X_{train} , is split into subsets by sampling. The subsets are used to train the K different models. A feature is selected to form the new dataset if it satisfies all three criteria.

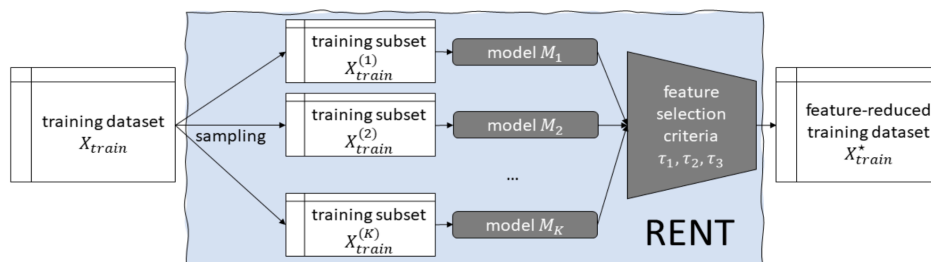


Figure 2.18: Illustration of the RENT procedure for feature selection, represented by the blue frame. Figure adapted from [48].

2.6 Ensemble learning

Ensemble learning refers to the process of combining several classifiers into one model. The advantage of combining multiple classifiers is that it can increase the performance of a model and reduce the likelihood of unfortunate predictions [49]. Some problems can be too complex for a single classifier to solve on its own. As mentioned in Sec. 2.4, a classifier can have errors that come from variance and bias, and that a good model should have a balance between these two errors. Ensemble learning is a way of finding a good balance. By combining different classifiers, the weakness of an individual classifier is cancelled out, creating a more robust model [14]. A strategic combination of classifiers can achieve diversity in the model to reduce the total error, assuming that the individual classifiers make different errors [49].

Methods

There exist different methods for creating ensembles, and some will be discussed further. There are also other methods for combining the predictions of an ensemble. A widely used technique is **majority voting** [14]. For classification, each classifier will predict a class label, and the final prediction is the class that the majority of the classifiers predict.

For individual classifiers, C_j , and m classifiers, the predicted class label can be found with the equation:

$$\hat{y} = \text{mode}\{C_1(x), C_2(x), \dots, C_m(x)\} \quad (2.42)$$

Mode is defined as the most frequent result in a set.

Bagging

Bootstrap aggregating, also known as bagging, is a technique that applies an ensemble built of similar classifiers [14]. The idea behind bagging is to train the classifiers on different samples of the training data, creating a diverse group of members in the ensemble [49]. The same training data is used but split into smaller, random subsets with replacement. Drawing random combinations of the training data with repetition can help reduce the variance error, reducing overfitting. The final prediction of the ensemble is the class predicted by the majority of the classifiers in the ensemble, as previously described as majority voting. Bagging is the technique used for the Random Forest classifier, mentioned in Sec. 2.3. The classifiers are all Decisions Trees that are trained on unique subsets of the training data. Figure 2.19 shows a bagging ensemble with decision trees.

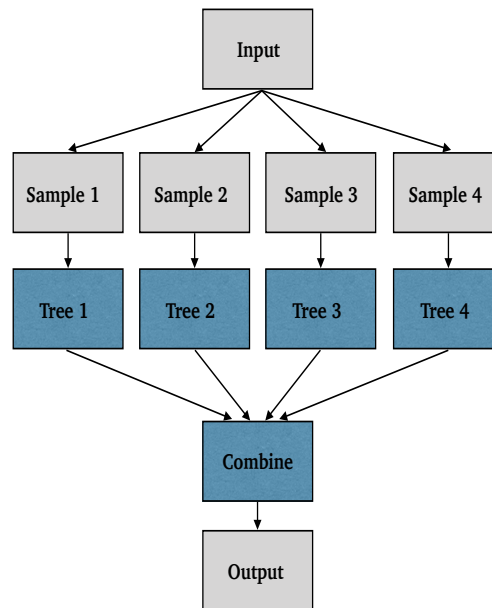


Figure 2.19: A bagging ensemble with decision trees. The input data, X , is split into samples used for training the different trees, and the decisions are combined to produce the output.

Boosting

Boosting is an ensemble of simple classifiers, also called weak learners [14]. The idea behind boosting is training on samples that are hard to classify and letting the weak learners improve the ensemble's performance by learning from the misclassified samples. The first classifier is trained on a random subset of the training data, and the next classifier attempts to correct predictions of the misclassified samples of the first classifier. The process continues through all the classifiers in the ensemble, and weights are adjusted based on the last classification. This process builds a more robust model and decreases the bias error [49]. Finally, the contribution from the classifiers is weighted after their performance, and the predictions are combined using voting or averaging [50]. Figure 2.20 shows a boosting ensemble.

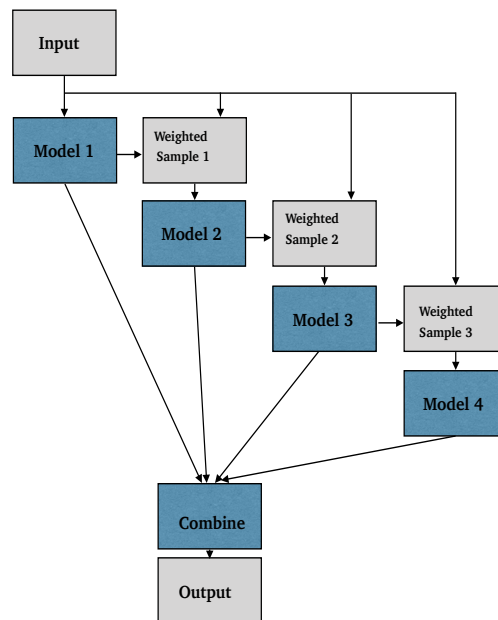


Figure 2.20: Boosting ensemble. The first classifier is trained on a sub-sample of the input, X , and the second classifier with a weighted sample from the first classifier. The results are combined to produce the final prediction.

Stacking

Stacking is an ensemble method that uses different types of classifiers to make predictions on the same training data. The diversity of this model is achieved through the use of different classifiers. Different classifiers will make different assumptions and have fewer correlated errors when predicting [50]. Depending on the classifiers combined, the ensemble can lead to a decrease in bias or variance errors. A stacking ensemble consists of layers. The classifiers in the first layer are referred to as level-0 classifiers and the final classifier as a level-1 classifier. It is common for the ensemble to consist of two levels, but it is possible to construct an ensemble with several levels [50]. The final classifier can also be referred to as a meta-classifier [49]. Cross-validation can be used to train the level-0 classifiers, and their predictions are used to form a new dataset. The meta-classifier is trained on this new dataset, along with the actual labels, and learns how to combine the predictions best. Finally, the meta-classifier makes the final prediction. Figure 2.21 shows an ensemble constructed with the stacking method.

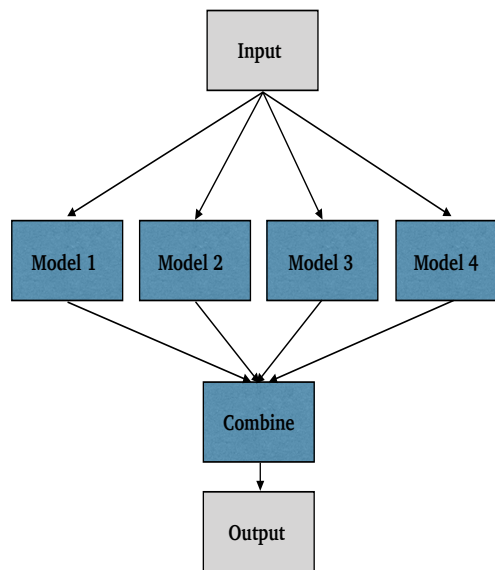


Figure 2.21: An ensemble of classifiers using stacking with two levels. The same data is used to train the four level-0 classifiers and their predictions are used to form a new dataset. The level-1 classifier is trained on this dataset and makes the final predictions.

Chapter 3

Materials

This section describes the material used. In this thesis, patient data were analysed. The data was provided by Computational Radiology and Artificial Intelligence (CRAI), in Oslo University Hospital. The data has been collected from several hospitals in Norway from 2013 to 2022 and consists of 1505 patient assessments measured with 1733 features. Some of the patients have several assessments, so the number of unique patients in the dataset is 789. The target feature used in this thesis is a binary feature where the value 1 indicates biomarker evidence for amyloidosis and the value 0 that no such evidence is present. The presence of amyloid-beta does not necessarily mean that a patient has Alzheimer's disease, but is considered a candidate for developing the disease. Out of the 1505 patient assessments, 1025 have a measurement for the target used.

3.1 Data collection

The data was collected through different hospitals, and collected from different sources. Some features contain information about the patient that was collected through doctor appointments or over the phone. Features containing measurements from clinical data, such as blood tests or measurements of blood pressure, were collected through doctor appointments. The dataset also includes images of the brain as measurements of magnetic resonance imaging (MRI). Features were extracted from the images and volumetric measurements of brain structures are used in this analysis. Different types of scanners were used in the collection of these images.

3.2 Block structure

Because the data comes from different sources, the features were divided into similar groups, called blocks. The variance can be different for each block and it could be beneficial to make local models adapted to the data rather than using one model for all the features. Removing the patients with any missing values lead to a different number of patients with complete data for the different blocks.

3.2.1 Block A: Background information

The first block consists of only one categorical feature, which is presented in Table 3.1. This feature contains background information about the patient before any tests are taken. It provides information about whether a patient belongs to a symptom group or a control group. There are seven different categories: cognitive symptom group, cognitive symptom group - extension from previous study, parkinsonism symptom group, spouse control group, ordinary control group, orthopedic control group and family history control group. In summary, there are three categories related to symptom groups and four categories related to control groups.

The number of patient assessments with complete data in Block A is 1023. One-hot-encoding this categorical feature transforms it into a matrix with dimensions (1023, 7). In Block A there are 777 unique patients.

Table 3.1: Feature in Block A. Containing background information about the patient.

Feature name	Description	Data type
subj_group	Which group the patient belongs to	Categorical

3.2.2 Block B: Environment and heritage

The second block contains personal information about the patient, and the features are listed in Table 3.2.

The youngest patient in this data has an age of 40, while the oldest has an age of 87. The feature edu_years ranges from 4 to 25 and edu_level from 0 to 5. For the feature smok a patient has a value of 0 if they are not smoking and a value of 1 if they either smoke or previously have smoked. The feature bp_recum_sys, measuring blood pressure, has the lowest value of 90 and the highest value of 225.

The feature cardiovascular_disease consist of binary values. If the value for this feature is 1, the patient has one or more out of ten conditions related to cardiovascular disease. If a patient has a mother or father with dementia, they will have a value of 1 for the feature degree1_dem, while 0 if they don't. As mentioned in Sec. 1 everyone has a pair of the three existing forms of the gene apolipoprotein. The categories in the feature bl_apoe are the six different combinations of the APOE pairs: e2/e2, e2/e3, e2/e4, e3/e3, e3/e4, e4/e4.

The number of patient assessments with complete data in Block B is 443, where 389 are unique patients. One-hot-encoding the categorical features transforms Block B into a matrix with dimensions (443, 15).

Table 3.2: Features in Block B. Containing personal information about the patient related to environmental factors and factors connected to heritage.

Feature name	Description	Data type
age	Age of patient	Continuous
edu_years	Years of education	Continuous
edu_level	Level of education	Ordinal
smok	Smoking or not	Binary
bp_recum_sys	Systolic blood pressure when patient is lying down	Continuous
cardiovascular_disease	Previous cardiovascular disease or not	Binary
gender	Male or female	Categorical
degree1_dem	Mother or father with dementia	Binary
bl_apoe	Apolipoprotein E alleles	Categorical

3.2.3 Block C: Cognitive tests

The third block contains seven features that are related to cognitive tests. The features are listed in Table 3.3 and include standard tests used for determining if a patient is developing cognitive impairment or dementia. All the features are continuous and represent the score a patient received on the individual tests.

The first feature, `mmse_total`, is used for determining cognitive function and included are tests of orientation, memory, language and visual-spatial skills. In the data used the values for the score range from 11 to 30. `clock_score` is a test used to look for signs of dementia and the features have a minimum score of 1 and the maximum score is 5. `tmtb_sec` and `tmta_sec` are tests used to measure the ability to focus on one or several things. `tmta_sec` has the lowest value of 4 and the highest value of 307. A maximum score for this feature was set to 71 and all the values over this were set to equal to the maximum score. For `tmtb_sec` the values range from 22 to 694, and a maximum score was set to 166. The `vosp_tscore` feature contains values ranging from 0 to 69.5 and is a test that focuses on object- and space perception. `cowat_tscore` measures verbal fluency and the lowest score is 18, while the highest score is 83. The last feature, `cerad_recall`, has a score ranging from 0 to 10.

There are 917 patient assessments with complete data in Block C, where 703 are unique patients. The dimensions of the data in Block C is (917, 7)

Table 3.3: Features in Block C. This block contains information about cognitive test results.

Feature name	Description	Data type
<code>mmse_total</code>	Mini Mental Status Evaluation total score	Continuous
<code>clock_score</code>	Clock test used in the assessment of dementia	Continuous
<code>tmtb_sec</code>	Trail making test A (TMT-A) measured in seconds	Continuous
<code>tmta_sec</code>	Trail making test B (TMT-B) measured in seconds	Continuous
<code>vosp_tscore</code>	Visual Object and Space Perception	Continuous
<code>cowat_tscore</code>	Controlled Oral Word Association Test	Continuous
<code>cerad_recall</code>	Consortium to Establish a Registry for Alzheimer's Disease (CERAD) word list recall	Continuous

3.2.4 Block D: White matter hyperintensity load by region

The features in the fourth block are all continuous and the original features are listed in Table 3.4. The features consist of lesion and white matter hyperintensity measurements, which is a proxy for small vessel diseases in the brain. The lesion

features are divided into four regions and layers, where the first letter gives information about which region, lobe. F is for frontal, P is for parietal, O is for occipital and T is for temporal. Each region is then divided into four layers. There are 16 features related to lesions. The lesions features are divided on a feature called TIV, which is a measure of the volume of the cranium. The feature for white matter hyperintensity, WMHo_rV, contains combined measurements for all layers. The feature PSMD is a measure of vascular damage. In Block D, there are 284 patient assessments that have complete data. The number of unique patients is 277.

Table 3.4: Original features in Block D.

Feature name	Feature name
WMHo_rV	PSMD
LesF1	LesF2
LesF3	LesF4
LesP1	LesP2
LesP3	LesP4
LesO1	LesO2
LesO3	LesO4
LesT1	LesT2
LesT3	LesT4

Later in the analysis the lesion features were combined, meaning that the four layers were combined for each region. Table 3.5 shows the combined features along with the minimum- and maximum value for each feature. The dimensions of the data in Block D with the combined features is (284, 6).

Table 3.5: Combined features in Block D. The table includes the minimum- and maximum value for each feature.

Feature name	Minimum value	Maximum Value
WMHo_rV	$3.33 * 10^{-4}$	$2.64 * 10^{-2}$
PSMD	$1.67 * 10^{-4}$	$7.52 * 10^{-4}$
LesF	$2.32 * 10^{-5}$	$3.51 * 10^{-2}$
LesP	$1.62 * 10^{-5}$	$1.37 * 10^{-2}$
LesO	$1.34 * 10^{-5}$	$5.41 * 10^{-3}$
LesT	$4.65 * 10^{-6}$	$5.30 * 10^{-3}$

3.2.5 Block E: MR images of subcortical brain structures

The fifth block contains ten continuous features that are listed in Table 3.6. The features are volumetric measurements of subcortical brain structures. The features are corrected for intracranial volume and mean thickness of cortex areas from MR images. There is also a correction related to the fact that the measurements come from different sites and that different scanners have been used for imaging. The features are combined measurements for the left- and the right side of the brain. There are 603 patient assessments, and 393 unique patients, with complete data in Block E. The dimensions of the data in Block E is (603, 10)

Table 3.6: Features in Block E. The features contain thickness measurements for different areas. The table includes the minimum- and maximum value for each feature.

Feature name	Minimum value	Maximum Value
Anterior_hippocampus	692	2939
Posterior_hippocampus	742	2302
MISC	-22.8	656
Meninges.PHC	4.8	355
ERC	211	885
Br35	227	1042
Br36	1007	3100
PHC	576	1440
ColSul	213	920
Meninges	313	835

Chapter 4

Methods

This chapter covers the methodology used in this thesis. Sec. 4.2 presents the workflow used in this study. Sec. 4.3 describes the preprocessing of the data before model training. Sec. 4.4 covers the methods used for explorative analysis of the data with PCA and PLSR. The different models used for classification are covered in Sec. 4.6, including the methods for evaluating and tuning the models. Sec. 4.7 describes the method of feature selection with RENT and Sec. 4.8 describes the methods used for ensemble learning.

The code used for this thesis is available at GitLab: https://gitlab.com/charlie_91/alzheimer.

4.1 Software

The software used in this study was implemented as a part of the Anaconda open source distribution [51]. Python version 3.9.5 was used. For reading the data, pyreadstat [52] version 1.1.0 was used. Pandas [53] version 1.2.4 and NumPy [54] version 1.20.2 were used for handling the data. The package Phi_k Correlation Analyzer [55] version 0.12.0 was used for analysing the correlation between the features. The packages Hoggorm [56] version 0.13.3 and Hoggormplot [57] version 0.13.2 were used for exploratory analysis with PCA and PLSR and visualization of the results. Other packages used for visualization include Matplotlib [58] version 3.4.2, Missingno [59] version 0.4.2, and Seaborn [60] version 0.11.1. Scikit-lego [61] version 0.6.8 was used for outlier detection. The package Rent [62] version 0.0.1 was used for feature selection, and Scikit-learn [63] version 0.24.2 was used for data handling and machine learning tasks.

4.2 Workflow

Fig. 4.1 shows an illustration of the workflow used in this study. This gives an overview of the steps involved, which are numbered in the figure. Step 1 was preprocessing the data, which include dividing the data into blocks. Because of the different number of patients assessments in the blocks, explorative analysis was performed block-wise (step 2). Step 3 was splitting the data into training- and test data. Exploratory analysis was again performed on the test data (step 4) and the training data (step 6). Step 5 covers the baseline analysis using the test data with all the features combined into one block. Step 7 includes the block-wise analysis with RENT for feature selection. Using the selected features from RENT, block-wise modelling was done on the training data (step 8). Using the classifiers that perform best for the individual blocks, the class labels for the test set were predicted (step 9). Step 10 was the final step and includes creating an ensemble using weighted majority voting. The individual steps will be described in more detail in the following sections.

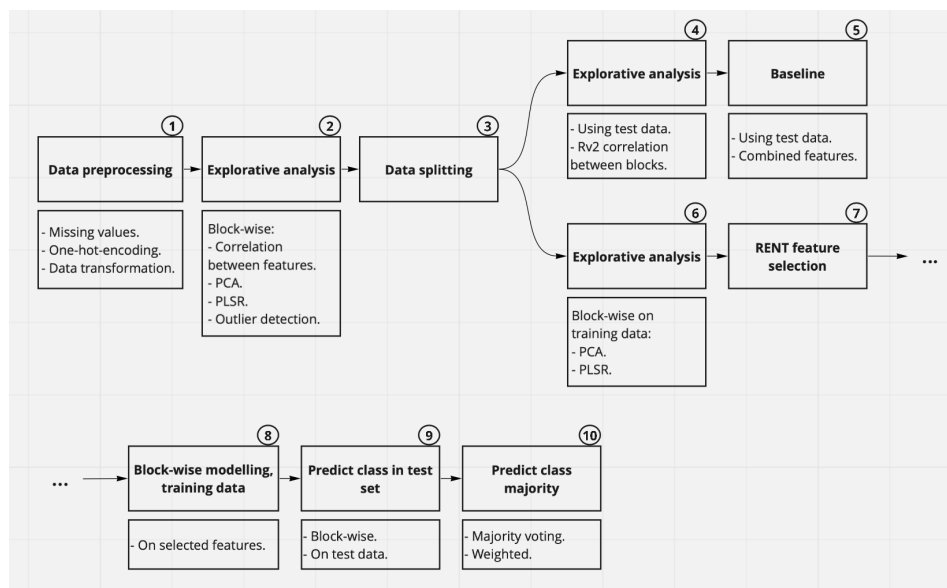


Figure 4.1: Illustration of the workflow used in this study.

4.3 Data preprocessing

Step 1 in the workflow was preprocessing the data, and the first step in this process was to identify patient assessments where a value for the target feature, a , was missing. After removing the assessments with a missing value for the target feature, the number of assessments went from 1505 to 1025.

The dataset was then divided into the blocks described in Sec. 3.2. The package Missingno was used to visualize the number of missing values for the different features. The patients with any missing values were removed, so only patients with complete data remained. Most of the missing values appeared as NaN (not a number). Some of the patients had empty strings in the categorical features. These were replaced with NaN so that all the NaN entries could be removed using Pandas *dropna* function. The number of patient assessments with complete data differs for the blocks. Table 4.1 gives an overview of the number of patient assessments with complete data in the different blocks.

Table 4.1: Number of patient assessments with complete data in the different blocks.

Block	Patient assessments
A	1023
B	443
C	917
D	284
E	603

4.3.1 Target

The target used in this study indicates if there is evidence for biomarker evidence for amyloidosis. The feature did not need any preprocessing as the values were binary and ready for use in classification. The target feature was separated as its own dataframe before preprocessing the rest of the features.

4.3.2 One Hot Encoding

The categorical features in Block A and Block B were converted to numerical values using the *get_dummies* function in Pandas. The function turns categorical labels into numerical values using One-hot-encoding, described in Sec. 2.1.

4.3.3 Data transformation

Different transformations of the data were tested. The scikit-learn package comes with modules for data transformation. All the features were scaled, and the distribution of the features was visualized. Due to skewness in the distribution, the features in Block D were power transformed with two different transformations. Models were trained separately with the different strategies in order to compare the results. The main strategies used for transformation are listed below.

1. Standardising all the features with Eq. 2.2.
2. Power transformation of the features in Block D using Yeo Johnson method, described mathematically in Eq. 2.4. The power transformation was applied to standardised data. The rest of the features were standardised using Eq. 2.2.
3. Power transformation of the features in Block D using the BoxCox method, described mathematically in Eq. 2.3. The power transformation was applied to normalised data using Eq. 2.1 and setting the range between 1 and 2. The rest of the features were standardised using Eq. 2.2.

4.4 Explorative analysis

This section describes the methods used for exploratory analysis of the data. After dividing the data into blocks, the correlation between the features in each block was analysed. PCA, PLSR and outlier detection with scikit-lego was performed on each block to get an overview over systematic variance in the data and potential outliers. This is step 2 in Fig. 4.1. After this the data was split into training- and test data as shown in Fig. 4.2. Using the test data, the RV_2 -coefficient was calculated to check for overlapping information between the blocks, which is step 4 in the workflow. The explorative analysis with PCA and PLSR was performed on the training data to verify that the variance did not change much before and after splitting. This is step 6 in the workflow and was performed to confirm no major changes compared to step 2.

4.4.1 Correlation between features and blocks

When analysing the correlation between the features within the blocks, the coefficient was calculated using the function *corr* in Pandas. The function calculated the correlation with the Eq. 2.6. ϕ_k is a correlation coefficient that is based on refinements to the mentioned correlation coefficient. It is possible to calculate the ϕ_k correlation coefficient between categorical and numerical features and is useful when analysing features of different types. The ϕ_k correlation coefficient between the features was calculated using the function *phik_matrix* from the Phi_K Correlation Analyzer package.

As described in Sec. 2.7 the correlation between the blocks can be calculated using the RV-coefficient, in order to look for common information. The function *RV2coeff* from the Hoggorm package was used to calculate the RV_2 -coefficient between the blocks. To calculate the RV_2 -coefficient between all the blocks, the number of measurements had to be equal and in the same order. The test set was used, where the patient assessments have complete measurements across the blocks.

4.4.2 PCA

The Hoggorm package was used to perform PCA on the blocks, with the function *nipalsPCA*. The exploratory analysis was done to examine the data for patterns or systematic variation that could be of interest. Both functions take the values of the dataframes as input, and it is possible to specify the number of components to be computed. The explained variance in each block was plotted with the package Hoggormplot to indicate how many components were necessary for explaining the variance in the different blocks. As mentioned in Sec. 2.2.4 PLSR is a supervised technique, and it is also possible to analyse the explained variance for the target, Y. Visualization of the explained variance was done for analysis, but the dimensions of the dataset were not reduced using these techniques.

In addition to dimensionality reduction, PCA can also be used for explorative analysis of the data. Using the Hoggormplot package, the scores, loadings, and correlation loadings were visualized for the different blocks. This was done to look for outliers and for clustering of features.

4.4.3 PLSR

The Hoggorm package was also used to perform PLSR on the blocks, with the function *nipalsPLS1*. Plots were created of the scores, loadings, and correlation loadings obtained from the PLSR analysis. With PLSR, it was also possible to visualize the explained variance for the target variable. The plot of the correlation loadings also includes the target variable.

The procedure used for explorative analysis with PCA and PLSR block-wise was also performed after splitting the data into training- and test-set, where the splitting is shown in Fig. 4.2. The analysis was done on the training data to verify that the systematic variation and patterns before and after splitting were similar. A table containing the explained variance in X and Y was created.

4.5 Data splitting

As described the features were divided into blocks of similar measurements. The data were analysed both separately for each block, but also for all blocks concate-

nated into one large single block to investigate whether block-wise modelling can provide better performance. The dataset with all the features combined was used for establishing a baseline. When combined, the total number of features is 33. After One-hot-encoding the categorical features, the number of features increased to 45. None of the patients belonged to the category parkinsonism symptom group from Block A, reducing the number of features to 44. The number of patient assessments with complete data across all the blocks is 172. When looking at unique patient ID's there are 154 with complete data across all the blocks. The 172 patient assessments were used to form the complete dataset, and also the test set for the different blocks, meaning that the number of patient assessments in the test set was the same for every block, but the number of features differ. Figure 4.2 gives a visual illustration of how the patients in the different blocks were split into a training- and a test set and is step 3 in the workflow presented in Fig. 4.1.

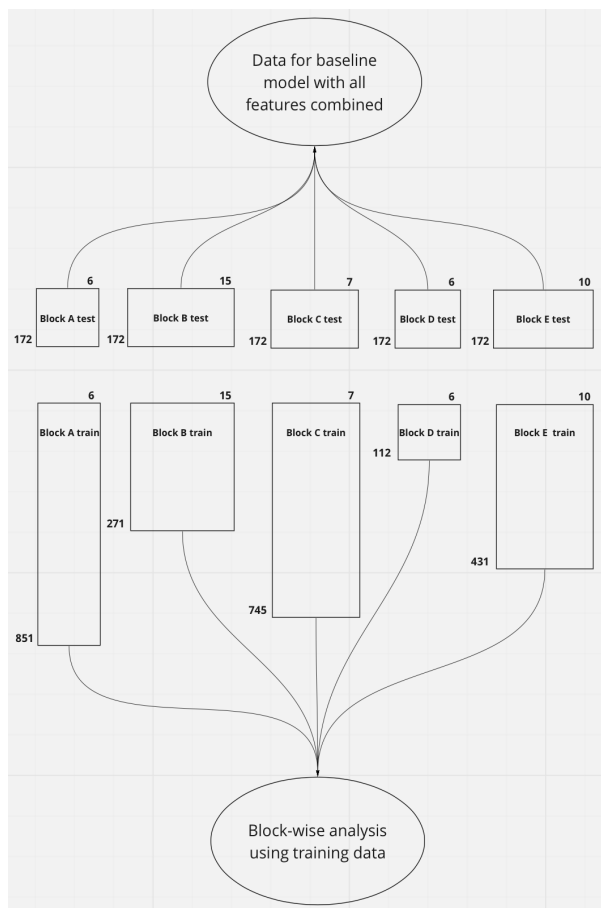


Figure 4.2: A visual representation of how the data was split into training- and test set for the different blocks. A baseline was established combining all the features with the patient assessments in the test set.

The number of patient assessments in the training data is different for the individual blocks. This is because some of the patient assessments have missing values for features in some of the blocks and therefore are not present in all blocks. The dataset used to establish a baseline contained 44 features after combining the features from each block.

4.6 Classification

4.6.1 Baseline model

Step 5 in the workflow is the baseline analysis. The dataset for establishing the baseline is based on the test data and consist of 172 patient assessments and 42 features, after one-hot-encoding the categorical features. The baseline was used to create a measure for improvement when predictions are made based on models trained on each block separately. The total sum of features was 44 after one-hot-encoding, but none of the 172 patients belonged to the category orthopedic control group from Block A or bl_apoe_E2/E2 from Block B, reducing the number of features by two. The dimensions for the test data was (172, 42). All the samples were used in repeated stratified k-fold cross-validation with $k = 5$.

4.6.2 Block-wise models

As mentioned, and illustrated in Fig. 4.2, the data was split into a training- and a test-set block-wise. Table 4.2 gives an overview of the data dimensions for the different blocks. For each block models were trained using the classifiers described in Sec. 2.3, and will also be described in the following section. These models were trained only on the training data, and for all the models k-fold cross-validation was used to identify the classifier with the highest MCC performance for the individual blocks. The block-wise modelling comes after the RENT analysis and is presented as step 7 in the workflow.

Table 4.2: Dimensions of training and test data for the different blocks

Block	Training	Test
A	(851, 6)	(172, 6)
B	(271, 15)	(172, 15)
C	(745, 7)	(172, 7)
D	(112, 6)	(172, 6)
E	(431, 10)	(172, 10)

4.6.3 Tuning

The algorithms used for classification are described in Sec. 2.3. These are the models that were tested and compared, except for the Decision Tree classifier; the theory behind this model was described to better understand the Random Forest model. The models are available through the Scikit-Learn package.

As mentioned in Sec. 2.4, the models have hyperparameters that can be adjusted to increase their performance. Parameter grids for the hyperparameters were defined, and grid search was performed on each model to find the best hyperparameters. For this purpose, the scikit-learn function *GridSearchCV* was applied that carried out a 5 fold cross-validated grid search over a defined grid of parameters to find the combination of hyperparameters that provided the best predictive performance on the training data. The estimator, parameter grid, scoring and number of fold in the cross-validation was used as input for *GridSearchCV*. Grid search was performed on both the baseline model and the block-wise models to get the best parameters for each individual case. Table 4.3 gives an overview of the different models and the hyperparameters that were tested.

Table 4.3: List of algorithms, their hyperparameters and a brief description of those.

Model	Hyperparameter	Description
Logistic Regression	C solver	Inverse of regularisation strength Algorithm for optimization problem
Passive Aggressive	C loss	Maximum step size for regularization Loss function to be used
Random Forest	n_estimators criterion max_depth max_features	Trees in the forest Function to measure the quality of a split Maximum depth of the tree Maximum number of features to consider
SVM	C kernel gamma	Regularisation parameter Kernel type to be used Kernel coefficient
KNN	n_neighbors weights algorithm	Number of nearest neighbors Weight function for prediction To compute the nearest neighbors

4.6.4 Performance metrics

The distribution of the classes was checked for the baseline data, the data with all the features combined. Out of 172 targets, 49 were class label 1 and 123 were class label 0. Given that the distribution of the classes was not even, the metric used for scoring was Matthews correlation coefficient (MCC), described in Sec. 2.4. The distribution was also checked for the training data for the different blocks, with specific numbers presented in Sec. 6.1. Uneven distribution of the classes was found in all the blocks. MCC was used as the main scoring metric for this thesis, but in some cases, the accuracy score and the f1-score were also calculated for comparison. The theory behind these scores is also described in Sec. 2.4.

4.6.5 Evaluation

The different classifiers were initialized with the parameters that gave the best results from the grid search. For evaluation, the function *RepeatedStratifiedKFold* from Scikit-Learn was used. The function was used with 10 repeats and 5 splits. The mean of the resulting scores was used for evaluating the performance of each of the classifiers.

4.7 Feature selection

Feature selection was done with Repeated Elastic Net Technique (RENT), presented in Sec. 2.5. Referring to the workflow in Fig. 4.1 this is step 7. In addition to the described parameters τ and K , RENT can take input values for parameters C and $l1_ratios$. C is the regularisation parameter for the models in the ensemble. A lower value for this parameter will lead to stronger regularisation. $l1$ -ratio is the ratio between $l1$ and $l2$ penalty. A value of 0 gives only $l2$ penalty, and a value of 1 gives only $l1$ penalty. The input values can be given as a list of numbers to test. This was done with values for C ranging from 0.01 to 10, and values for $l1_ratios$ from 0 to 1.

500 models were used in the ensemble, and the parameter *autoEnetParSel* was set to True, which uses 5-fold cross-validation, prior to RENT, to identify the best performing combination of C and $l1$ -ratio for prediction. RENT comes with a function *get_enetParam_matrices()* that returns three dataframes holding results for all the pair-wise combinations of C and $l1$ -ratio. The results include the average scores for performance, average percentage of feature weights that were set to zero, and a scaled harmonic mean between the two previous dataframes. RENT uses the harmonic mean results to select the best combination of the two parameters, but they can also be set manually, which is what is done in this study. Heatmaps of the matrices containing the scores and fraction of feature weights set to zero were plotted to choose the best value for the parameters.

After selecting the parameters, RENT was performed with different values for the parameters τ_1 and τ_2 . This was done for values ranging from 0.2 to 1 in intervals of 0.05 and $\tau_1 = \tau_2$. For each value of τ , a new dataset was created with the features selected by RENT. Repeated stratified k-fold with 5 splits and 10 repeats was used to evaluate the different classifiers on the created dataset. This was visualised with the score as a function of τ_1 for the different classifiers in order to make a suitable selection for τ_1 and τ_2 .

This procedure was done for all the features combined to establish a baseline and also separately for each block. For the baseline data with combined features, k-fold cross-validation was performed on all the test data. For the block-wise evaluation of classifiers using k-fold cross-validation, the training data was used.

For the baseline, a final score for the classifiers was found through repeated stratified k-fold with the selected value for the parameters. In the block-wise analysis, a repeated RENT procedure was followed to investigate the features selected by RENT. A repeated method ensures a more robust analysis. The selected values for the parameters were used for each individual block and a repeated stratified k-fold procedure with 4 folds and 10 repeats was used, resulting in 40 RENT feature selections. RENT is performed on 3 out of the 4 folds and, using the selected features, performance was calculated on the last fold. Using 10 repeats gives 40 feature selections and it is possible to aggregate the results over these 40 sets of selected features in order to count how often a feature is selected.

4.8 Ensemble learning

Weighted average

A new dataset was created using the features selected through the RENT analysis. Grid search and repeated stratified k-fold were performed block-wise to find the classifier with the highest performance in each block, which was explained in more detail in Sec. 4.6. The classifier with the highest MCC performance in each block was used for predicting the class labels for the test set when using only the selected features through RENT. In the workflow, this is presented as step 9. Step 10 in the workflow is the final step and involves creating an ensemble using a weighted majority vote. The predicted class labels from each block were used to calculate a weighted average from the best classifier in each block, where the weights correspond to the MCC-performance for each classifier found through repeated stratified k-fold. By using the MCC-performance as weights a poor performing classifier will have less influence on the final prediction compared to a well-performing classifier. The weighted average was created using Numpy's *average* function. The weighted average was compared against the true label for a final prediction.

Chapter 5

Results

This chapter will present the results of this analysis. A short explanation will be given in this chapter, while a more detailed discussion of the results will be covered in the next chapter, Ch. 6.

Sec. 5.1 will present the results from preprocessing and contains visualisation from the power transformation of the data and missing values. Results from the explorative analysis will be covered in Sec. 5.2. This includes heatmaps of the correlation coefficients, results from PCA and PLSR analysis.

Sec. 5.3 will present the results from classification for the baseline model and the block-wise models. Sec. 5.4 will present the results from RENT for feature selection, first for the baseline analysis and then for the block-wise analysis. Results from the ensemble of models, the final model, will be covered in Sec. 5.5.

5.1 Data preprocessing

5.1.1 Power transformation

Different types of transformations of the data were tested. Skewed distributions were mainly an issue for the features in Block D. Fig. 5.1 shows the distribution of the four different LesP features in Block D. The figure on the left is the distribution before transformation, and the figure on the right shows the distribution after Yeo-Johnson transformation was applied to the data. Later in this study, the Les-features were combined. Fig. 5.2 shows the distribution before and after Yeo-Johnson transformation on the combined LesP features. Similar results could be observed in the transformation of the other Les features.

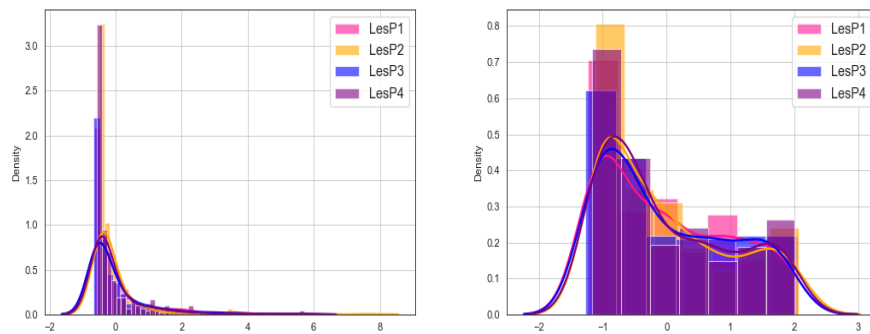


Figure 5.1: Distribution for four of the features in Block D. The figure on the left is the original distribution, and the figure on the right shows the distribution after Yeo-Johnson transformation was applied.

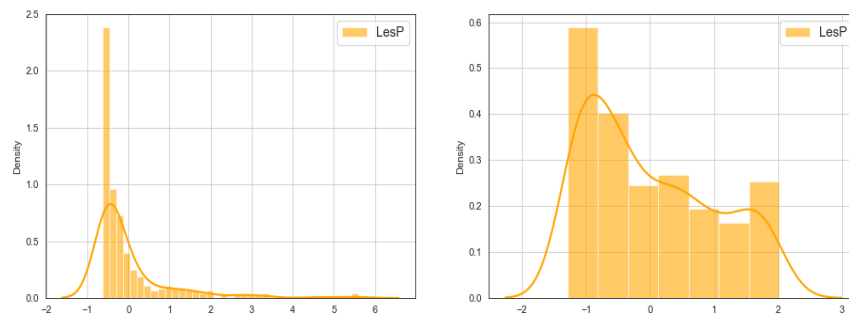


Figure 5.2: Distribution for the combined LesP features in Block D, before and after power transformation.

5.1.2 Missing values

Different plots were created for visualisation of the missing values in the features using the missingno package. The matrix plot, also called the nullity matrix, visually displays the missing values as white lines and makes it simple to identify missing values in the different features. The nullity matrix is shown for Block A in Fig. 5.3. Block A consists of only one feature, labelled at the top of the figure. Almost all the patient assessments have a category for the feature in Block A, and it has only 2 missing values. Fig. 5.4 shows the nullity matrix for Block B, and the features are labelled on the top. As seen by all the white spacing, the feature `degree1_dem` is the feature in this block with the highest amount of missing values. In Block B, 580 assessments were removed due to missing values.

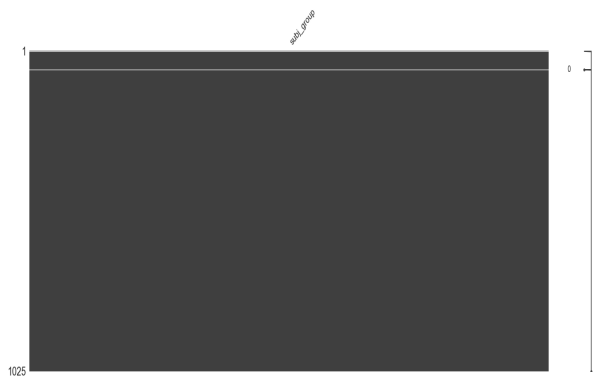


Figure 5.3: Nullity matrix for the feature in Block A.

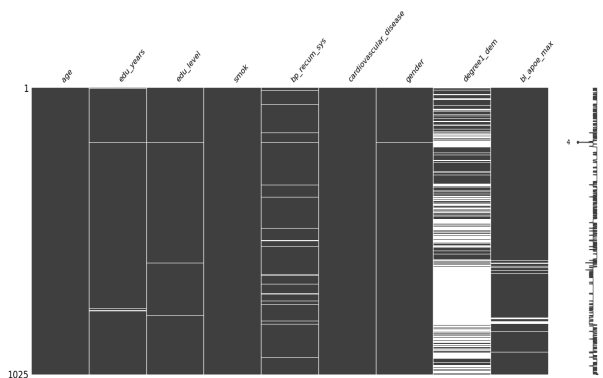


Figure 5.4: Nullity matrix for the features in Block B. The columns are labelled with the name of the feature at the top.

Fig. 5.5 shows the nullity matrix of the features in Block C. The individual features don't contain many missing values, but patients are removed if they have missing values in any of the features. This reduces the number of patients by 108 for this block. Fig. 5.6 and 5.7 shows the nullity matrix for Block D and E, respectively. Block D is the block missing most values for the measurements and resulted in the removal of 740 patient assessments. For Block E the figure illustrates that some patients have no measurement for any of the features. 422 patient assessments were removed due to missing values in Block E.

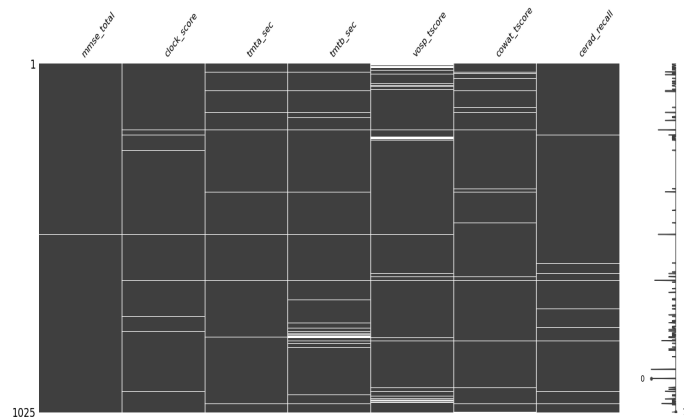


Figure 5.5: Nullity matrix for the features in Block C.

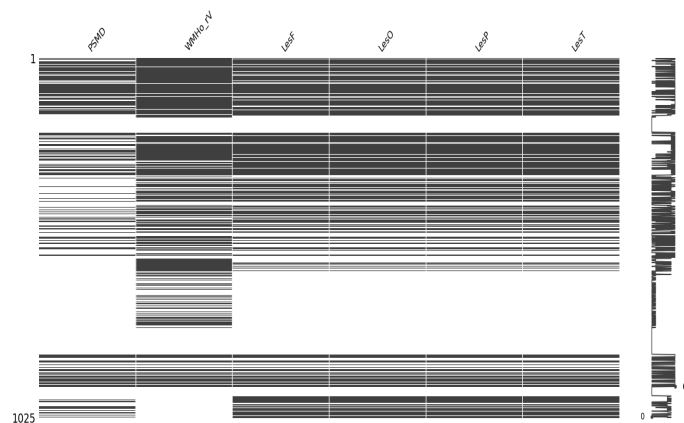


Figure 5.6: Nullity matrix for the features in Block D.

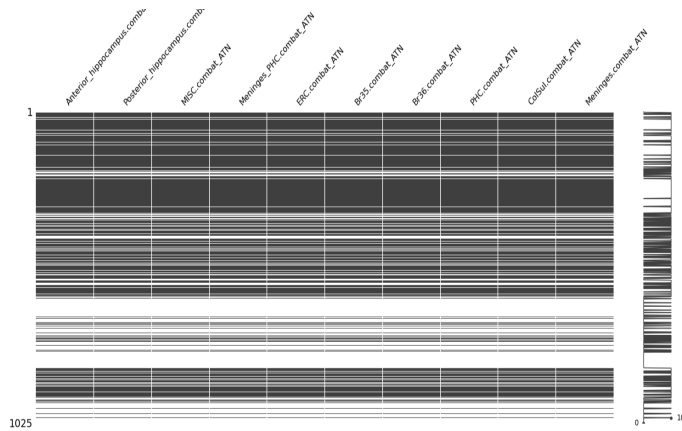


Figure 5.7: Nullity matrix for the features in Block E.

5.2 Explorative analysis

5.2.1 Correlation

Correlation between the blocks

The RV_2 -coefficient was calculated between the blocks using the test data, where the patient assessments have complete data for all the blocks. This was done in the explorative analysis in step 4 from the workflow presented in Fig. 4.1. Values for the RV_2 -coefficient is shown as a heatmap in Fig. 5.8. The block names are listed on both the horizontal axis and the vertical axis. Along the diagonal is the RV_2 -coefficient between the individual blocks themselves, giving the value 1. The low values for the coefficient between the different blocks indicate that the blocks contain little common or overlapping information.



Figure 5.8: Heatmap of the R_{v2} correlation coefficient between the blocks.

Feature correlation within blocks

The correlation coefficient between the features in each block is part of the explorative analysis presented as step 2 in the workflow, and are also presented graphically as heatmaps. For some of the blocks, the PhiK coefficient was used. All the heatmaps contain a colour bar ranging from -1 to 1 . The darker the colour the more strongly the features are correlated, and the lighter the colour the higher the negative correlation is between the features.

Block A contains only one categorical feature, so no correlation could be calculated initially. After one-hot-encoding the categorical feature, the PhiK correlation coefficient was calculated between the categories in the feature. The heatmap of the correlation coefficients between the features in Block A is presented in Fig. 5.9. There is a low correlation between most of the features in this block. The category cognitive symptom group is the category that shows the most correlation with the other categories, especially the categories family control group, orthopedic control group and the ordinary control group.

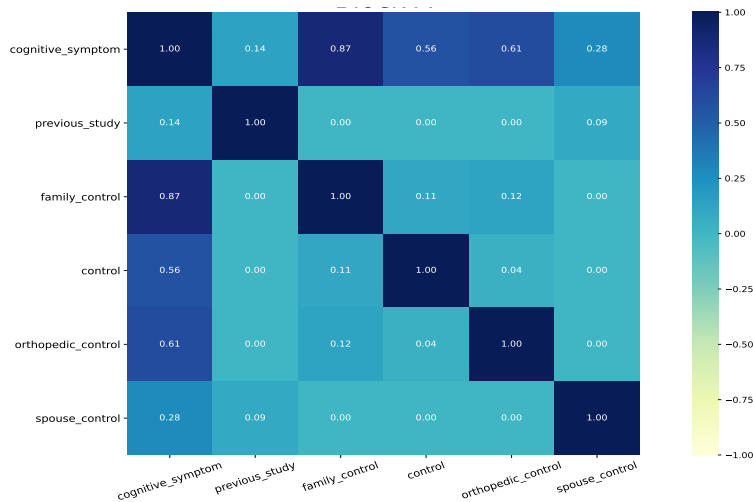


Figure 5.9: Heatmap of the PhiK correlation coefficient between the categories in Block A.

Block B consists of both categorical and continuous features and the PhiK correlation coefficient was calculated and shown in Fig. 5.10. The heatmap shows a high correlation between the feature edu_years and edu_level. The feature age shows a value for correlation coefficient of 0.35 and 0.37 between the features edu_years and edu_levels respectively. The feature age also shows values for the PhiK correlation coefficient that indicates correlation with the features bp_recum_sys, cardiovascular_disease and degree1_dem.

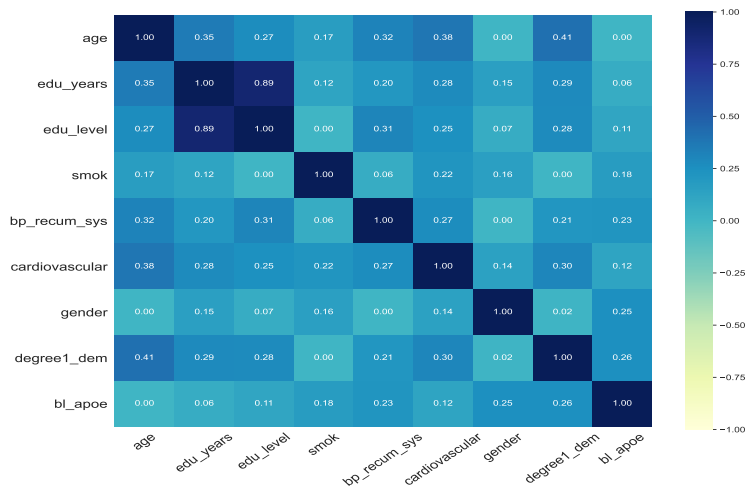


Figure 5.10: Heatmap of the PhiK correlation coefficient between the features in Block B.

For Block C the heatmap of the PhiK correlation coefficients is shown in Fig. 5.11. The PhiK correlation coefficient was used in this block as the continuous values represent scores, so they could be considered as an ordinal data type. The highest correlated features in this block are the feature tmta_sec and tmtb_sec, with a value for the coefficient of 0.64. The heatmap also indicates a high correlation between the features mmse_total and clock_score, with a value for the coefficient of 0.58.

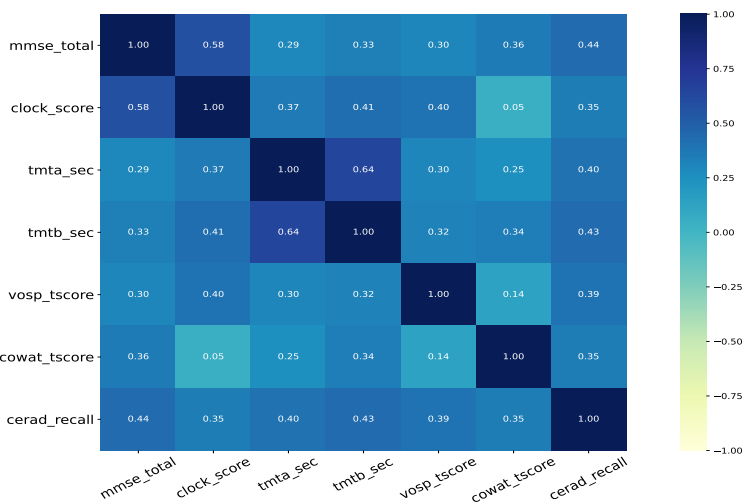


Figure 5.11: Heatmap of the PhiK correlation coefficient between the features in Block C.

Fig. 5.12 presents the results for Block D. The value for the coefficient indicates a strong correlation between the features, with the lowest value being 0.32 between the features PSMD and LesO. The highest value for the correlation coefficient is between the features LesP and LesT, with a value of 0.90.

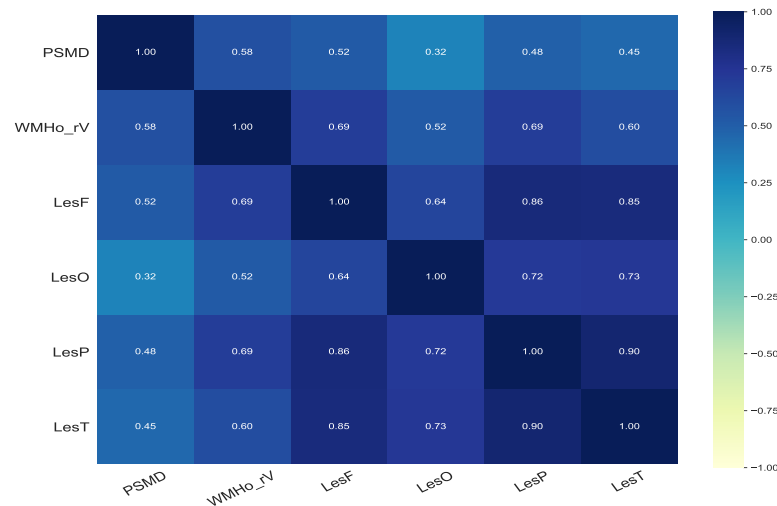


Figure 5.12: Heatmap of the correlation coefficient between the features in Block D.

Block E is the block with the most features before one-hot-encoding, and the heatmap is presented in Fig. 5.13. The values for the correlation coefficient vary, but none of the features indicates strong anticorrelation. Some strong correlations that can be pointed out are between the feature Anterior_hippocampus the features Posterior_hippocampus, ERC and Br35. The features that show a strong correlation with Anterior_hippocampus, also show a strong correlation with the feature Posterior_hippocampus. Br35 indicate a strong correlation between the features PHC, Br36 and ERC, with a value for the coefficient of 0.59, 0.53 and 0.65 respectively.

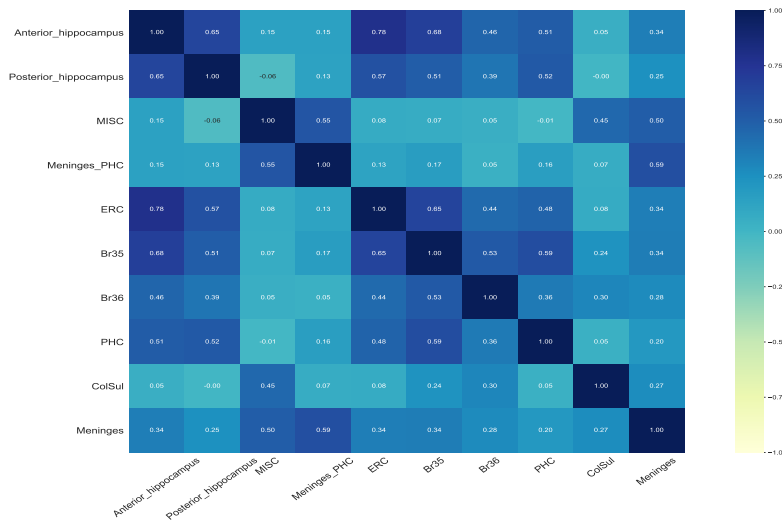


Figure 5.13: Heatmap of the correlation coefficient between the features in Block E.

5.2.2 PCA

Results from the exploratory analysis using PCA include figures of the scores, loadings, correlation loadings and explained variance, and were computed for each block. The following results are based on Block B, while results for the remaining block are included in the Appendix, Sec. 7. For this section and the sections regarding PLSR and outlier detection, all the results are presented from the exploratory analysis performed in step 2 from the workflow. For the exploratory analysis in step 6, tables with results from the PLSR analysis is presented in Sec. 6.3.

The plot of the PCA scores is presented in Fig. 5.14. The first principal component is represented by the horizontal axis and the second component by the vertical axis. The percentage of the explained variance that each of the components contribute to is given in the parenthesis. The plot indicates a clustering of the samples into two groups, but show no extreme outliers.

The plot of the PCA loadings is presented in Fig. 5.15, which have the same axis as the plot of the scores. It shows that the two clusters in the scores plot represent male and female since gender_male and gender_female are on the opposite side along the second component.

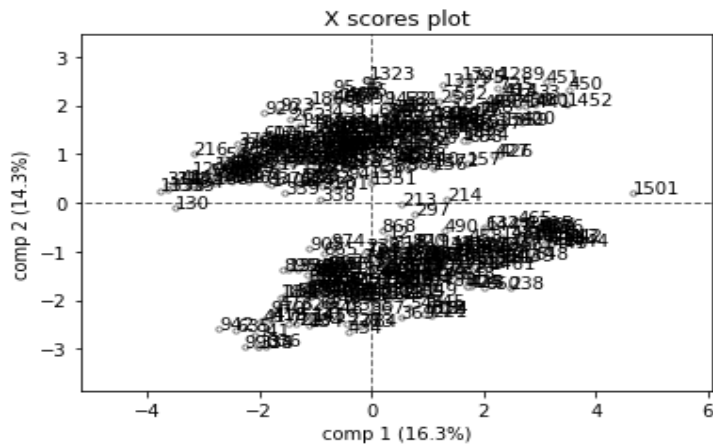


Figure 5.14: PCA scores. The first principal component is along the horizontal axis and the second along the vertical axis. The percentage of the explained variance that each of the components contribute to is given in the parenthesis.

As mentioned in Sec. 2.2.3 a plot of the loadings shows how strongly each feature influences the components. The features `edu_level` and `cardiovascular_disease` lie on the opposite side along the horizontal axis. This can be interpreted that patients with a high level of education contribute to a relatively low proportion of cardiovascular-related disease.

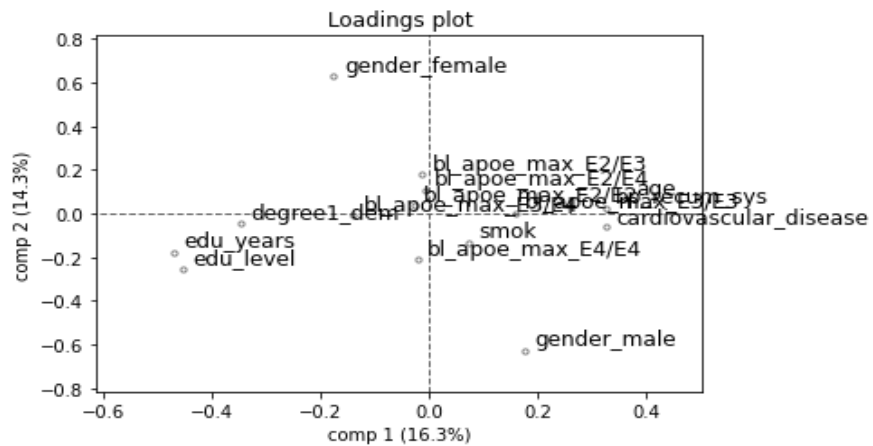


Figure 5.15: Plot of PCA loadings.

The correlation loadings plot is shown in Fig. 5.16. The features `gender_female`, `gender_male`, `edu_levels` and `edu_years` are the only features where the first component and the second component explain more than 50% of their variance. Both gender features are explained almost 100%. Fig. 5.17 shows the explained variance in x as a function of the number of principal components. The variance is accumulated and the vertical axis shows the value for the percentage. The explained variance increases approximately linearly until it reaches a maximum of 70% variance with 6 components.

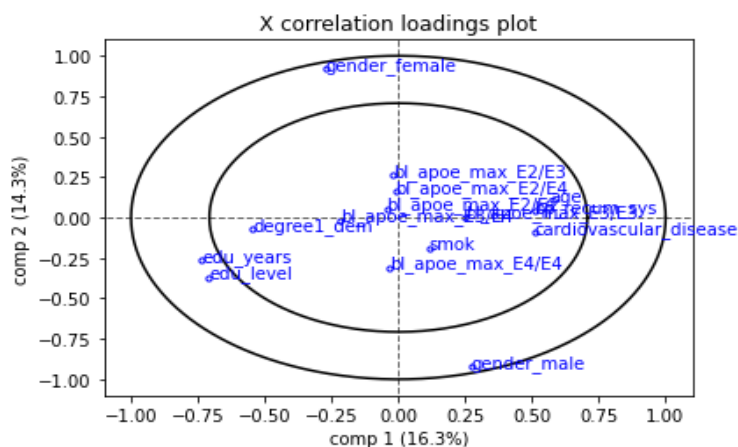


Figure 5.16: Correlation loadings plot from PCA.

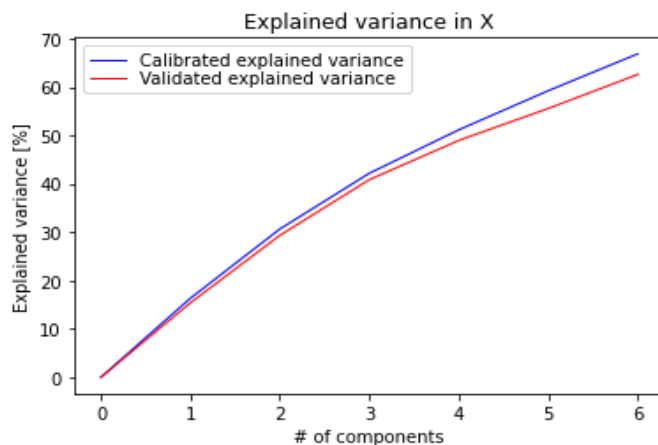


Figure 5.17: Explained variance, given as a function of the number of principal components. The blue line is the calibrated variance, and the red line is the validated variance.

5.2.3 PLSR

The results from the explorative analysis with PLSR include the same type of plots as used with PCA. The results differ because PLSR is a supervised technique, including the target feature in the analysis. In the correlation loadings plot, the target is included, and it is possible to analyse the explained variance in the target. The results from Block B are presented in this section. The results for the other blocks can be found in the Appendix, Sec. 7.

Fig. 5.18 presents the PLSR scores plot, with the first component along the horizontal axis, and the second component along the vertical axis. The percentage of the explained variance that each of the components contribute to is given in the parenthesis. The first number represents the explained variance in x , and the second number the explained variance in y . The scores plot does not show the same grouping as the PCA. The plot of the loadings is shown in Fig. 5.19. It is plotted on the same axis as the scores, but only the explained variance in x is shown for each component.

Compared to PCA the features `gender_male` and `gender_female` are not so dominating along the second component, and other features like `smok`, `cardiovascular_disease`, `edu_level` and `edu_years` contribute in the direction of the second component. Features including `bl_apoe` were insignificant in PCA, having low values for the first component, but have more influence on the first component in PLSR. This is because they are relevant for the modelling of the target.

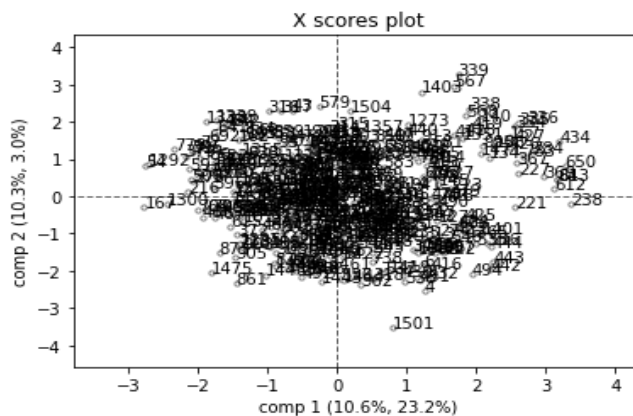


Figure 5.18: Plot of the scores from the PLSR analysis. The first component is along the horizontal axis, including the percentage of explained variance in both x and y . The second component is along the vertical axis.

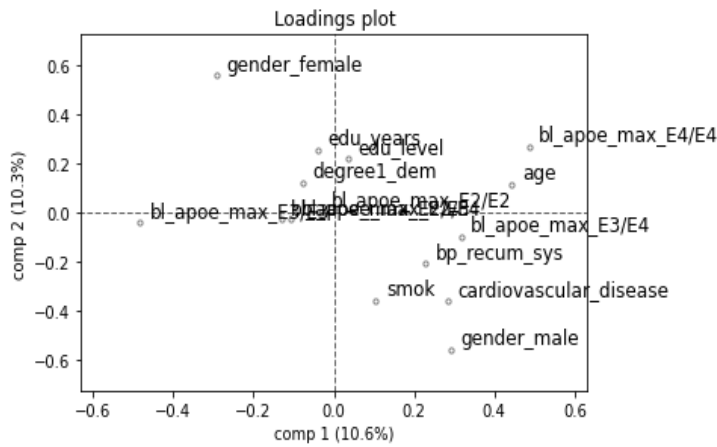


Figure 5.19: Plot of the loadings from the PLSR analysis.

Fig. 5.20 shows the correlation loadings plot. In comparison to the PCA plot, it contains the target feature. The red labels are the features, and the blue label is the target feature. High values for the target feature seem to be correlated with high age and high values for bl_apoe_E4/E4. The patient assessments in the scores plot that lies on the right side are those with high value for the target feature a high age and high bl_apoe_E4/E4. Both the target feature a and bl_apoe_E4/E4 have binary values, where the highest value is 1.

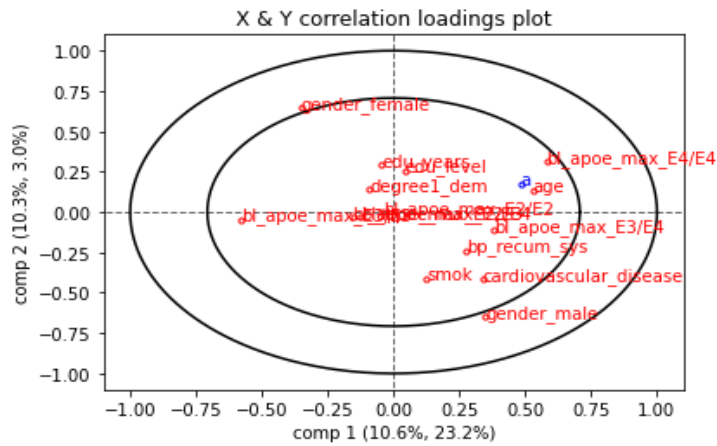


Figure 5.20: Plot of the x and y correlation loadings.

Fig. 5.21 present the explained variance in x. The number of components is given along the horizontal axis, and the percentage of the explained variance is shown along the vertical axis. The validated explained variance increases almost linearly until it reaches about 80% with 10 components. The explained variance in y is presented in Fig. 5.22. The validated explained variance is almost at its highest level at around 22% after only one component. It does increase some with two components, but more than two components do not contribute to any more explained variance in the target.

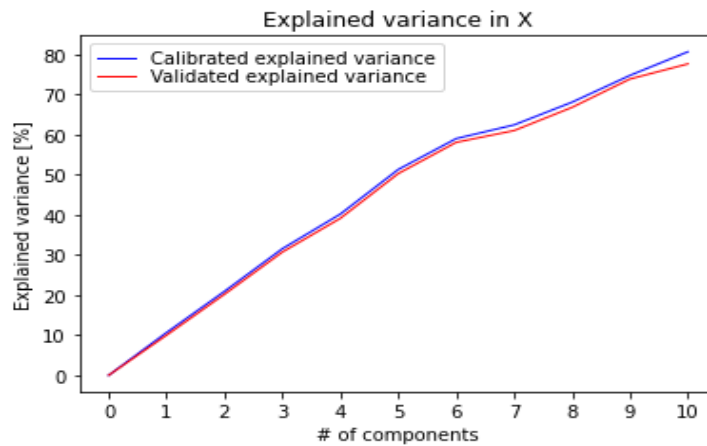


Figure 5.21: Explained variance in x as a function of the number of components used in the analysis..

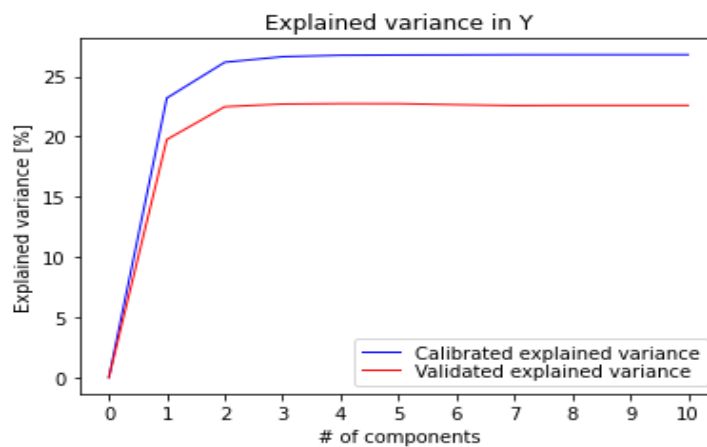


Figure 5.22: Explained variance in the target.

5.2.4 Outlier detection

The package `scikit-lego` automatically produces figures that can be used for identifying outliers. Decomposition based outlier detecting with PCA was used, and the analysis with two components is presented in Fig. 5.23. The figure on the left is the outlier detection shown through parallel coordinates, and the figure on the right shows a scatter plot of the outlier analysis. The figure on the right was only possible to produce when using two components, but using two components was not sufficient to separate out potential outliers, which are coloured in purple. From the figure on the left, it is clear to see that the feature `bl_apoe_E2/E2` contains some abnormal measurements. Analysing this further it showed to have an abnormal value due to the effects of standardisation, so the patient was not removed. Figures for the rest of the block can be found in the Appendix, Sec. 7.

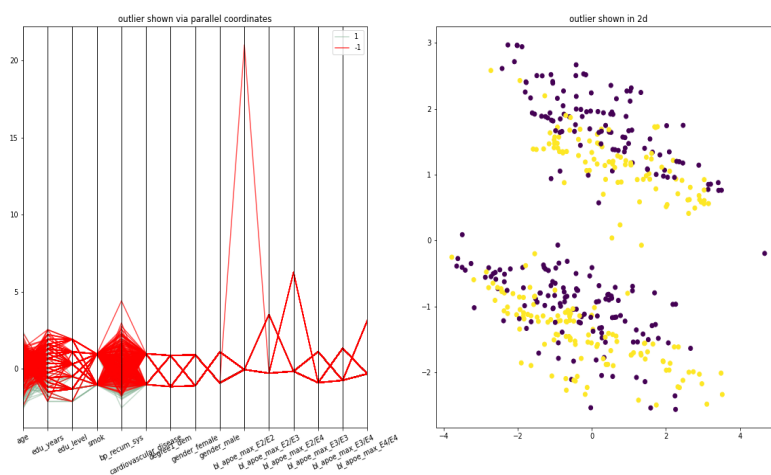


Figure 5.23: Results from decomposition based outlier detection using two components.

5.3 Classification

5.3.1 Baseline model

Step 5 in the workflow was establishing a baseline. Fig. 5.24 shows the first results in evaluating the classifier Logistic Regression on the complete dataset, the test dataset containing all the features. The figure shows the learning curve with the training and validation score as a function of the number of training samples used in learning. The scores are produced using a 5-fold cross-validation with an increasing amount of training samples. The Logistic Regression classifier was fitted with the best-suited hyperparameters found through grid search, which are

presented in Table 5.1. The figure shows that 110 samples are enough for this classifier to get a low enough difference between training- and test performance, and the highest validation score of around 0.5. The green area around the validation curve is calculated with one standard deviation above and below the mean value, indicating a high variance in the performance for the validation set.

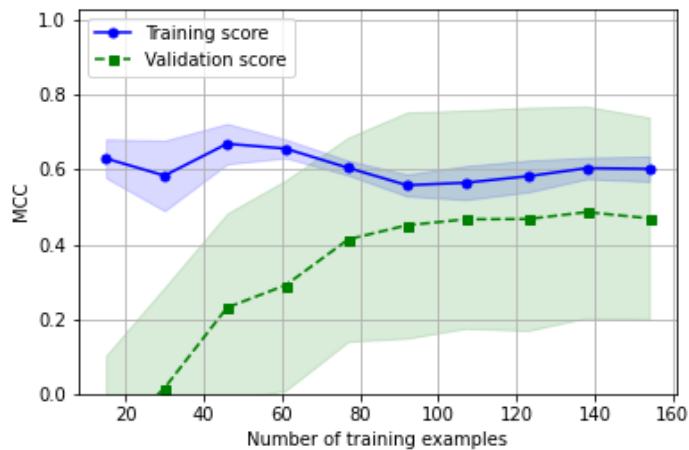


Figure 5.24: Validation curve for Logistic Regression.

The same figures for the Passive Aggressive Classifier are presented in Fig. 5.25. Fitted with the best-suited hyperparameters, Passive Aggressive Classifier shows an MCC score for the validation curve around 0.5, this for 90 samples and all the samples.



Figure 5.25: Validation curve for Passive Aggressive Classifier

Fig. 5.26 shows the results for Random Forest Classifier, with the highest validation score of around 0.36. The large gap between the training and validation score indicates that the model is overfitted on the training data. This can also be seen in the validation curve for the KNN classifier and the SVM classifier, shown in Fig. 5.27 and Fig. 5.28 respectively. For the KNN classifier, the highest validation score is just above 0.3, while for the SVM classifier it is just above 0.4.

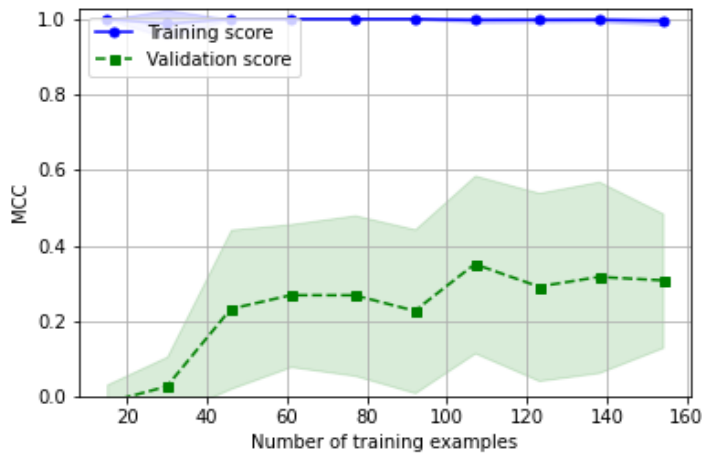


Figure 5.26: Validation curve for Random Forest classifier.

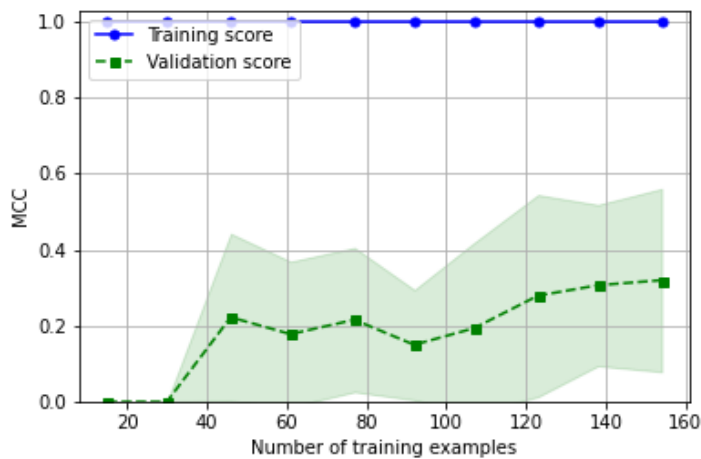


Figure 5.27: Validation curve for KNN classifier.

Table 5.1 shows an overview of the best-suited hyperparameters for each classifier found through gridsearch. The table also includes the highest obtained MCC score for each classifier, which was the resulting score using the given values for the hyperparameters. A short description of the listed hyperparameters was given in Table 4.3.

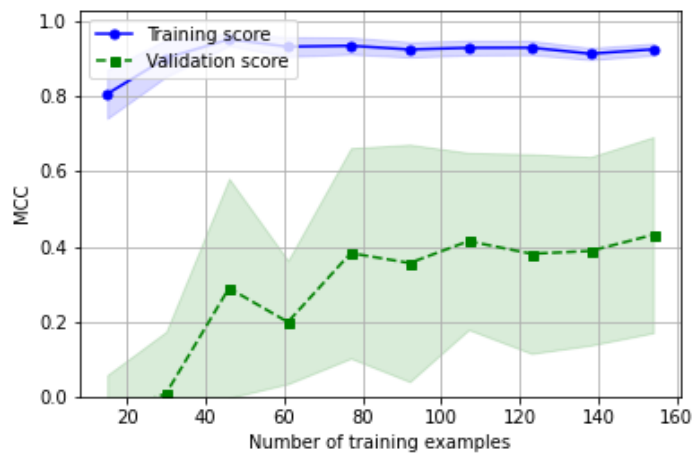


Figure 5.28: Validation curve for SVM classifier.

Table 5.1: List of algorithms, their hyperparameters and the value that gave the highest MCC score through gridsearch. The last column shows the MCC score.

Model	Hyperparameter	Value	MCC score
Logistic Regression	C solver	0.01 liblinear	0.4852
Passive Aggressive	C loss	0.0001 hinge	0.4248
Random Forest	n_estimators criterion max_depth max_features	40 gini 10 auto	0.4831
SVM	C kernel gamma	10 rbf 0.01	0.4112
KNN	n_neighbors weights algorithm	10 distance auto	0.4073

The different classifiers were also evaluated with different types of transformation of the data, as mentioned in Sec. 4.3. The classifiers were fitted with the best-suited parameters found through grid search, and the scores were found by calculating the mean of repeated stratified k-fold. The results are presented in Table 5.2. The first column is the mean MCC score using standardised data. The second column contains the MCC scores when the features from Block D were transformed using Yeo-Johnson transformation and the rest of the features were standardised. In the third column, the features in Block D were normalised to a range between 1 and 2 before power transformation using the BoxCox method, and the rest of the features are standardised.

The highest MCC score is obtained using the Logistic Regression classifier, with around 0.46 for both the different power transformation methods.

Table 5.2: Mean MCC scores for the different classifiers with repeated stratified k-fold, and for different types of transformation of the data.

Classifier	Standardised	Yeo-Johnson	BoxCox
Logistic Regression	0.4384	0.4551	0.4578
Passive Aggressive Classifier	0.4501	0.4320	0.4335
Random Forest	0.3531	0.3312	0.3464
KNN	0.2533	0.3173	0.3804
SVM	0.4483	0.4414	0.4425

5.3.2 Block-wise models

Some initial testing of the models for the block-wise analysis was done before the RENT analysis, but is not described as an own step in the workflow. Using repeated stratified k-fold with 5 splits and 10 and ten repeats, the average MCC score was obtained to get an overview of the performance of the classifier after the features were divided into blocks. Table 5.3 shows the results for the standardised training data. For Block D the features are power transformed using the Yeo-Johnson method. All of the classifiers show a score around 0 when being trained on the data in Block A, which is the same as random guess using the MCC metric. The highest MCC score is found in Block B, with a value of around 0.55 for both the Logistic Regression classifier and SVM.

Table 5.3: MCC scores for the different classifiers block-wise. The scores are the mean value obtained through with repeated stratified k-fold.

Classifier	Block A	Block B	Block C	Block D	Block E
Logistic Regression	0.0	0.5480	0.4059	0.3199	0.3918
Passive Aggressive Classifier	0.073	0.5028	0.4027	0.3243	0.3759
Random Forest	0.006	0.5174	0.3580	0.3123	0.3604
KNN	-0.007	0.3805	0.3403	0.3527	0.3106
SVM	0.0	0.5435	0.3993	0.3631	0.3755

5.4 RENT

RENT analysis for feature selection was performed on both the baseline data and block-wise. For the baseline data, this was part of step 5 in the workflow. For the block-wise data, the RENT analysis is step 7 in the workflow.

5.4.1 Baseline

The first step in the RENT analysis was finding a good balance between a high MCC score and a high fraction of features with weights equal to 0, which is done by finding the best combination of the parameters C and l1-ratio. Figure 5.29 shows the heatmaps that were produced. The value for the parameter C is along the horizontal axis, and the value for the l1-ratio is along the vertical axis. The first heatmap shows the MCC scores for the different combinations and the second figure shows the fractions of features having weight zero. The best combination was found with a value of C=0.1, and l1-ratio=0.5. This gives an MCC score of 0.3662, and a fraction of zeroes of 0.663, meaning that 66.3% of the variables were removed.

Figure 5.30 shows a plot of the selection frequency of the features. The features are along the horizontal axis, numbered by the index position. Along the vertical axis is the value for the cutoff parameter τ . Selecting a value for τ to 0.8 means only including the five features that lie above this value. Figure 5.31 shows the 500 elementary models used in RENT along the horizontal axis. The figure contains the value for the MCC score and the fraction of weights that are set to zero for the different models. The MCC score varies over the 500 elementary models, where the lowest score is -0.1 and the highest score is around 0.7.

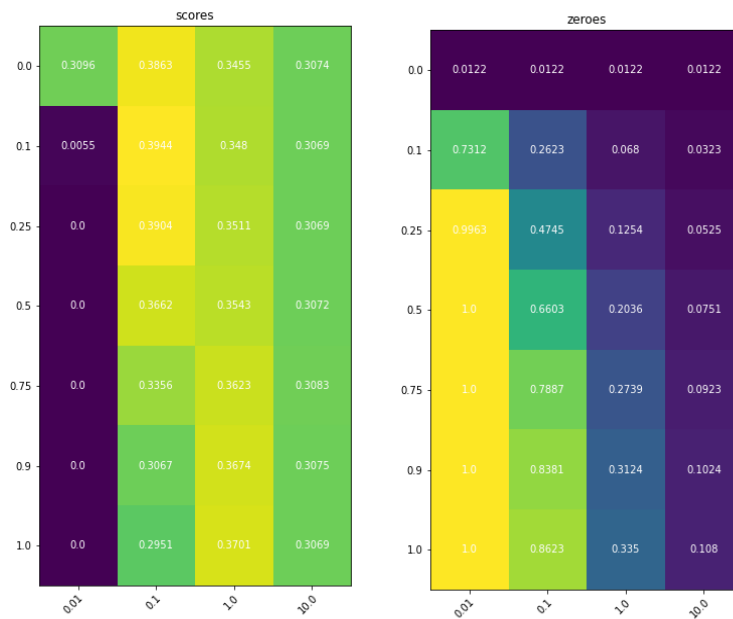


Figure 5.29: Heatmaps of the scores and zeroes from the RENT analysis. The value for the parameter C is along the horizontal axis, and the value for II -ratio is along the vertical axis.

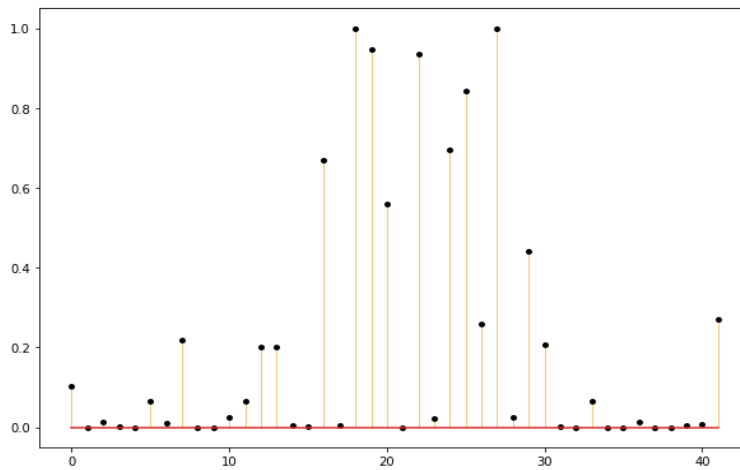


Figure 5.30: The selection frequency of the features. The features are numbered by index along the horizontal axis, and the vertical axis shows the value for the cutoff parameter τ .

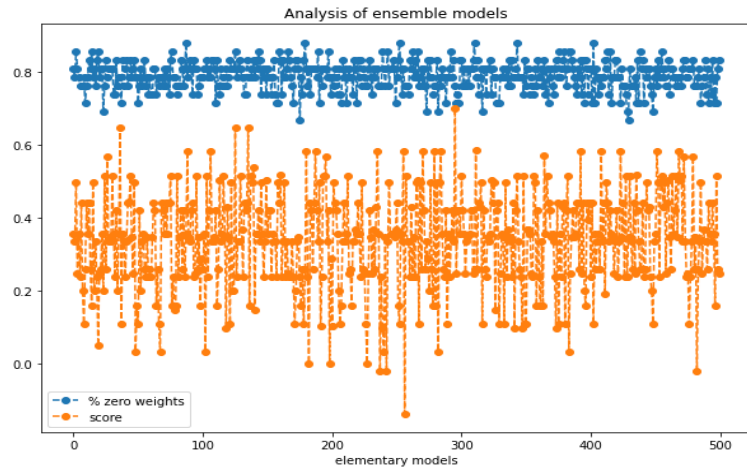


Figure 5.31: Value for scores and zeroes for the 500 elementary models used in RENT. The horizontal axis shows the different models, and the vertical axis is the value for scores and fraction of zeroes.

Figure 5.32 shows the MCC score as a function of different values for the parameters τ_1 and τ_2 . The value for the parameter is shown on the horizontal axis. Scores for the five different classifiers are shown in the figure. The highest score is obtained for a value of τ_1 at 0.35 for Logistic Regression, but for a higher value, the dimensions of the dataset are further reduced as it removes more features. The value for τ was selected to be 0.6, as the performance is reduced for several classifiers for larger values of τ_1 and τ_2 .

The features selected by RENT for some of the different values for τ are listed in Table 5.4. A reduced dataset was created using the features listed for the value for τ at 0.6.

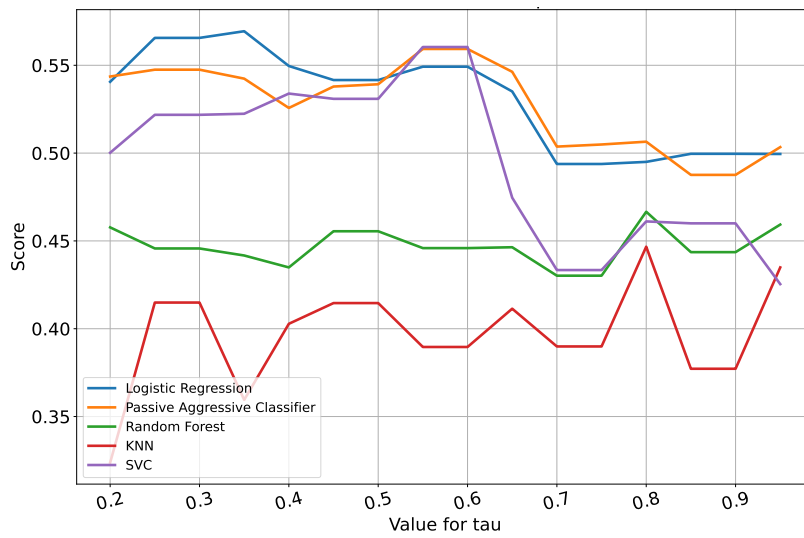


Figure 5.32: MCC scores for the different classifiers, and for increasing values for the parameters τ_1 and τ_2 .

Table 5.4: Listing of the selected features for some of the different values for the parameters τ_1 and τ_2 .

0.6	0.7	0.8	0.9
gender_female	bl_apoe_E3/E3	bl_apoe_E3/E3	bl_apoe_E3/E3
gender_male	bl_apoe_E4/E4	bl_apoe_E4/E4	bl_apoe_E4/E4
bl_apoe_E3/E3	mmse_total	mmse_total	mmse_total
bl_apoe_E4/E4	clock_score	clock_score	tmtb_sec
mmse_total	tmtb_sec	tmtb_sec	cowat_tscore
clock_score	cowat_tscore	cowat_tscore	cerad_recall
tmtb_sec	cerad_recall	cerad_recall	WMHo_rV
cowat_tscore	WMHo_rV	WMHo_rV	
cerad_recall	Meninges		
WMHo_rV			
LesO			
Meninges			

Classifiers were trained on a reduced dataset containing only the selected features, and an average of the MCC score was obtained by repeated stratified k-fold. The mean score and the standard deviation for the different classifiers are listed in Table 5.5. The highest obtained value for the MCC score was at 0.5604 with the SVM classifier.

Table 5.5: Scores for the different classifiers with repeated stratified k-fold using selected features from RENT with $\tau = 0.6$.

Classifier	Mean score	Std
Logistic Regression	0.5492	0.1340
Passive Aggressive Classifier	0.5593	0.1363
Random Forest	0.4381	0.1727
KNN	0.4302	0.1437
SVM	0.5604	0.1500

5.4.2 Block-wise

The same procedure used in the RENT analysis for the baseline data was used on the individual blocks. Figure 5.33 shows the heatmaps for scores and zeroes and for Block A. The highest score for this block was 0.0, which is the same as random guessing. Looking at the heatmap of zeroes, three of the parameter combinations that produce this score removes all the features. Due to this, and the low performance overall, there are fewer results presented for Block A.

Heatmaps of the scores and zeroes can be found in the Appendix, Sec. 7 for the rest of the blocks. The best combination of the parameters C and $l1$ ratio for the different blocks are listed in Table. 5.6.

Table 5.6: The parameters used in RENT for the different blocks.

Block	C	l1-ratio
B	1.0	0.9
C	0.1	0.9
D	1.0	0.9
E	0.1	0.5

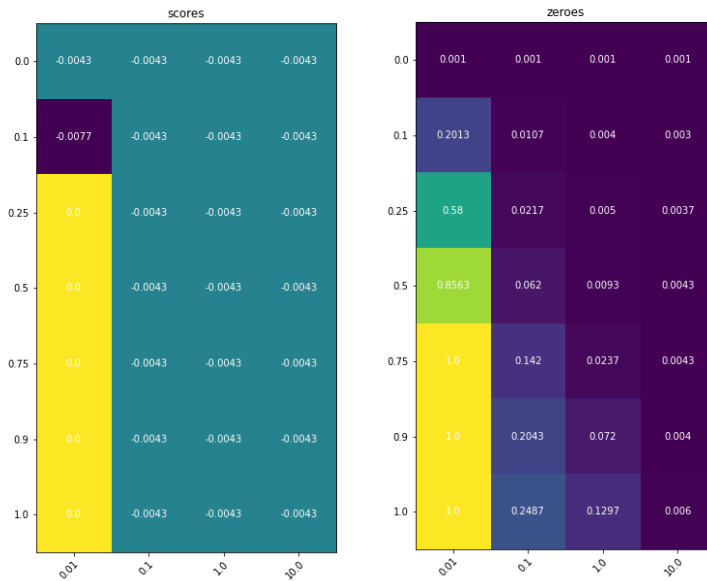


Figure 5.33: Heatmaps of the scores and zeroes for Block A from the RENT analysis. The value for the parameter C is along the horizontal axis, and the value for $l1$ -ratio is along the vertical axis.

Fig. 5.34, Fig. 5.35, Fig. 5.36 and Fig. 5.35 contain three plots from the RENT analysis for the different blocks. The plot on the top is the MCC score as a function of the parameter τ , where values for the parameter are given on the horizontal axis. The plot in the middle is of the selection frequency with values for the parameter τ is given on the vertical axis. The horizontal axis in this plot represents the index position of the features in the given block. The plot on the bottom is of the 500 elementary models used in RENT, with the fraction of features set to zero and the MCC score on the vertical axis. The value for τ was set to 0.8 for all the blocks.

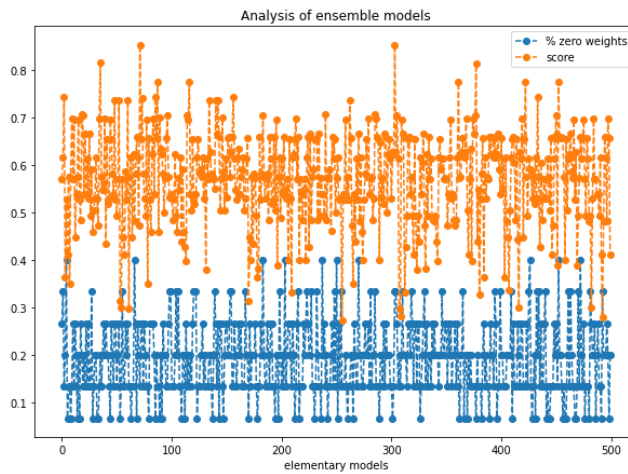
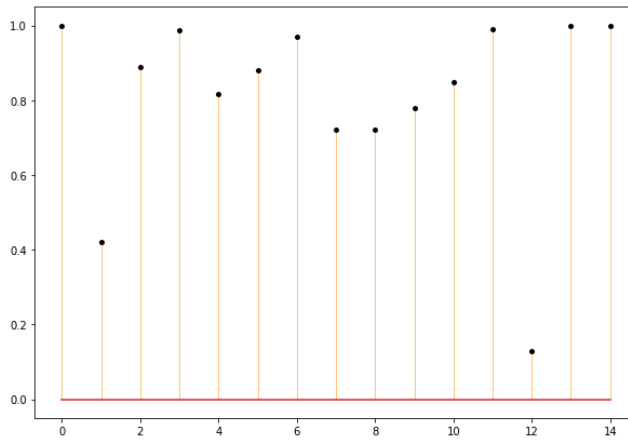
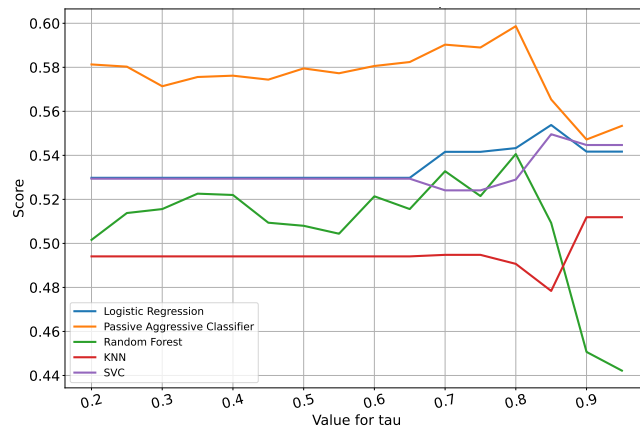


Figure 5.34: Block B: MCC score for different values of τ , selection frequency and elementary models.

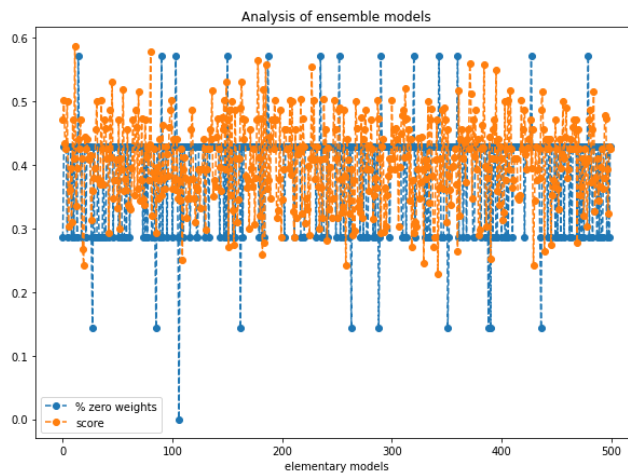
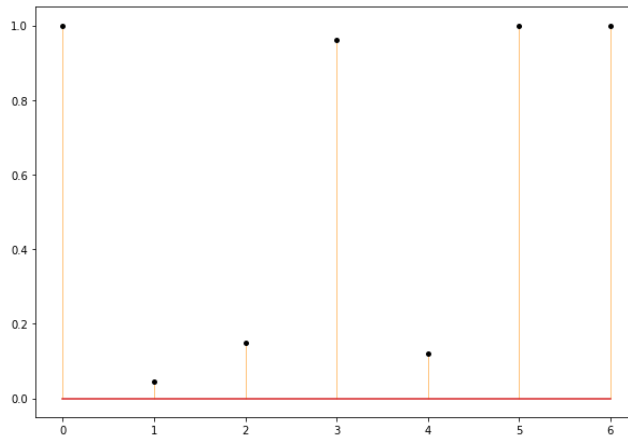
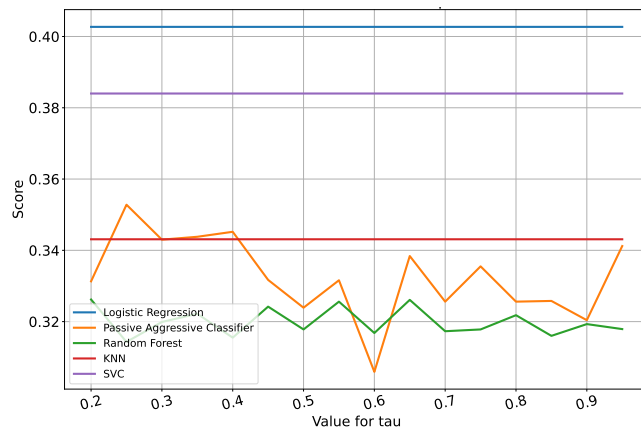


Figure 5.35: Block C: MCC score for different values of τ , selection frequency and elementary models.

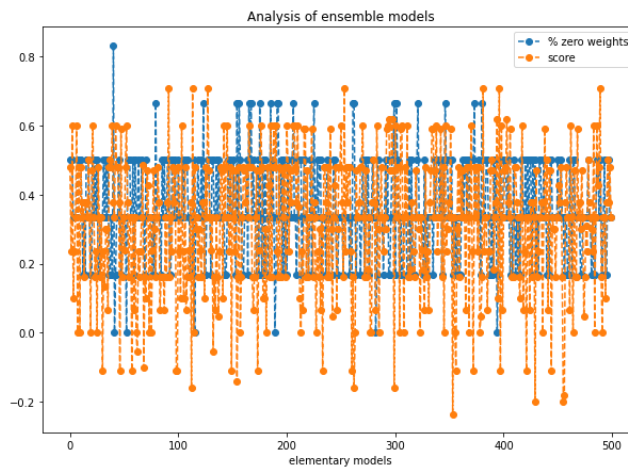
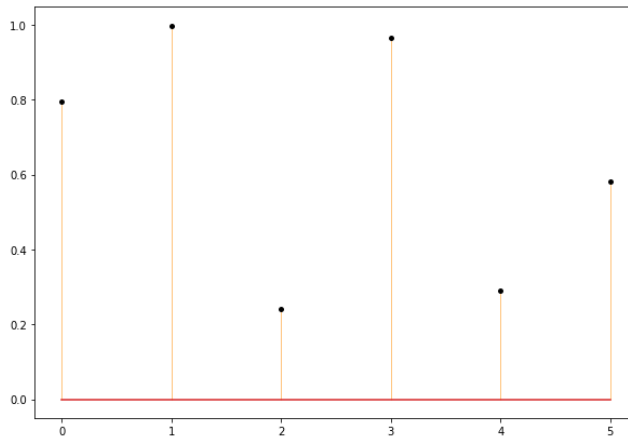
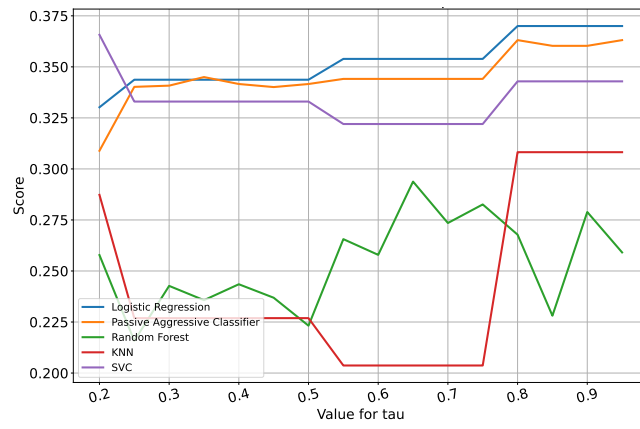


Figure 5.36: Block D: MCC score for different values of τ , selection frequency and elementary models.

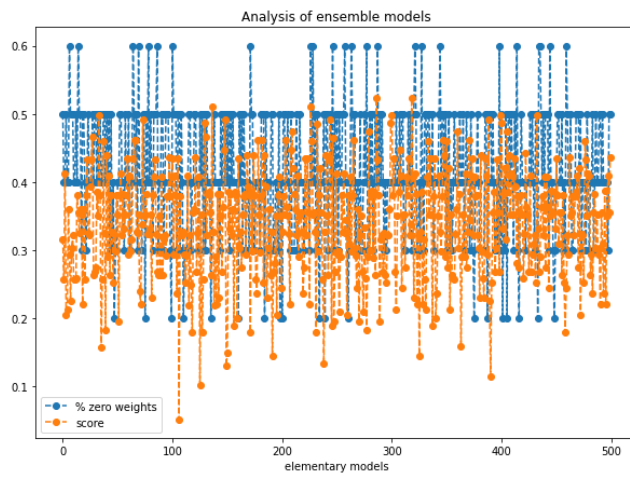
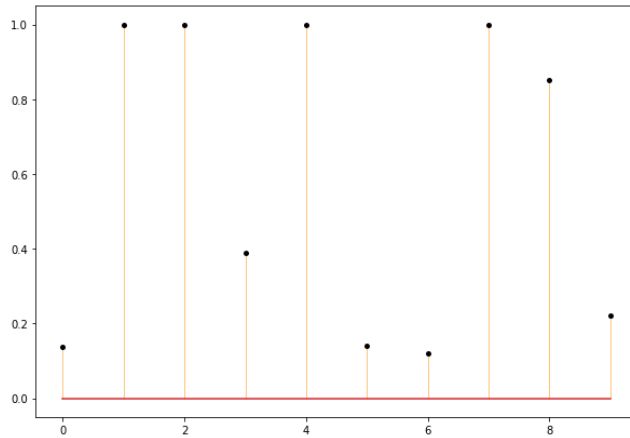
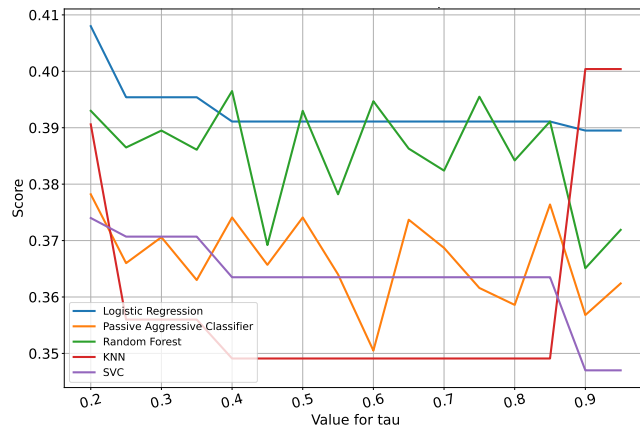


Figure 5.37: Block E: MCC score for different values of τ , selection frequency and elementary models.

The number of times a feature was selected by RENT over the 40 RENT feature selections was counted. The procedure using 40 RENT feature selections was described in Sec. 4.7. For Block B the results are listed in Table 5.7. The features age, bl_apoe_E4/E4 and bl_apoe_E3/E4 are selected by all the models. After One-hot-encoding, this is the block consisting of the highest number of features and also have the highest number of features selected by RENT. The last four features, gender_female, gender_male, bl_apoe_E3/E3 and edu_years, are selected less than 25% of the time. The only feature that was not selected once is bl_apoe_E2/E2.

Table 5.7: Block B: name of the feature selected and how many times it was selected across the 40 feature selections.

Feature	Times selected
age	40
bl_apoe_E4/E4	40
bl_apoe_E3/E4	40
bl_apoe_E2/E4	37
smok	30
degree1_dem	29
cardiovascular_disease	18
bl_apoe_E2/E3	16
edu_level	15
bp_recum_sys	11
gender_female	5
gender_male	5
bl_apoe_E3/E3	2
edu_years	1

Table 5.8 presents the results for Block C. As the feature vosp_score was only selected by one model, this indicates that it is not an important feature. The block consist of seven features and the features clock_score and tmta_sec were not selected by any of the 40 RENT feature selections.

Table 5.8: Block C: name of the feature selected and how many times it was selected across the 40 feature selections.

Feature	Times selected
mmse_total	40
cerad_recall	40
cowat_tscore	35
tmtb_sec	22
vosp_tscore	1

Table 5.9 present the results of features selected in Block D. The feature *WMHo_rV* represent white matter hyperintensity and was selected every time. LesO was chosen more often than the other three regions and belongs to the occipital region of the brain. The two features LesT and LesP were not selected by any of the 40 feature selections. Table 5.10 shows the features in Block E. The block contains 10 features, where four of them were not selected. This includes the features Anterior.hippocampus, Meninges, Br_35 and Br_36.

Table 5.9: Block D: name of the feature selected and how many times it was selected across the 40 feature selections.

Feature	Times selected
WMHo_rV	40
LesO	32
PSMD	15
LesF	5

Table 5.10: Block E: name of the feature selected and how many times it was selected across the 40 feature selections.

Feature	Times selected
Posterior_hippocampus	40
PHC	40
MISC	40
ERC	39
ColSul	20
Meninges_PHC	3

A score for the performance was obtained by predicting the test labels in the k-fold validation, using a Logistic Regression model. Logistic Regression was chosen since RENT uses this model with elastic net regularisation for feature selection. The model was trained with the dataset containing the features selected by RENT. The metric MCC was mainly used through this study, but other metrics for scoring the performance were calculated in this case in order to get an understanding of how good the performance is. This includes the metrics accuracy and F1-score. By flipping the labels, the F1-score for the negative class, in this case, 0, was also calculated. The average results are presented in Table 5.11. Fig 5.38 shows boxplots of the MCC scores for the 40 feature selections. The average is marked with an orange line. The highest score was obtained with the data in Block B. The boxplot also shows that the scores in Block D have the largest spread.

Table 5.11: Average scores over the 40 models. The table shows the MCC-, accuracy-, and the f1-score. The f1-score is calculated for both the positive and negative classes.

Block	MCC	Accuracy	F1-positive	F1-negative
B	0.5519	0.8310	0.6449	0.8880
C	0.4048	0.7612	0.5230	0.8401
D	0.3422	0.7741	0.4157	0.8586
E	0.3712	0.7475	0.4971	0.8310

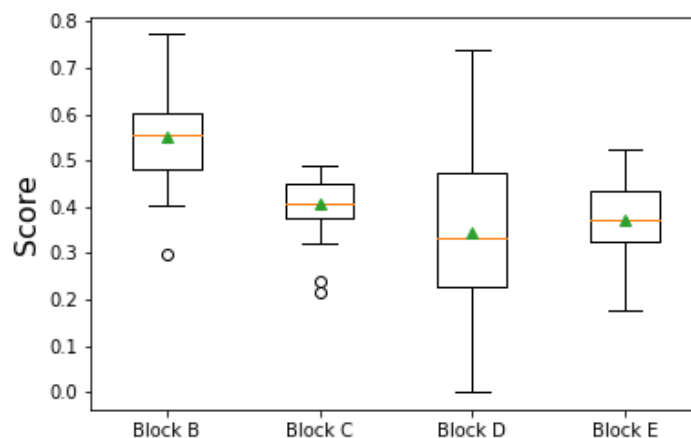


Figure 5.38: Boxplot of the MCC score for the 40 feature selections. The vertical axis represents the score, and the average value is marked with an orange line.

5.5 Ensemble

The first step towards creating the ensemble was the block-wise modelling on the selected features to find the best-suited classifier for each block, which is step 8 in the workflow. The selected features are presented in Table 5.12. It was unsure whether to include all the features selected by RENT or use a limit for how many times a feature was selected to determine if a feature should be included. The final selection was mainly based on results from repeated RENT, but also in collaboration with the neurologists. For Block B it was decided to remove features that were selected less than 10 times, which corresponds to 25%. For Block C vosp_tscore was only selected once and removed. For Block D all the features selected were included, even though LesF was only selected 5 times. For Block E, the features ColSul and Meningens_PHC were removed, even though ColSul was selected 20 times.

Table 5.12: The features from each block used in further analysis.

Block B	Block C	Block D	Block E
age	mmse_total	PSMD	ERC
bl_apoe_E4/E4	cerad_recall	WMHo_rV	PHC
bl_apoe_E3/E4	cowat_tscore	LesO	MISC
bl_apoe_E2/E4	tmtb_sec	LesF	Posterior_hippocampus
smok			
degree1_dem			
cardiovascular_disease			
bl_apoe_E2/E3			
edu_level			
bp_recum_sys			

For each block, the best-performing classifier along with the best-suited hyperparameter for each classifier was found through grid search. The classifiers and hyperparameters tested are listed in Table 4.3. Repeated stratified k-fold was used to evaluate the classifiers fitted with the best parameters. The best classifier for each block was selected by finding the classifier that gave the highest score with repeated stratified k-fold. The classifiers are listed in Table 5.13, along with the k-fold scores.

Table 5.13: The final models for each block, using the best result for the test score in repeated stratified k-fold.

Block	Best classifier	k-fold score
A	Passive Aggressive Classifier	0.0324
B	Passive Aggressive Classifier	0.5862
C	Logistic Regression	0.4035
D	SVM	0.3888
E	Logistic Regression	0.3893

The classifiers listed in Table 5.13 were used for predicting the labels for the test data, step 9 in the workflow. A heatmap of the predictions made for the individual block is shown in Fig. 5.39. The heatmap also includes the actual test labels in the last column. Dark blue indicates a label of 1 and light blue a label of 0. The predictions are sorted after the column that contains the actual test labels. This also gives a visualisation of the imbalance in the distribution of the two classes.

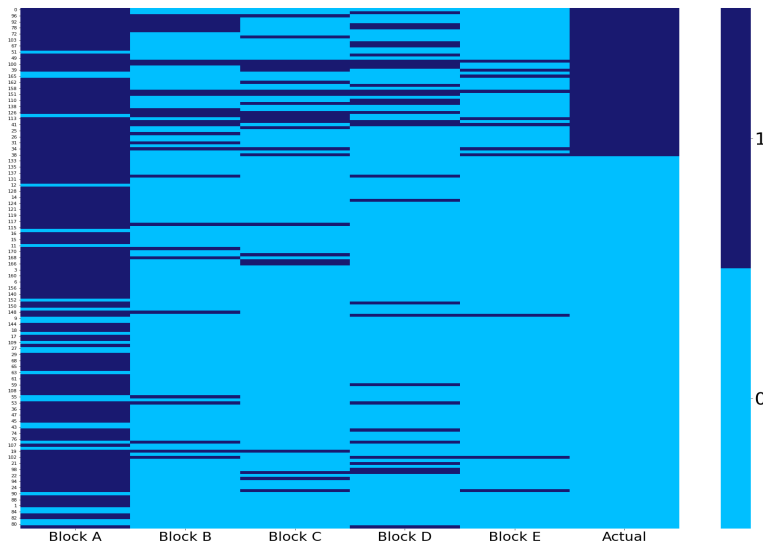


Figure 5.39: Heatmap of the final predictions of the test labels for each block. The last column represents the actual test label. Dark blue indicates a target value of 1, and light blue a value of 0.

Step 10 in the workflow is the final prediction of the class labels for the test data using predictions from each block. For the final prediction, a weighted average was calculated using the predictions from each block. The k-fold scores presented in Table 5.13 were used as weights for each block, meaning that the predictions from Block A had almost no influence on the final prediction, while the predictions from Block B got the most influence.

To determine whether the weighted average should be predicted as class label 0 or class label 1, different values for a threshold were tested. The initial value was set to 0.5, meaning that if the weighted average had a value over 0.5 the final prediction would be class label 1. This gave an MCC score of 0.3528 and an accuracy of 0.7674. The highest value for the MCC score was obtained by setting the threshold value at 0.3. This gave a value for the MCC at 0.4669 and accuracy of 0.7965.

Chapter 6

Discussion

6.1 The data

As already mentioned in Sec. 3 the target used in this analysis is a measure of the presence of amyloid-beta and does not necessarily mean that a patient has Alzheimer's disease. The classification in this thesis was between patients that are considered candidates for developing AD and patients that are not at risk of developing the disease. Out of the 1025 patients assessments investigated here, 698 belong to class 0 and 327 belong to class 1. There is an imbalance in the distribution of the classes, where 68% belongs to class 0. For the baseline study with 172 patient assessments, 72% belongs to class 0. Checking the distribution after splitting the data into training- and test set (see figure 4.2), the blocks of the training data contained 67% of the patient assessments belonging to class 0 for Block A, 73% for Block B, 68% for Block C, 74% for Block D and 68% for Block E. The distribution of the classes was similar for all the blocks and for the test data. The metric MCC was used throughout this study and is unaffected by the imbalance in the distribution of the classes.

In this analysis, the focus was on using the available data with no missing values in the features used. As described in Ch. 3 a patient can have several assessments. The number of assessments and the number of unique patients are given for each block in this chapter. Using only unique patients would have resulted in fewer samples to use for the analysis, but it is unknown how it would affect the final results. In this analysis, all the assessments were treated as individual samples. In the case of splitting the data into training- and test, this approach might not have been optimal, as some of the patients appeared in both the training and the test data with different assessments. The data used for training and the data used for test should be independent of one another and using the same patient for both sets could cause the information to leak into the test set.

Using only the first assessment for each patient results in 789 patients before removing any missing values. 646 of these patients have a value for the target feature and 135 patients have complete data for all the features used in this study, meaning that the test set used for the baseline analysis would have contained 135 samples when using unique patient ID's, compared to 172 when using patient assessments. Due to the time limit, the analysis was not repeated using only the first assessment for each patient, but it could be of interest to test in further analysis.

6.2 Data preprocessing

Power Transformation

Different transformations of the data were tested, where the main types were listed in Sec. 4.3. Given that the range of the values did differ in some of the features, standardisation was performed on all the features. When it comes to power transformation, the Yeo-Johnson method was preferable over the BoxCox method. This is because it was possible to apply directly to the standardised data which typically contains negative values. The BoxCox method does not work for negative numbers, so the data was chosen to be normalised to a range between 1 and 2. During testing of what features to power transform, the results were different for the individual classifiers. Only the Logistic Regression model showed better performance on only standardised data when working with the test set in the baseline modelling. The rest of the classifiers increased their performance when the features in Block D were power transformed. It was therefore concluded to continue to work with the features in Block D with Yeo-Johnson power transformation. The effect of the transformation is shown in Fig. 5.1 and Fig. 5.2 for the LesP features.

Combining features

At the beginning of the analysis, the Les features did consist of 16 different features. LesF, LesO, LesP, LesT were all divided into four regions, shown in Table 3.4. When analysing the results from the selected features from RENT, it was suggested by the neurologist who owns the data that the regions should be combined. Part of this suggestion was because the feature WMHo_rV already is a combined feature for the four different regions of the brain. This was not known at the beginning of the study, and for an equal assessment of the features through RENT, the Les features were combined.

An alternative would have been to separate the WMHo_rV feature into regions. After combining the Les features, the average MCC score obtained through repeated RENT increased by almost 0.1. This shows that the separation did not contribute to any useful information for the model, but may have increased the noise.

Framingham Risk Score

Later in the study, a feature was added with a score related to Framingham Risk Score. The score combines features to calculate the risk related to developing cardiovascular vascular disease [64]. Since cardiovascular disease and AD are related, the score is also useful when analysing patients with AD. When added to Block B, the features age, gender, smok, cardiovascular_disease, and bp_recum_sys could be removed as they are all involved in the score feature. The score feature also includes information related to diabetes and body mass index (BMI). Since the feature was added late in the analysis, only part of the total workflow was repeated with the use of this feature. The block-wise RENT analysis gave similar results for the average MCC score. It could be useful to use the score in further analysis due to the fact that it is a score combining several factors related to the risk of developing cardiovascular disease. The Framingham score could be of high importance when analysing patients with the risk of developing AD but was not appropriately tested in this study.

6.3 Explorative analysis

Correlation between features

The correlation between the features in the different blocks was included in the explorative analysis, which is step 2 in the workflow presented in Fig. 4.1. For Block B, the heatmap of the correlation coefficient was shown in Fig. 5.10. The heatmap showed a high correlation between the features edu_level and edu_years. They were both included in the analysis as the number of years of education does not necessarily contribute to the same level of education. As seen in Table 5.7 the level of education appeared as a more important feature than the years of education. The feature edu_years was removed when making the final dataset. In Block C the features tmta_sec and tmtb_sec show a high value for the correlation coefficient, as seen in the heatmap presented in Fig. 5.11. tmtb_sec was selected 22 times out of the 40 RENT feature selections, as listed in Table 5.8. The feature tmta_sec was not selected once and removed from the dataset. When looking at the plot of the loadings in PCA, given in Appendix, Sec. 7, the two features tmta_sec and tmtb_sec are correlated. This indicates that they contribute to the same information.

Chemometrics

The explorative analysis using PCA and PLSR was performed in both step 2 and step 6 presented in the workflow. PCA and PLSR are both techniques for exploratory analysis. PCA is an unsupervised technique, while PLSR is a supervised technique and the target is included in the analysis. For the block-wise analysis, the methods for exploratory analysis was first applied to all the patient assessments, which is step 2 in the workflow. The results presented in Sec. 5.2 are from step 2. From the plot of the scores, none of the techniques showed any indication of patients that could be considered as outliers. From the analysis with PCA, Block B shows clustering of the patients into two groups. When compared to the plot of the loadings, it indicates a separation of female and male patients. Whether a patient is female or male appear as an essential feature in the baseline analysis using RENT. Including the target in the analysis with PLSR, reduced the contribution related to gender in the direction of the second component, while the features `smok`, `cardiovascular_disease`, `edu_years` and `edu_level` showed higher contribution compared to PCA.

To ensure that the systematic variation in the data did not change after splitting into training and test, the exploratory analysis was performed again on the patient assessments in the training data, which is step 6 in the workflow. The explained variance as a function of the number of components was compared. Results from the PLSR analysis are presented in Tables 6.1, 6.2, 6.3, 6.4 and 6.5 for the different blocks. The tables include the cumulative calibrated explained variance for the components for both the data and the target. The rows including 'train' are results for the analysis only including the training data. Comparing the results from all the samples and the samples in the training set, there is not much difference in explained variance across the number of components. It can be assumed that the systematic variance is similar and did not change after splitting the data.

Table 6.1: Block A: Cumulative explained variance using 5 components.

Component:	1	2	3	4	5
Variance X	30%	45%	60%	80%	100%
Variance X, train	29%	47%	63%	82%	100%
Variance Y	3.8%	4%	4%	4%	4%
Variance Y, train	4.3%	4.5%	4.5%	4.5%	4.5%

Table 6.2: Block B: Cumulative explained variance using 6 components.

Component:	1	2	3	4	5	6
Variance X	10%	20%	30%	38%	50%	60%
Variance X, train	9%	20%	30%	39%	49%	56%
Variance Y	23%	27%	27%	27%	27%	27%
Variance Y, train	27%	32%	33%	33%	33%	33%

Table 6.3: Block C: Cumulative explained variance using 6 components.

Component:	1	2	3	4	5	6
Variance X	45%	50%	60%	70%	80%	90%
Variance X, train	44%	55%	65%	72%	83%	94%
Variance Y	16%	21%	21%	21%	21%	21%
Variance Y, train	16%	21%	21%	21%	21%	21%

Table 6.4: Block D: Cumulative explained variance using 6 components.

Component:	1	2	3	4	5	6
Variance X	70%	75%	82%	90%	95%	100%
Variance X, train	75%	79%	89%	95%	97%	100%
Variance Y	12%	13%	13%	13%	13%	13%
Variance Y, train	14%	16%	16%	16%	16%	16%

Table 6.5: Block E: Cumulative explained variance for 6 components.

Component:	1	2	3	4	5	6
Variance X	30%	60%	65%	75%	80%	85%
Variance X, train	35%	60%	67%	76%	80%	84%
Variance Y	11%	13%	15%	16%	16%	16%
Variance Y, train	14%	17%	18.5%	19%	19%	19%

6.4 Classification

Classifiers

There exist many different classifiers, and choosing the best classifier depends on the problem. In this thesis, five different classifiers were used during the analysis to compare performance. The first evaluation of the different classifiers was in the baseline analysis, step 5 in the workflow. Referring to the overview presented in Table 5.2, Logistic Regression, Passive Aggressive classifiers and SVM were the best performing classifiers. Common for all the three classifiers is that they have an adjustable regularisation parameter, C . As described in Sec. 2.3, the Passive Aggressive classifier belongs to a different category of machine learning algorithms than the other classifiers, which is Online learning algorithms. It uses incremental learning and is commonly used for classifying massive streams of data. The Passive Aggressive classifiers performed well on the data used in this study, even though it is mostly used for online streaming and large datasets. Similar to the SVM classifier, it uses a margin to separate the two classes. SVM can solve linear and non-linear problems by changing the parameter for the kernel, and both linear and non-linear were tested. Through grid search, the non-linear kernel was found to be the best performing for the baseline data. When comparing results for the used types of transformation of the data, the scores vary some for the different classifiers. KNN is the classifier that shows the highest increase in performance when using power transformations compared to just standardisation.

When analysing the results from the block-wise analysis, presented in Table 5.3, none of the classifiers performs well on Block A. This is discussed in further detail in the following section. Excluding the results from Block A, the difference between the minimum and maximum scores was calculated. Table 6.6 shows the minimum and maximum scores for the different classifiers when their performance is measured for the different blocks. The difference in percentage is calculated as the difference between the values divided by the average. Table 6.7 shows the minimum and maximum scores for the different blocks obtained by the different classifiers.

Table 6.6: Comparing the minimum and maximum scores obtained block-wise for the different classifiers.

Classifier	Min. score	Max score	Difference
Logistic Regression	0.3199	0.5480	52.6%
Passive Aggressive Classifier	0.3243	0.5028	47.8%
Random Forest	0.3123	0.5174	49.4%
KNN	0.3106	0.3805	20.2%
SVM	0.3631	0.5435	39.8%

Table 6.7: Comparing the minimum and maximum scores for the different classifiers in each block.

Block	Min. score	Max score	Difference
B	0.3805	0.5480	36.1%
C	0.3403	0.4059	17.6%
D	0.3123	0.3631	15.0%
E	0.3106	0.3918	23.1%

When looking at the difference between the minimum and maximum score in the two tables presented, there is a larger difference between the blocks than within the blocks. This could indicate that the information in the blocks affects the score more than the choice of classifier. The biggest difference between the worst- and best performing classifiers is in Block B. This is also the block that obtains the highest scores. KNN is the model that performs worst across all the blocks and is also the classifier with the lowest difference between the worst- and best performance. As mentioned in Sec. 4.6 and in the beginning of this chapter, the dataset is imbalanced, with the highest number of patients belonging to class 0. The imbalance may affect the performance of the KNN classifier, as it may have a bias towards the majority class, which is class label 0. The KNN classifier is sensitive to outliers and missing values, but the exploratory analysis did not reveal any obvious outliers and all the missing values were removed. The KNN algorithm always performs better with power transformed data, so it seems to be sensitive to skewed data. Skewed data could have a negative effect when it comes to calculating the distance to the neighbouring points.

The final evaluation of the performance takes place when creating the ensemble of best performing classifiers for each block. This is done when using the selected features after the RENT analysis and is step 8 in the workflow. Block B had overall the best results throughout this analysis, with the highest MCC score of 0.5862 with the Passive Aggressive classifier. The best-suited classifier for each block was decided by using the highest MCC score obtained through repeated stratified k-fold.

Blocks

Both the features used in this study and how they were divided into blocks were based on suggestions from the neurologist that provided the data. Adjustments and different approaches were tested throughout this study.

Block A consists of only one feature, subject group. In the original dataset, this feature had seven categories, three of the different types of symptom groups and four control groups. As shown in Table 5.3 all the classifiers showed an MCC

score equivalent to random guessing. Different approaches to how to combine the categories were tested. One of them was combining them into four categories - two types of symptom groups and two types of control groups. Another approach was making two categories - a symptom group and a control group. All of the approaches gave similarly poor results when evaluating the performance of classification. Whether a patient belongs to a symptom- or control group should be of significance when predicting the target, and it is not clear why none of the models finds any information in this block.

Block B contains factors related to environment and heritage and has throughout this study been the block that has obtained the highest MCC scores. As having the form e4 of the APOE gene is related to a higher risk of developing AD, some testing was done by only investigating the presence of this form. This was done by creating a feature that had the value 1 if e4 was present in the APOE pair, and removing the features regarding the other forms - e2 and e3. This did not improve the performance, so all the combinations were included.

Block C contains cognitive tests and no changes were done to the features in this block. Some of the tests are correlated and it was suggested that the features cerad_recall and vosp_tscore should be calculated to the same scale and only use the feature with the lowest score. Due to the time limit, this was not tested in this study. Only the feature cerad_recall was used when training classifiers in the final prediction, as vosp_tscore was only selected once in the repeated RENT analysis.

Block D contains measurements of white matter hyperintensity load, lesion in different parts of the brain and vascular damage in the brain. During a meeting with the neurologists, this block was mentioned as the one with the most significant features related to AD. The average MCC score from repeated RENT was 0.3422, but the boxplot shown in Fig. 5.38 shows a wide spread in the scores. This block showed varying results throughout this study. As seen in Fig. 5.6 and Fig. 4.2 Block D contains a lot of missing values and ends up with only 112 patients in the training data, and there are more patients in the test data. Using repeated stratified k-fold, the classifiers are trained on a low number of patients, which could explain the high variation in the results. Training data with few samples can also make it harder for the models to capture the relevant patterns in the data.

Block E contains features measuring subcortical brain structures, and different approaches were also tested with the features in this block. The features used are combined measurements for the left- and the right side of the brain. The original dataset also contains the features separated, resulting in 20 features in the block. Different approaches for the correction of the scanners used in obtaining these images are also available in the original dataset. Two different corrections were tested, both with the combined and separated features, resulting in four different versions of this block. There was no significant difference in the corrections, but the combined features gave a higher MCC score compared to the separated ones.

6.5 RENT

Parameters

Referring to the workflow in Fig. 4.1, RENT for the baseline analysis is step 5 and for the block-wise analysis step 7. For both the baseline- and block-wise analysis, heatmaps were produced in order to manually select the best suitable parameters. If a block contains few features, it may be desirable to find a combination of the parameters that give a higher value for the MCC score, even though the fraction of features set to zero is not at its highest value. For blocks with many features, the focus could be on a combination of parameters that give a wanted reduction of the features but still a reasonable MCC score. The parameters were manually chosen in collaboration with the supervisors involved in this thesis.

The default value for the cutoff-parameters τ_1 and τ_2 is 0.9. After visualising the scores as a function of the value for this parameter, this showed to be a too high value in the baseline analysis. As shown in Fig. 5.32, several classifiers show a decrease in the MCC score after 0.6. The parameter is also involved in the decision of how many features are finally selected. By visualising the MCC score as a function of the parameter, it was possible to choose the value that gave the highest MCC score.

Referring to Fig. 5.34, $\tau_1 = 0.8$ shows to be the best value for the parameter in Block B. For Block C, the value for τ was also set to 0.8, but a value between 0.2 and 0.9 would have resulted in the same selection of features. This can be seen from the plot of selection frequency in Fig. 5.35, which shows a clear boundary between which features are selected by the elementary models in the RENT analysis. 4 features are above 0.95, while 3 of the features lie below 0.2. The plot of the MCC score as a function of τ shows a consistent MCC score for Logistic Regression, KNN and SVC. Passive Aggressive Classifier and Random Forest show some variation in the MCC score even though the selected features are the same, which can come from internal randomness in the classifiers.

For Block D, the best choice for parameter τ is 0.8, as shown in Fig. 5.36. For Block E, the value was set to 0.8, but 0.85 was also considered. The plot of selection frequency in Fig. 5.37 shows the case of a feature that lies on the value 0.85, which means that the feature is selected with a fraction of 85% over the 500 elementary models. The MCC score for KNN significantly increases when removing this feature, while the score decreases some for SVM when removing the feature. Again, the MCC score for Passive Aggressive Classifier and Random Forest vary more than for the other classifiers.

The final choice for the cutoff-parameter τ was decided in collaboration with the supervisors involved in this thesis.

Selected features

Looking at the selected features in the baseline analysis for the value of τ equal to 0.6, presented in Table 5.4, 4 of the features are from Block B, 5 of the features are from Block C, 2 from Block D and 1 from Block E. None of the categories from Block A appear among the selected features. As mentioned in Sec. 1, the e4 form of the APOE gene is related to an increased risk of developing AD, and it was unexpected that `bl_apoe_E3/E3` came out as more significant than `bl_apoe_E4/E3`. Comparing the features that are from Block B to the block-wise analysis of selected features, shown in Table 5.7, there are several differences. The feature `age` does not appear as an important feature in the baseline analysis, while `gender` is more significant than for the block-wise analysis. In the baseline analysis, most of the selected features are from Block C, indicating that the cognitive tests are of importance for the classification task. Both the baseline- and block-wise analyses seem to agree on which of the tests are important, except for the `clock_score`. From the PCA loadings plot, the features `clock_score` and `vosp_score` are correlated, indicating that they contain the same information. The plots are shown in Appendix, Sec. (7). Still - `vosp_score` was only selected once over the 40 models. There are two features from Block D in the baseline analysis, `WMHo_rV` and `LesO`. These two features came out as the most important features in Block D, as shown in Table 5.9. There is only 1 feature selected from Block E, `Meninges`, which shows a low count for Block E, referring to Table 5.10. When all the features are combined, the features from Block E do not come out as significant compared to the other features. It is important to note that the baseline analysis was done with only one RENT model. For a more robust analysis of the feature importance, repeated RENT should be used, as in the case of the block-wise analysis.

The final features selected for creating a new dataset, shown in Table 5.12, were mainly based on results from repeated RENT but is also selected after a discussion with the providers of the dataset. It was unsure if all the features selected by RENT should be included, or if a limit should be set for how many times a feature was selected to determine if a feature should be included. The neurologists could provide suggestions on what features should be included or removed based on background knowledge of AD. This is why LesF was included in the final dataset even though it was only selected 5 times, and the feature ColSul was excluded despite the fact that it was selected 20 times.

Classifiers and scores

RENT uses Logistic Regression with elastic net regularisation for feature selection. The features selected through RENT could therefore be more compatible with the Logistic Regression classifier compared to the other classifiers, and this could give the classifier an advantage when it comes to scoring the performance.

For the baseline analysis, Logistic Regression, Passive Aggressive Classifier and SVM show similar performance for repeated stratified k-fold when using all the features, as shown in Table 5.2. Using the features selected by RENT, the performance of the classifiers was evaluated by using repeated stratified k-fold. The scores are presented in Table 5.5. All classifiers showed an increase in performance when using features selected by RENT. For the baseline analysis, Table 6.8 gives an overview of the average MCC score using all the features compared to the average MCC score using features selected by RENT. The increase in MCC score after the RENT analysis is given in the last column as a percentage.

Classifier	All features	Selected features	Change
Logistic Regression	0.4551	0.5492	20.7%
Passive Aggressive Classifier	0.4320	0.5593	29.5%
Random Forest	0.3312	0.4381	32.2%
KNN	0.3173	0.4302	35.6%
SVM	0.4414	0.5604	26.9%

Table 6.8: Comparing average MCC score with for all the features in the test data and using the features selected by RENT. Since performance for all the classifiers did increase using the selected features, the change is given as an increase in percentage.

Before feature selection, Logistic Regression shows the highest value for the MCC score for Block B, Block C and Block E, referring to Table 5.3. The average MCC score for repeated RENT, shown in Table 5.11, is obtained using a Logistic Regres-

sion for prediction. Comparing the MCC score for Logistic Regression before and after feature selection with RENT does not show improvement in the performance. For Block B, the MCC score increases by 0.71% after using selected features by RENT. For all the other blocks, the performance slightly decreases. The reduction of features and increase in interpretability makes the minor reduction in performance acceptable. The procedure for obtaining the scores are not the same, so the comparison is not as accurate as for the baseline analysis. Repeated RENT was used for more robust results for the feature selection, so only the Logistic Regression classifier was used to get an indication of the performance over the 40 feature selections. The repeated RENT procedure was not used in the baseline analysis. After creating new datasets with the selected features, the performance of the other classifiers was again evaluated through grid-search.

Squared- and combined features

RENT also includes an option to define a parameter named *poly*. By default, it is set to 'OFF'. When defining the parameter to 'ON', the interaction of features and squares of features are included. If the parameter is set to 'ON' new features are created and could result in more features than originally used, and is not a useful approach if the performance does not increase significantly. Initial testing with the parameter set to 'ON' was done early in the block-wise analysis. Creating squared and interactions between features from the results did not improve performance compared to the analysis done with the parameter set to 'OFF'. Running the RENT analysis with the parameter set to 'ON' comes with a computational cost, and the resulting squared- and combined features have to be created for further analysis. Due to the fact that the performance showed similar results using this technique and that the procedure was more time consuming, further analysis was done with *poly* set to 'OFF', and the results from the combined features are not included.

Misclassified patients

The summaries for the 40 RENT feature selections used in repeated RENT was aggregated and the misclassified patients were analysed. This was done in step 7 in the workflow. The summary contains an overview of the percentage of patient assessments that were misclassified across the 500 elementary models. Adjusting for the fact that an assessment could be in either the RENT-set or the test-set for the 40 different RENT feature selection, the percentage of misclassification was aggregated. A list of misclassified patients was created. PCA analysis was done with the reduced dataset of selected features and different colours for the correctly- and incorrectly classified patients.

The results are presented in Fig. 6.1, Fig. 6.2, Fig. 6.3 and Fig. 6.4 for Block B, Block C, Block D, and Block E respectively. The figures show the PCA scores with the first component along the horizontal axis and the second component along the vertical axis. The correctly classified patients have a green colour, while the incorrectly classified patients have a red colour. None of the blocks shows any sign of obvious clustering between the correctly- and incorrectly classified patients, and no further analysis was done. For further work, it could be interesting to do further analysis on these results to find out what separates the two groups of patients.

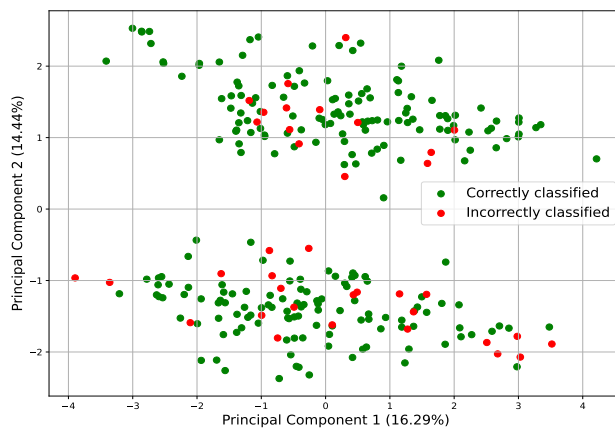


Figure 6.1: PCA scores for Block B.

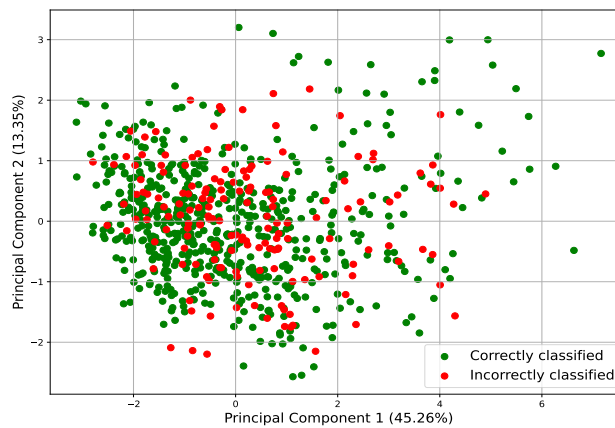


Figure 6.2: PCA scores for Block C.

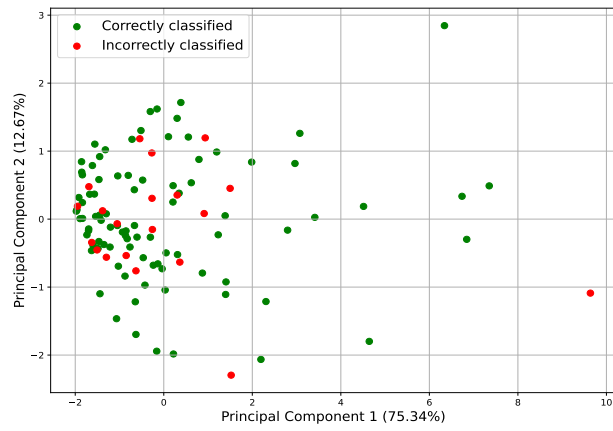


Figure 6.3: PCA scores for Block D.

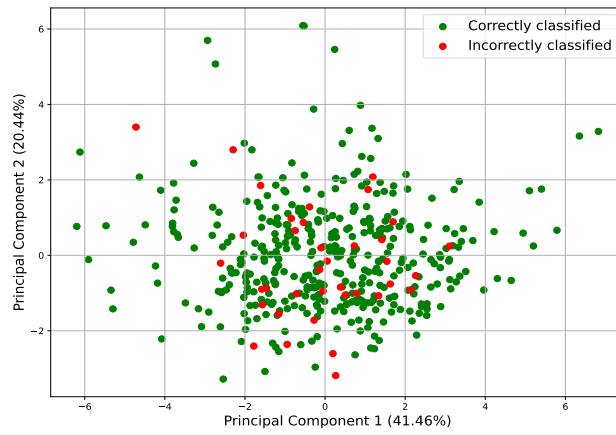


Figure 6.4: PCA scores for Block E.

6.6 Weighted Average

The three last steps in the workflow include finding the best classifier for each block, predicting test labels block-wise and combining the predictions from each block into a final prediction. Several different strategies were tested when creating an ensemble using predictions from each block, and a final prediction using soft majority voting. This includes how the final classifier for each block was selected, whether to use best results from grid-search or k-fold scores. Different weights for the prediction of the best model for each block were tested, and also the threshold for when the final label should be predicted to be the value 0 or 1. The procedure and results presented in Sec. 5.5 was what produced the highest value for the final MCC score. Further adjustments made could have improved the final score, including different approaches for finding the best classifier and how the predictions from the different blocks are combined to make the final prediction. The final MCC score was lower than anticipated. Although the separation of the features into blocks did contribute to more patients being included in the analysis, it was hoped that the use of blocks would contribute to better performance when finding individually adapted classifiers for each block. It is not fair to compare the final MCC score to the score obtained for the baseline analysis. Predicting labels for unseen data typically gives a lower score than the score obtained through repeated stratified k-fold.

6.7 Further work

This section concludes the discussion chapter. Even though some have been mentioned already, this section contains an overview of further work that can be done.

1. Analysis using only the first assessment for each patient.
2. Repeated RENT analysis for more robust analysis of selected features when using all the features combined.
3. Further analysis regarding the misclassified patients.
 - See if any clear differences can be found that separates these from the patients that are classified correctly.
4. Using dynamic ensembles for final prediction.

Chapter 7

Conclusions

Overall, this study showed that the block containing the factors related to environment and heritage gave the highest MCC score when classifying between patients with the presence of amyloid-beta and patients without. The average score with repeated RENT for this block was similar to the score for the baseline analysis with RENT. With all the features combined, the factors related to environment and heritage, and cognitive tests were identified as the most important features. Among the features regarding images of the brain, the features measuring white matter hyperintensity load and lesion in the occipital lobe came out as important for the classification task in this study.

Using the features selected by RENT improved the performance of all classifiers in the baseline analysis, even though the features came from different sources. The performance remained approximately the same for the block-wise models, but the RENT analysis resulted in a reduction of features and easier interpretation. Through a repeated RENT procedure it was possible to aggregate the results over the models to investigate how many times a feature was selected. Dividing the features into blocks resulted in a larger number of patient assessments block-wise compared to the patient assessments that had complete data for all the features, meaning that the classifiers had more samples to use for training. Compared to the baseline, the MCC score did not improve when creating a weighted ensemble with block-wise predictions.

Overall, the predictive performance achieved in this study was not extremely good. The highest obtained MCC score was 0.5862 using the Passive Aggressive classifier trained on data from Block B. Still, there is a fair amount of information contained in the data that can be used for interpretation. The features selected through the repeated RENT analysis provided insight into the importance of all features and are in accordance with the domain knowledge of the neurologists working with the patients.

Bibliography

- [1] “What to know about alzheimer’s disease,” <https://www.medicalnewstoday.com/articles/159442>, accessed: 2021-06-07.
- [2] “2020 alzheimer’s disease facts and figures,” *Alzheimer’s & Dementia*, vol. 16, no. 3, pp. 391–460, 2020. [Online]. Available: <https://alz-journals.onlinelibrary.wiley.com/doi/abs/10.1002/alz.12068>
- [3] BruceBlaus, “Pet scan-normal brain-alzheimers disease brain,” 2018. [Online]. Available: https://commons.wikimedia.org/wiki/File:Alzheimers_Disease.jpg
- [4] Health and H. S. Department, “Pet scan-normal brain-alzheimers disease brain,” 2013. [Online]. Available: https://commons.wikimedia.org/wiki/File:PET_scan-normal_brain-alzheimers_disease_brain.PNG
- [5] J. Tian, G. Smith, H. Guo, B. Liu, Z. Pan, Z. Wang, S. Xiong, and R. Fang, “Modular machine learning for alzheimer’s disease classification from retinal vasculature,” *Scientific Reports*, vol. 11, p. 238, 01 2021.
- [6] I. R. R. da Silva, G. dos Santos Lucas e Silva, R. G. de Souza, M. A. de Santana, W. W. A. da Silva, M. E. de Lima, R. E. de Souza, R. Fagundes, and W. P. dos Santos, “Chapter four - deep learning for early diagnosis of alzheimer’s disease: a contribution and a brief review,” in *Deep Learning for Data Analytics*, H. Das, C. Pradhan, and N. Dey, Eds. Academic Press, 2020, pp. 63–78. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/B9780128197646000053>
- [7] “Diagnosing alzheimer’s: How alzheimer’s is diagnosed,” <https://www.mayoclinic.org/diseases-conditions/alzheimers-disease/in-depth/alzheimers/art-20048075>, accessed: 2021-08-04.
- [8] “Dementia statistics,” <https://www.alzint.org/about/dementia-facts-figures/dementia-statistics/>, accessed: 2021-06-08.
- [9] G. Castellazzi, M. G. Cuzzoni, M. Cotta Ramusino, D. Martinelli, F. Denaro, A. Ricciardi, P. Vitali, N. Anzalone, S. Bernini, F. Palesi, E. Sinforiani,

- A. Costa, G. Micieli, E. D'Angelo, G. Magenes, and C. A. M. Gandini Wheeler-Kingshott, "A machine learning approach for the differential diagnosis of alzheimer and vascular dementia fed by mri selected features," *Frontiers in Neuroinformatics*, vol. 14, p. 25, 2020. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fninf.2020.00025>
- [10] "Detecting alzheimer's earlier with the help of machine-learning algorithm," <https://www.genengnews.com/news/detecting-alzheimers-earlier-with-the-help-of-machine-learning-algorithm/>, 2020, accessed: 2021-09-14.
- [11] J. Giorgio, S. M. Landau, W. J. Jagust, P. Tino, and Z. Kourtzi, "Modelling prognostic trajectories of cognitive decline due to alzheimer's disease," *NeuroImage: Clinical*, vol. 26, p. 102199, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S221315822030036X>
- [12] "Machine learning for comprehensive forecasting of alzheimer's disease progression," *Scientific Reports*, vol. 9, no. 1, p. 13622, 2019. [Online]. Available: <https://doi.org/10.1038/s41598-019-49656-2>
- [13] "What causes alzheimer's disease?" <https://www.nia.nih.gov/health/what-causes-alzheimers-disease>, accessed: 2021-06-07.
- [14] S. Raschka and V. Mirlalili, *Python machine learning, Third Edition*, 3rd ed. Birmingham: Packt Publishing, 2019.
- [15] "Difference between classification and regression in machine learning," <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/>, accessed: 2021-11-10.
- [16] V. Agarwal, "Research on data preprocessing and categorization technique for smartphone review analysis," *International Journal of Computer Applications*, vol. 131, no. 4, pp. 30–31, 2015.
- [17] J. Brownlee, "How to use power transforms for machine learning," <https://machinelearningmastery.com/power-transforms-with-scikit-learn/>, accessed: 2021-01-28.
- [18] S. Weisberg, "Yeo-johnson power transformations," <https://www.stat.umn.edu/arc/yjpower.pdf>, accessed: 2021-01-28.
- [19] A. Fawcett, "Data science in 5 minutes: What is one hot encoding?" 2021. [Online]. Available: <https://www.educative.io/blog/one-hot-encoding>
- [20] A. Spriestersbach, B. Röhrig, J. Prel, A. Gerhold-Ay, and M. Blettner, "Descriptive statistics the specification of statistical measures and their presentation in tables and graphs part 7 of a series on evaluation of scientific publications," *Deutsches Ärzteblatt international*, vol. 106, pp. 578–83, 2009.

- [21] A. Garcia Asuero, A. Sayago, and G. González, “The correlation coefficient: An overview,” *Critical Reviews in Analytical Chemistry - CRIT REV ANAL CHEM*, vol. 36, pp. 41–59, 01 2006.
- [22] A. Smilde, H. Kiers, S. Bijlsma, C. Rubingh, and M. Erk, “Matrix correlations for high-dimensional data: The modified rv-coefficient,” *Bioinformatics (Oxford, England)*, vol. 25, pp. 401–5, 02 2009.
- [23] V. Gupta, G. Singh, R. Singh, R. Singh, and H. Singh, “An introduction to principal component analysis and its importance in biomedical signal processing,” vol. 11, 2011.
- [24] D. Lay, S. Lay, and J. McDonald, *Linear Algebra and Its Applications, Global Edition*, 5th ed. London: Pearson, 2015.
- [25] A. Sykes, “An introduction to regression analysis,” *Coase-Sandor Working Paper Series in Law and Economics*, 10 1993.
- [26] T. Haslwanter, “Residuals for linear regression fit,” 2013. [Online]. Available: https://commons.wikimedia.org/wiki/File:Residuals_for_Linear_Regression_Fit.png
- [27] D. V. Guebel and N. V. Torres, *Partial Least-Squares Regression (PLSR)*. New York, NY: Springer New York, 2013, pp. 1646–1648. [Online]. Available: https://doi.org/10.1007/978-1-4419-9863-7_1274
- [28] Mitchell, “Rosenblattperceptron,” 2012. [Online]. Available: <https://commons.wikimedia.org/wiki/File:Rosenblattperceptron.png>
- [29] Chrislb, “Sigmoidfunction,” 2006. [Online]. Available: <https://commons.wikimedia.org/wiki/File:SigmoidFunction.svg>
- [30] C. Zhang, X. Shao, and D. Li, “Knowledge-based support vector classification based on c-svc,” *Procedia Computer Science*, vol. 17, pp. 1083–1090, 12 2013.
- [31] Larhmam, “Svm margin,” 2008. [Online]. Available: https://commons.wikimedia.org/wiki/File:SVM_margin.png
- [32] H. Patel and P. Prajapati, “Study and analysis of decision tree based classification algorithms,” *International Journal of Computer Sciences and Engineering*, vol. 6, pp. 74–78, 10 2018.
- [33] Pkuwangyan06, “Decision trees,” 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Decision_Trees.png
- [34] V. Jagannath, “Random forest diagram complete,” 2017. [Online]. Available: https://commons.wikimedia.org/wiki/File:Random_forest_diagram_complete.png

- [35] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, “Random forests and decision trees,” *International Journal of Computer Science Issues(IJCSI)*, vol. 9, 09 2012.
- [36] S. Dreiseitl and L. Ohno-Machado, “Logistic regression and artificial neural network classification models: a methodology review,” *Journal of Biomedical Informatics*, vol. 35, no. 5, pp. 352–359, 2002. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1532046403000340>
- [37] A. Ajanki, “Knnclassification,” 2007. [Online]. Available: <https://commons.wikimedia.org/wiki/File:KnnClassification.svg>
- [38] S. Gupta and P. Meel, “Fake news detection using passive-aggressive classifier,” in *Inventive Communication and Computational Technologies*, G. Ranganathan, J. Chen, and Á. Rocha, Eds. Singapore: Springer Singapore, 2021, pp. 155–164.
- [39] “Underfitting,” 2021. [Online]. Available: <https://www.ibm.com/cloud/learn/underfitting>
- [40] T. Isaksson, “Statistical bias and statistical noise illustration,” 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Statistical_bias_and_statistical_noise_illustration.png
- [41] Leomaurodesenv, “Underfitting e overfitting,” 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:Underfitting_e_overfitting.png
- [42] “Complete guide to regularization techniques in machine learning,” 2021. [Online]. Available: <https://www.analyticsvidhya.com/blog/2021/05/complete-guide-to-regularization-techniques-in-machine-learning/>
- [43] C. Xiaoli, “Regularization,” 2018. [Online]. Available: <https://commons.wikimedia.org/wiki/File:Regularization.jpg>
- [44] C. De Mol, E. De Vito, and L. Rosasco, “Elastic-net regularization in learning theory,” *Journal of Complexity*, vol. 9, pp. 201–230, 2009.
- [45] F. Flöck, “K-fold cross validation en,” 2016. [Online]. Available: https://commons.wikimedia.org/wiki/File:K-fold_cross_validation_EN.jpg
- [46] D. Chicco and G. Jurman, “The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, no. 1, p. 6, 2020. [Online]. Available: <https://doi.org/10.1186/s12864-019-6413-7>
- [47] J. Brownlee, “How to choose a feature selection method for machine learning,” 2020. [Online]. Available: <https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/>

- [48] A. Jenul, S. Schrunner, K. H. Liland, U. G. Indahl, C. M. Futsæther, and O. Tomic, “Rent—repeated elastic net technique for feature selection,” *IEEE Access*, vol. 9, pp. 152 333–152 346, 2021.
- [49] M. I. Uddin, I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, “Fake news detection using machine learning ensemble methods,” *Complexity*, vol. 2020, p. 8885861, 2020. [Online]. Available: <https://doi.org/10.1155/2020/8885861>
- [50] J. Brownlee, “A gentle introduction to ensemble learning algorithms,” 2021. [Online]. Available: <https://machinelearningmastery.com/tour-of-ensemble-learning-algorithms/>
- [51] “Anaconda software distribution,” 2020. [Online]. Available: <https://docs.anaconda.com/>
- [52] Roche, “pyreadstat,” 2021. [Online]. Available: <https://github.com/Roche/pyreadstat>
- [53] J. Reback, jbrockmendel, W. McKinney, J. V. den Bossche, T. Augspurger, P. Cloud, S. Hawkins, gyoung, M. Roeschke, Sinhrks, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, P. Hoefler, S. Naveh, M. Garcia, J. Schendel, A. Hayden, D. Saxton, J. Darbyshire, R. Shadrach, M. E. Gorelli, F. Li, M. Zeitlin, V. Jancauskas, A. McMaster, P. Battiston, and S. Seabold, “pandas-dev/pandas: Pandas 1.3.4,” Oct. 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5574486>
- [54] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with NumPy,” *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: <https://doi.org/10.1038/s41586-020-2649-2>
- [55] KaveIO, “Phi_k correlation analyzer library,” 2020. [Online]. Available: <https://phik.readthedocs.io/en/latest/>
- [56] O. Tomic, T. Graff, K. Liland, and T. Næs, “hoggorm: a python library for explorative multivariate statistics,” 2021. [Online]. Available: <https://joss.theoj.org/papers/10.21105/joss.00980>
- [57] O. Tomic, “hoggormplot,” 2021. [Online]. Available: <https://github.com/olivertomic/hoggormPlot>
- [58] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [59] A. Bilogur, “Missingno: a missing data visualization suite,” *Journal of*

Open Source Software, vol. 3, no. 22, p. 547, 2018. [Online]. Available: <https://doi.org/10.21105/joss.00547>

- [60] M. L. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: <https://doi.org/10.21105/joss.03021>
- [61] koaning, “scikit-lego,” 2021. [Online]. Available: <https://github.com/koaning/scikit-lego>
- [62] A. Jenul, S. Schrunner, B. N. Huynh, and O. Tomic, “Rent: A python package for repeated elastic net feature selection,” *Journal of Open Source Software*, vol. 6, no. 63, p. 3323, 2021. [Online]. Available: <https://doi.org/10.21105/joss.03323>
- [63] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [64] R. D’Agostino, R. Vasan, M. Pencina, P. Wolf, M. Cobain, J. Massaro, and W. Kannel, “General cardiovascular risk profile for use in primary care the framingham heart study,” *American Heart Association*, pp. 743–753, 2008.

Appendix

Appendix A: Figures from exploratory PCA

This section includes the figures from the exploratory PCA. Separated into blocks, the figures include 4 plots. The plots on the top are the scores and loadings for the first two principal components. The plot in the bottom left is the explained variance in x as a function of the number of components. The correlation loadings for the first two principal components is shown in the plot in the bottom right of the figure.

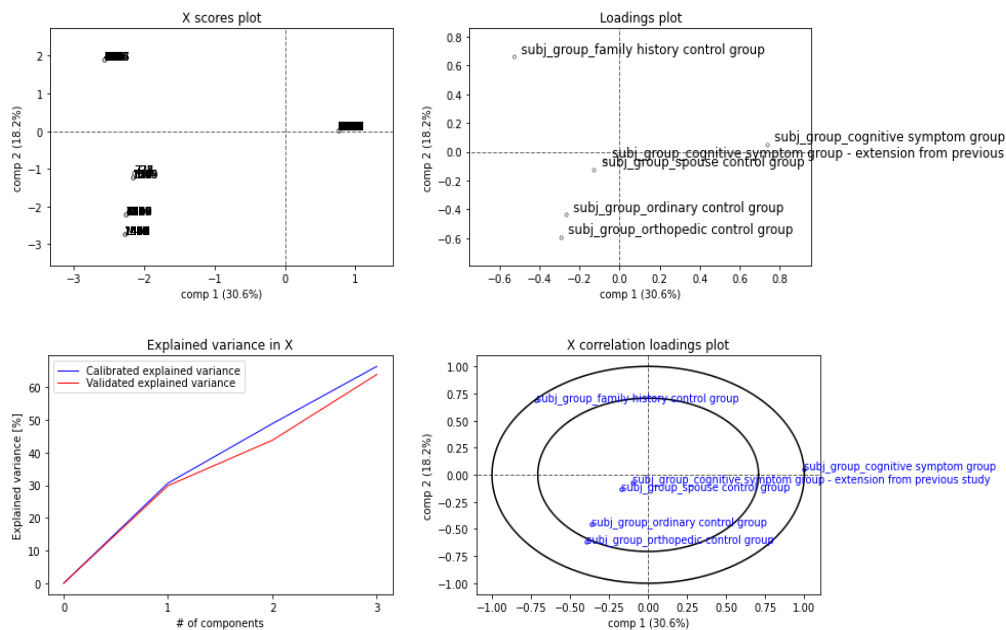


Figure 7.1: Block A: Figures from PCA.

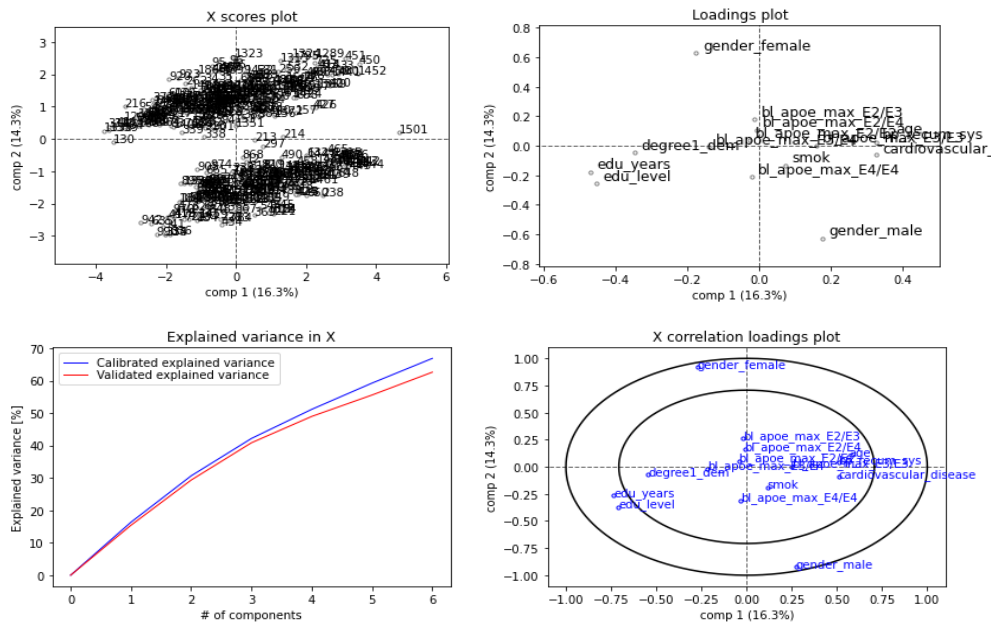


Figure 7.2: Block B: Figures from PCA.

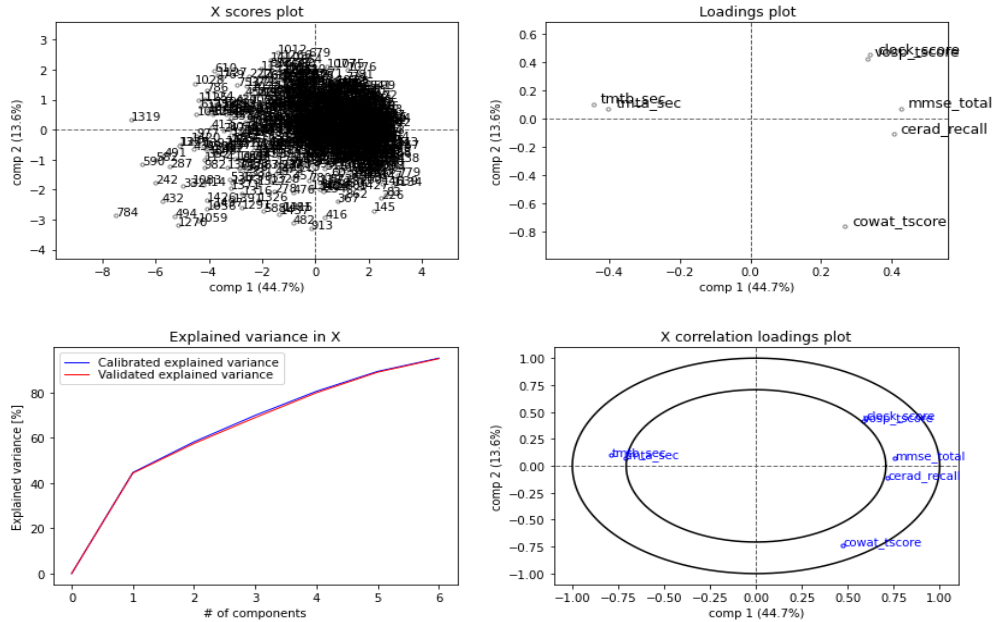


Figure 7.3: Block C: Figures from PCA.

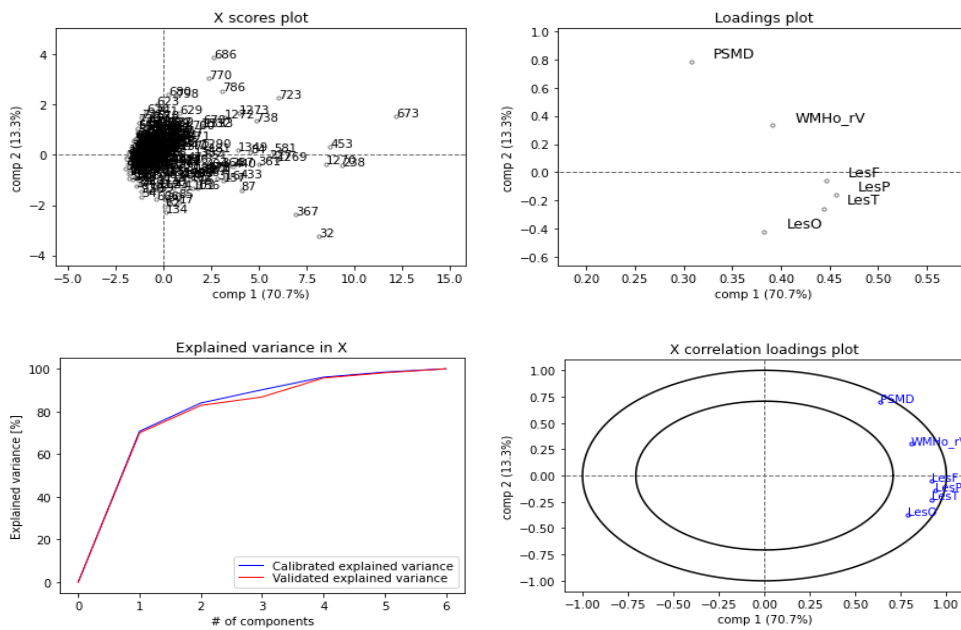


Figure 7.4: Block D: Figures from PCA.

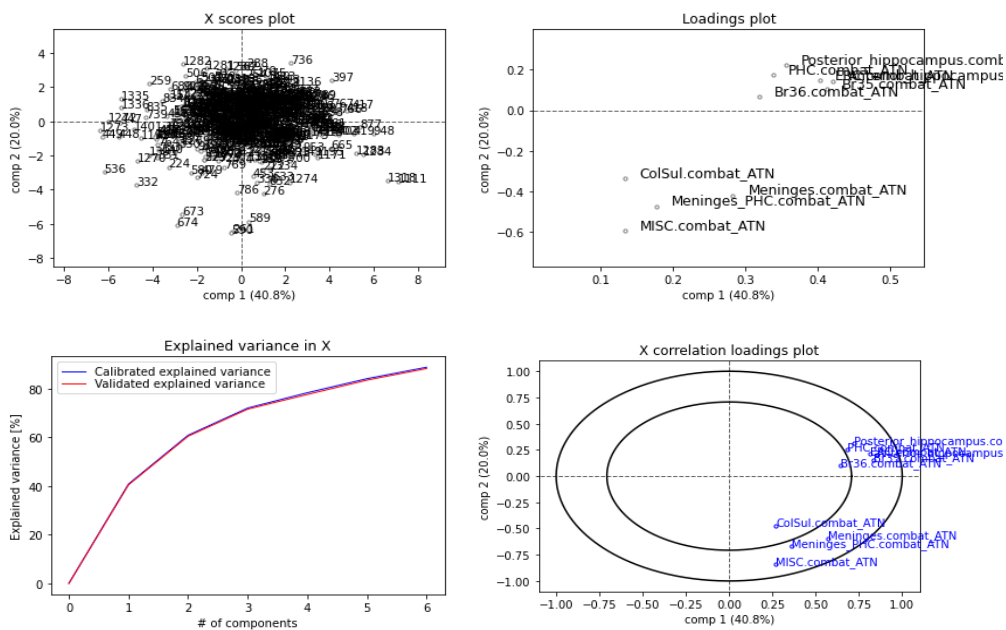


Figure 7.5: Block E: Figures from PCA.

Appendix B: Figures from exploratory analysis with PLSR

This section includes the figures from the exploratory analysis with PLSR. Separated into blocks, the figures include 4 plots. The plots on the top are the scores and correlation loadings for the first two principal components. The correlation loadings are for x and y. The plots on the bottom are the explained variance as a function of the number of components. The left plot is of the explained variance in x, and the right is the explained variance in y.

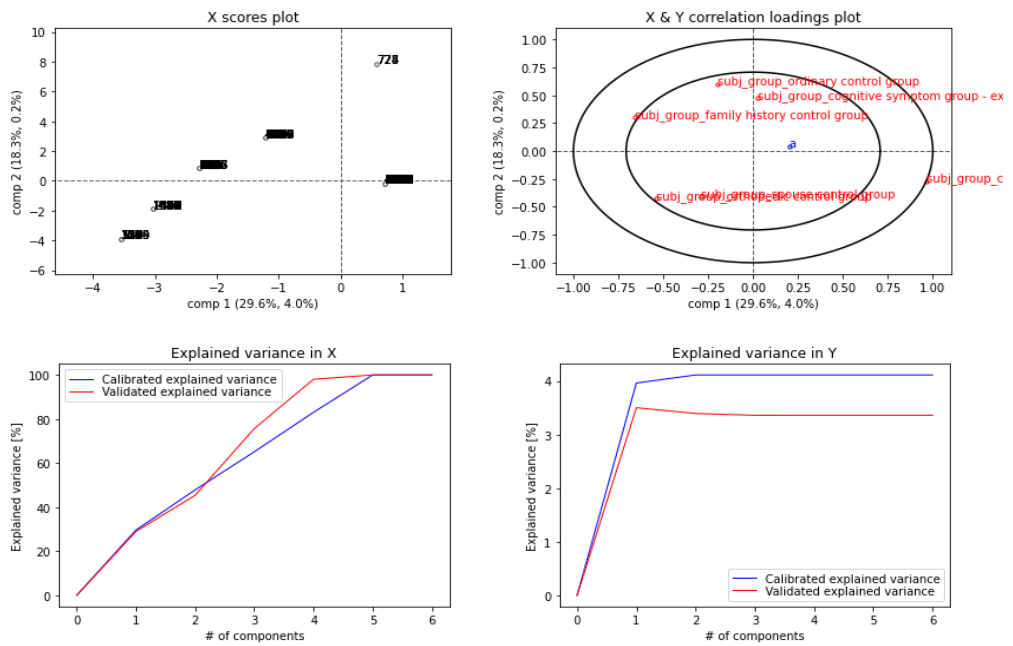


Figure 7.6: Block A: Figures from PLSR analysis.

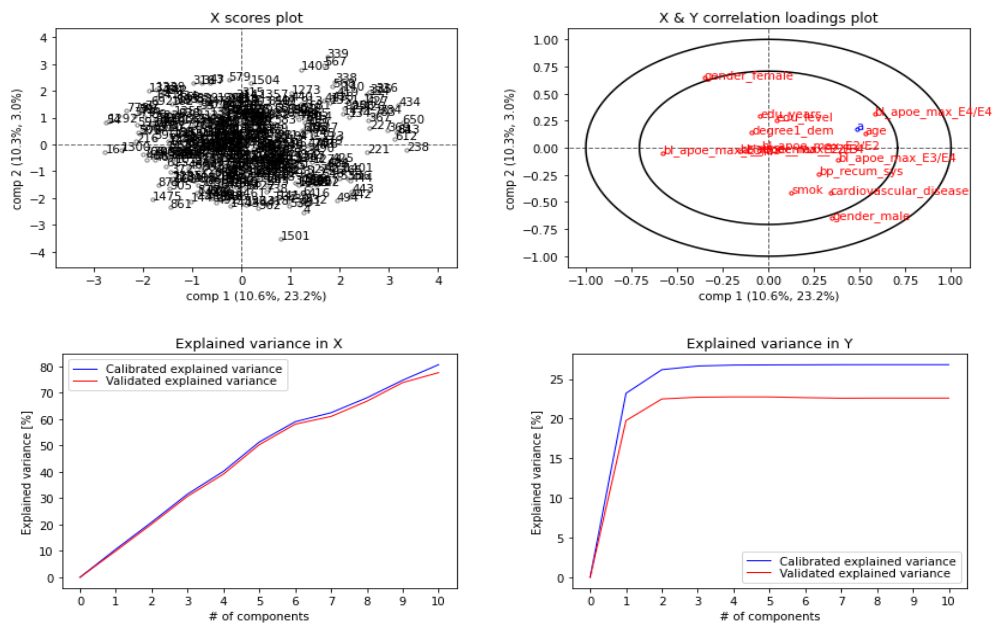


Figure 7.7: Block B: Figures from PLSR analysis.

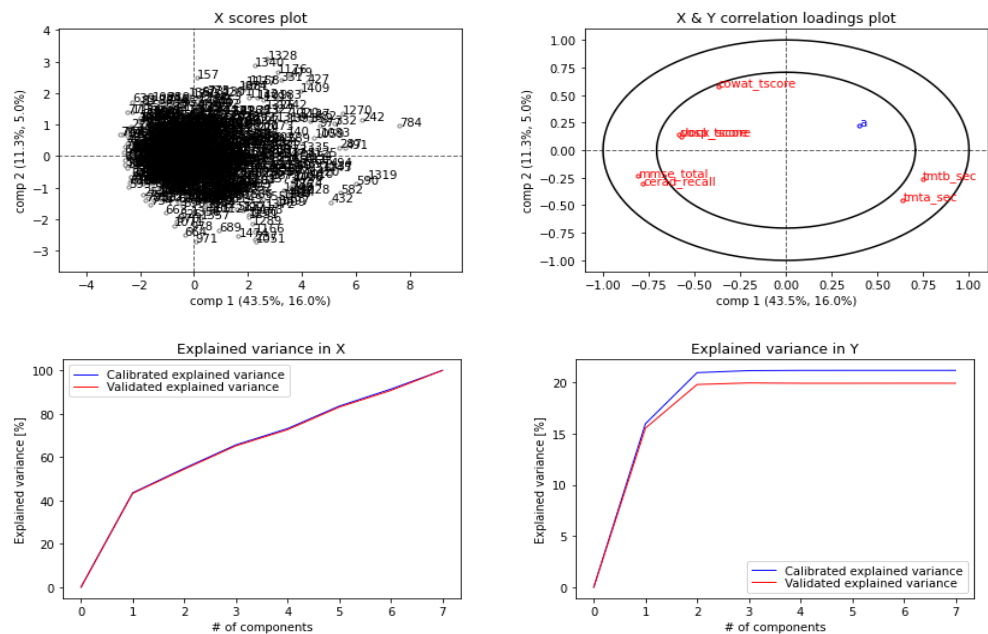


Figure 7.8: Block C: Figures from PLSR analysis.

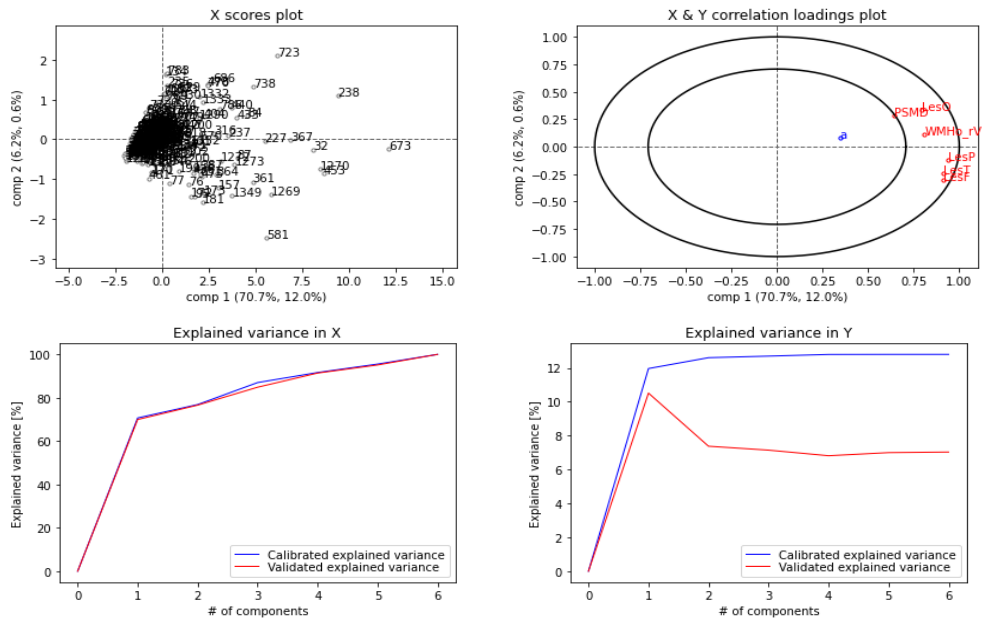


Figure 7.9: Block D: Figures from PLSR analysis.

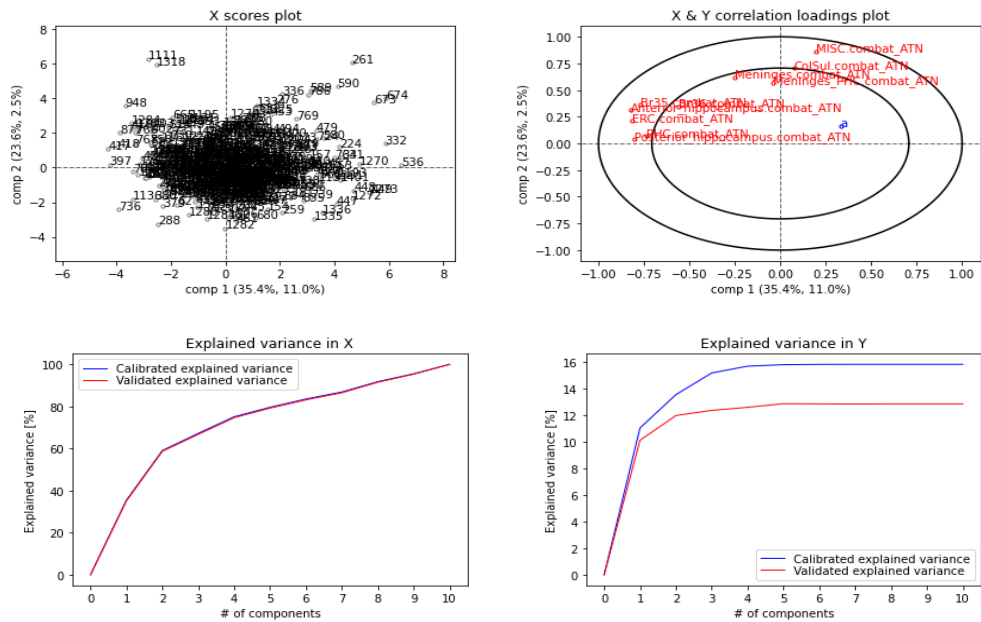


Figure 7.10: Block E: Figures from PLSR analysis.

Appendix C: Figures from Outlier detection

This section includes the figures from the decomposition based outlier detection with the scikit-lego package. For each block the figure includes two plots. The plot on the left is outlier detection via parallel coordinates. The plot on the right is outlier detection for the first two principal components.

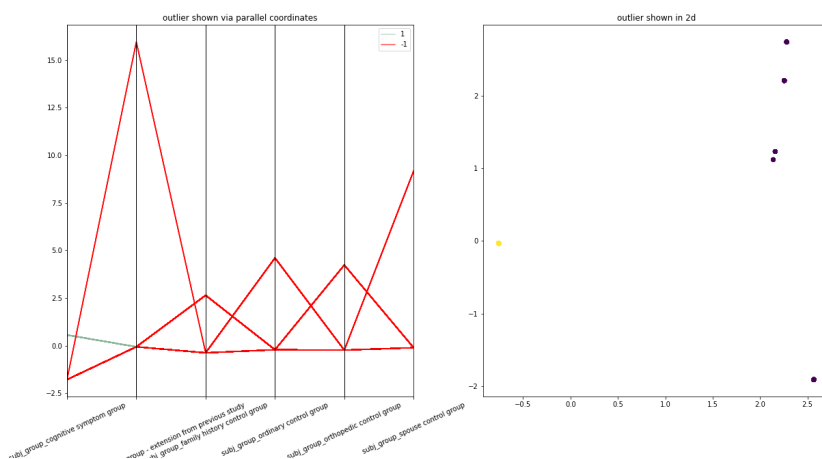


Figure 7.11: Block A: Outlier detection.

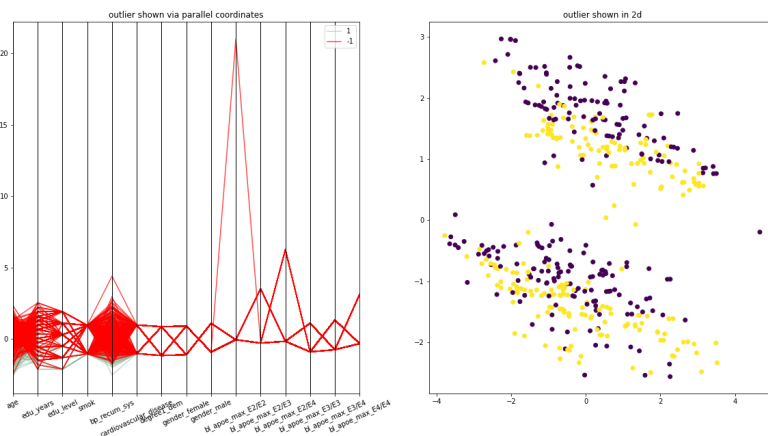


Figure 7.12: Block B: Outlier detection.

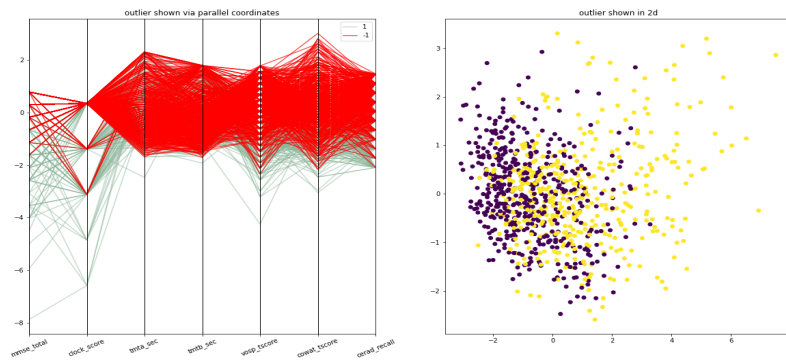


Figure 7.13: Block C: Outlier detection.

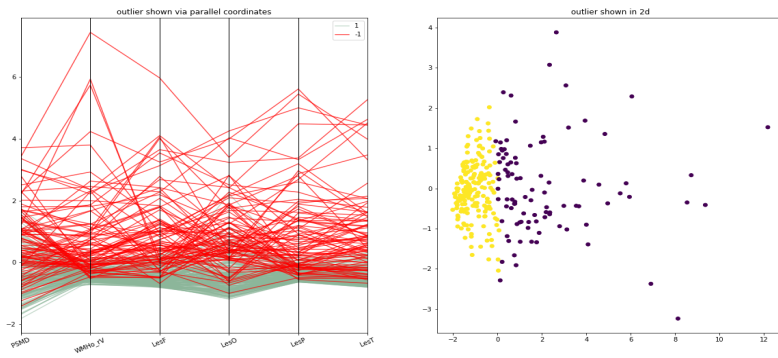


Figure 7.14: Block D: Outlier detection.

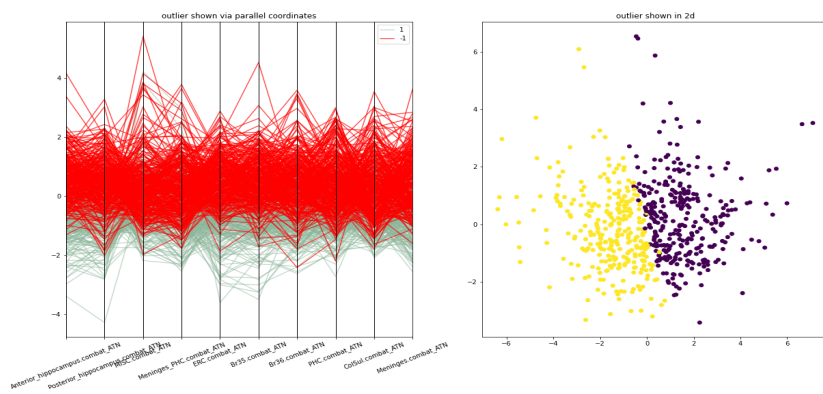


Figure 7.15: Block E: Outlier detection.

Appendix D: Heatmaps from RENT analysis

This section includes the heatmaps from the block-wise RENT analysis. Two heatmaps are provided for each block. The first is the heatmap of the scores, where the values are the MCC score. The second heatmap presents the zeroes, which is the fraction of variables set to zero. The axis are the same for all the plots, with the value for the parameter C along the horizontal axis and the value for the l1-ratio along the vertical axis.

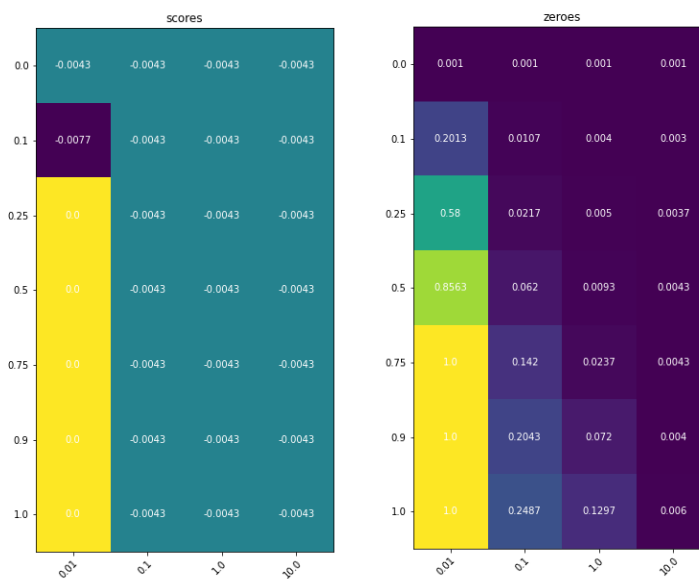


Figure 7.16: Block A: heatmaps of scores and zeroes.

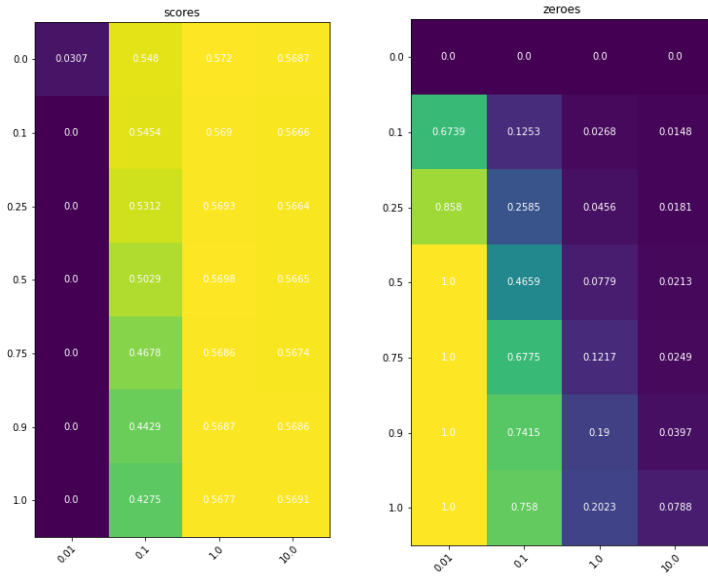


Figure 7.17: Block B: heatmaps of scores and zeroes.

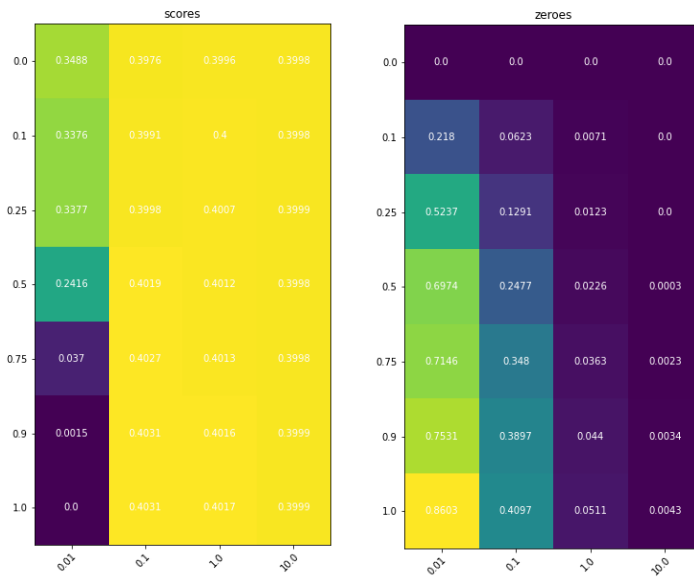


Figure 7.18: Block C: heatmaps of scores and zeroes.

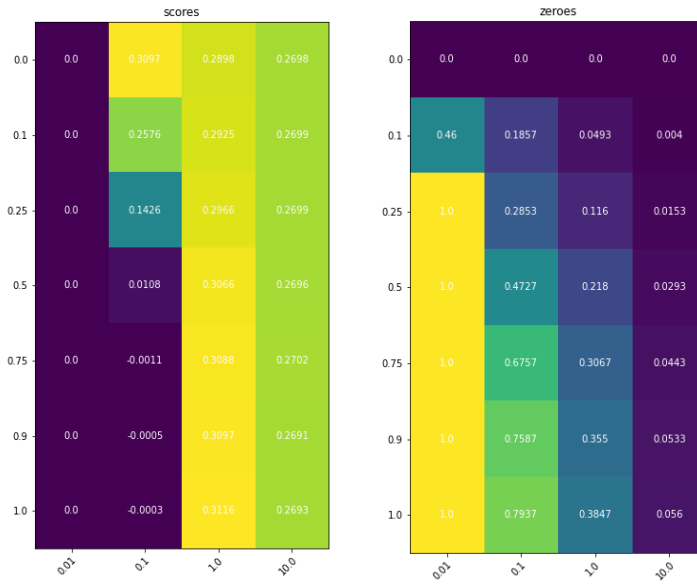


Figure 7.19: Block D: heatmaps of scores and zeroes.

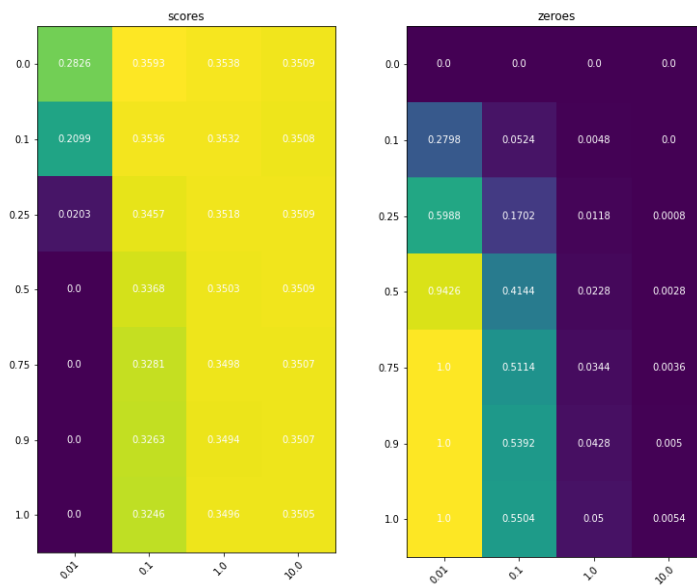


Figure 7.20: Block E: heatmaps of scores and zeroes.



Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitenskapelige universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway