University of Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study programme/specialisation: Master of Science in Data Science	Spring/ ~~Autumn~~ semester, 2020 Open / ~~Confidential~~
Author: Andrijana Podhraški, Trond Tjersland	
Programme coordinator: Supervisor(s): Ketil Oppedal, Álvaro Fernández Quílez	
Title of master's thesis: Prediction of Psychosis in Parkinson's Patients using Machine Learning	
Credits: 30 ECTS	
Keywords: Parkinson's Disease, Psychosis, Machine Learning, Deep Learning, Feature Selection, Longitudinal Study	Number of pages: 196 + supplemental material/other: Python code on GitHub Stavanger, 15.6.2020

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Prediction of Psychosis in Parkinson's Patients using Machine Learning

Master's Thesis in Computer Science
by

Andrijana Podhraški, Trond Tjersland

Supervisors

Ketil Oppedal

Álvaro Fernández Quílez

June 13, 2020

Abstract

Parkinson's disease is one of the most common neurological disorders with an estimated 6.3 million PD patients worldwide, which makes it a great threat to public health. Psychosis is a common symptom of Parkinson's disease and over half of patients with PD develop symptoms of psychosis over the course of their disease. PD patients who develop such symptoms have greater rates of cognitive decline and progression to dementia. It is important to detect patients that will eventually develop psychosis as early as possible, so that appropriate treatment can be started and negative effects can be reduced for both patient and caretakers.

Early detection of psychosis in PD patients is a difficult task from a clinical viewpoint. This project therefore aimed to develop various ML models, on a subset of data collected by the Parkinson's Progression Markers Initiative (PPMI), to predict psychosis in PD patients. We selected features from the PPMI database that are more easily collected and therefore more available. Examples include clinical tests, questionnaires and demographic data, but not for example brain imaging scans or biological samples such as cerebrospinal fluid. This makes our models applicable on a wide range of PD patients. We developed a preprocessing pipeline for the selected features and three different prediction approaches: using baseline data, derived longitudinal statistics and deep learning. The latter two approaches benefit from the temporal nature of the data from the PPMI study.

The derived statistics approach gave us the best results. It was followed by the baseline approach, while the deep learning approach came last. Support vector machines and logistic regression were our best performing models. Among tree based models, random forest showed the best performance. The deep learning approach made use of a long short-term memory (LSTM) network, but did not produce the best results.

Acknowledgements

We would like to thank our supervisors, Ketil Oppedal and Álvaro Fernández Quílez, for guidance and invaluable feedback throughout our work on this thesis.

We would also like to thank MD Maria Camila Gonzalez Velez, who has been very helpful in explaining some clinical details related to Parkinson's Disease and the motivation of the project.

Finally, Andrijana would like to thank her family, especially her son, for understanding and patience during her Master's degree. Trond would like thank his family and significant other for providing advice and encouragement during the writing of this thesis.

Contents

Abbreviations

ADASYN	Adaptive Synthetic Sampling
ANN	Artificial Neural Network
AUC	Area under the ROC Curve
BJLO	Benton Judgement of Line Orientation
DT	Decision Trees
EDS	Excessive Day Time Sleepiness
ESS	Epsworth Sleepiness Scale
FN	False negative
FP	False positive
GDS	Geriatric Depression Scale
HEOM	Heterogeneous Euclidean-Overlap Metric
HVLT	Hopkins Verbal Learning Test
ICOTE	Immune Centroids Oversampling Technique
LNS	Letter-Number Sequencing
LR	Logistic Regression
LSTM	Long Short-Term Memory
MCC	Matthews Correlation Coefficient
MD	Medical doctor
MDS-UPDRS	Movement Disorder Society - Unified Parkinson's Disease Rating Scale
ML	Machine Learning
MLP	Multilayer Perceptron
MoCA	Montreal Cognitive Assessment
MRI	Magnetic resonance imaging
MTDF	Mega-trend Diffusion Function
MWMOTE	Majority Weighted Minority Oversampling Technique

OOB	Out-of-Bag estimation error
PCA	Principal Component Analysis
PD	Parkinson's disease
PPMI	Parkinson's Progression Marker Initiative
ROC	Receiver Operating Characteristic
RBD	Rapid Eye Movement Sleep Behavior Disorder
REM	Rapid Eye Movement
RNN	Recurrent Neural Network
SCOPA-AUT	Scales for Outcomes in Parkinson's Disease - Autonomic Dysfunction
SDMT	Symbol Digit Modalities Test
SFT	Semantic Fluency Test
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
SWEDD	Subjects with Scans without Evidence of a Dopaminergic Deficit
TD/PIGD	Tremor Dominant and Postural Instability and Gait Disorder
TN	True negative
TP	True positive
TRkNN	Couples Top-N Reverse k-Nearest Neighbor
UPSIT	University of Pennsylvania Smell Identification Test
WHO	World Health Organization

List of Symbols

symbol	name
X	Data set, $X = \{x_1, x_2, ..., x_n\}$, $X \epsilon \mathbb{R}^{n*p}$
x_i	Observation i with p features $x_i = (x_{i1}, x_{i2}, ..., x_{ip}) \epsilon \mathbb{R}^p$
n	Number of observations (measurements)
p	Number of features
Θ	Set of parameters of the ML model, $\Theta = \{w_j \vert j = 1, ..., d\}$
$Obj(\Theta)$	Objective function
$L(\Theta)$	Training loss
$\Omega(\Theta)$	Regularization
HEOM(x,y)	Heterogeneous Euclidean-Overlap distance between observations x and y
μ_j	Mean of the jth feature
σ_j	Sample standard deviation of the jth feature
$\mathbf{X} \epsilon \mathbb{R}^{n*p}$	Matrix that represents data set $X = \{x_1, x_2, ..., x_n\}$, $X \epsilon \mathbb{R}^{n*p}$, formed of observations placed in rows.
$\mathbf{1}$	Vector of ones, $1 \epsilon \mathbb{R}^n$
$\boldsymbol{\mu}$	Empirical mean vector, $\boldsymbol{\mu} \epsilon \mathbb{R}^p$
$\mathbf{C}$	Covariance matrix, $\mathbf{C} \epsilon \mathbb{R}^{p*p}$
$\mathbf{v}_i$	Eigenvector
λ_i	Eigenvalue corresponding to the eigenvector $\mathbf{v}_i$
$\mathbf{D}$	Diagonal matrix of eigenvalues, $\mathbf{D} \epsilon \mathbb{R}^{p*p}$
$p(x)$	Probability of x
$L(\beta)$	Likelihood function
$\langle x, y \rangle$	Inner product between x and y
$K(x, y)$	Kernel function

R_i	Sub-region of the feature space
G	Gini index for a sub-region
D	Entropy for a sub-region
$\hat{y}_i$	Prediction for ith observation
f_k	Function that represents a tree
F	Space of all tree functions
z_i	Output value from neural network
y_i	Target value (neural network)
$J(w)$	Total training error (neural network)
w	Vector of weights in neural network
η	Learning rate
U, V, W	Matrices of weights
b_h, b_o	Bias terms
$f_h(\cdot), f_o(\cdot)$	Activation functions
h_t	Hidden unit from step t
$\sigma(\cdot)$	Element-wise sigmoid function
$\odot$	Element-wise product
T	T-score

Chapter 1

Introduction

This chapter serves as an introduction to the project. We present the motivation for the work done, state our purpose, main goals and requirements. We further state the contributions we have made by the completion of the project. Finally, we give an outline for the structure of this thesis.

The code that was developed for this thesis is available on GitHub at: https://github.com/Tjersland/Master_2020_Prediction_of_Psychosis_in_Parkinsons_Patients_using_Machine_learning

1.1 Motivation

Parkinson's disease (PD) is a neurodegenerative disorder which is generally characterized by a slow, but irreversible decline in motor functions. Typical symptoms include tremors, slow rigid movements and postural instability. While the disease is most often recognized by its motor symptoms, many PD patients experience symptoms unrelated to movement such as depression and cognitive impairment, up to and including dementia. These symptoms are collectively known as non-motor symptoms and they can often be more impactful on a PD patient than the motor symptoms. One of the most common non-motor symptoms of PD is psychosis. As a matter of fact, more than half of patients with PD develop symptoms of psychosis over the course of their disease [1].

Psychosis in PD patients is usually characterized by hallucinations and delusions. Hallucinations involve seeing, hearing, smelling, tasting or feeling something which does not exist in reality. Delusions are a set of wrongly-held beliefs not based in reality, but considered the truth by the delusional individual. According to a research article, Zahodne et al (2010) [2], Parkinson's psychosis is a *"leading reason for nursing home*

placement of patients with Parkinson's disease (PD). It may also be the single greatest stressor for caregivers of PD patients, it is generally persistent, and its presence markedly increases the risk of mortality.". Another research article, Barrett et al (2018) says [3] *"Psychosis in Parkinson disease (PD) is a clinical marker of advanced disease and is associated with dementia, increased institutionalization and increased mortality"*. It is therefore important to recognize and treat the symptoms of psychosis as early as possible in a PD patient.

The early detection of psychosis in PD patients with the help of machine learning (ML) algorithms is a novel idea. Early detection of psychosis in PD patients based on clinical diagnosis can be difficult and often inaccurate [4]. If ML models could be developed to predict, with a high enough certainty, whether or not a PD patient will develop psychosis symptoms before they make themselves apparent, appropriate treatment and prevention could begin a lot earlier in the development of the psychosis and therefore reduce the negative effects of such symptoms for both patient and caretakers.

1.2 Project Purpose and Requirements

In this project we will select relevant features and develop models using data from the Parkinson's Progression Markers Initiative (PPMI) database. We will develop a preprocessing pipeline for the selected data and apply this processed data to train various ML algorithms. Our purpose is to develop ML models that can predict, with a high enough certainty, whether or not a PD patient will develop psychosis symptoms based on currently available data where the patient has not yet shown signs of psychosis.

The main project goals can be summed up as follows:

1. To select and obtain features from the PPMI database that can aid in prediction of psychosis using ML models.
2. To develop appropriate preprocessing pipelines for the selected features.
3. To tune selected ML models in order to find the best performing models.
4. To use our tuned models for inference if possible.

The requirements for this project is as follows:

1. To perform manual feature selection on the PPMI database based on two research articles, Barret et al (2018) [3] and Ffytche et al (2017) [5]

2. To develop prediction models trained in two scenarios: on baseline data with no benefit of temporal data and on longitudinal data.

3. To at least include these classical ML models: (1) support vector machines because they often give good results for binary classification problems, (2) random forest and boosted trees that, as ensemble methods, are less susceptible to overfitting than decision trees, but still provide some inference.

4. To at least train one deep learning model on the longitudinal data and compare the results to classical ML models trained on derived longitudinal statistics.

1.3 Problem Definition

We want to predict development of psychosis in PD patients, based on data collected in early stages of the disease. Baseline assessment is done when a subject joins the study and serves as criteria for assigning the subject to one of the study's cohorts. The subjects in the cohorts we use in this project are diagnosed with PD at most two years before the baseline assessment. They are followed for at least five years and a new full assessment is undertaken at yearly intervals. If there are no missing observations, we should have at least six observations for each of our subjects, one at baseline and a further five for each yearly assessment. We want to use a subset of the data gathered at each assessment for our predictions. This subset consists of data that is relatively easy to gather and therefore often available, e.g questionnaires or clinical assessments. Our goal is to develop a model that can be applied to a wide range of PD patients. The first and most crude prediction models will be based only on baseline data. We will then try to refine these models by incorporating features from the yearly assessments. We will do so by defining some statistics on these yearly assessment so that we can incorporate some of the information they contain into our baseline models. Finally, we want to fully incorporate all the information contained in the yearly assessments. To do so we must move away from the classical machine learning models used in our baseline and refined approaches and instead employ deep learning. We would make use of an LSTM network for this task. Models will be measured in classification performance using multiple metrics.

1.4 Contributions

The contributions we make in this thesis can be summarised as:

1. We develop a preprocessing pipeline that selects and processes features from the PPMI database so that they can be used for ML models.
2. We implement three different prediction approaches for psychosis in PD patients: (1) using baseline data to train classical ML models, (2) using temporal data to derive longitudinal statistics to train classical ML models and (3) using temporal data to train a deep learning model.
3. We make use of a variety of different ML models, including deep learning and report comprehensively on their results.
4. We analyze the results from our models and do inference when applicable.

The authors of this thesis wish to state that both authors contributed equally to the thesis and no individual ownership is claimed for any part or section of it.

1.5 Thesis Structure

The thesis is divided into six chapters.

Chapter one, **Introduction** seeks to establish the motivation and main purpose of the project. It also states our main goals and requirements and gives an outline of the structure of the thesis.

Chapter two, **Parkinson's Disease, PPMI and Related Work** gives a more indepth introduction to Parkinson's disease and introduces the Parkinson's Progression Markers Initiative (PPMI). Finally, it gives an overview of the related works of Barret et al (2018) [3] and Ffytche et al (2017) [5], which this thesis is partly based on.

In chapter three, **Introduction to Methodology and Theoretical Framework** we try to succinctly present necessary theoretical background for the work performed in this project. The chapter begins with an introductory part about ML basics and data set design, after which important preprocessing steps are introduced. Further, a set of classical ML and deep learning algorithms used for this project are presented. We finish the chapter with theory related to model validation and tuning related procedures.

Chapter four, **Methods and Materials** explains the execution of the project, from data set construction to preprocessing steps, and later, model set up, tuning and validation.

In chapter five, **Results and Discusion**, we present the results from our three different prediction approaches. Optimal hyperparameters and various performance measures are shown. We finish the chapter by analyzing and discussing these results.

Finally, the sixth chapter, **Future Work and Conclusion** gives an overview of the challenges we faced during the project and discusses some possible future developments for the project. It finishes with a conclusion of our work.

Chapter 2

Parkinson's Disease, PPMI and Related Work

This chapter is meant to give an overview of Parkinson's disease, introduce the Parkinson's Progression Markers Initiative (PPMI) and to give a summary of the related works of Barret et al (2018) [3] and Ffytche et al (2017) [5].

2.1 Parkinson's Disease

2.1.1 Overview

World Health Organization (WHO) [1] recognizes neurological disorders as one of the greatest threats to public health and one of the most common neurological disorders is Parkinson's disease.

It is estimated that worldwide there are 6.3 million PD patients, with 1.2 million in Europe alone. In Norway there are around 8000 patients diagnosed with PD [6]. PD affects around 1% of the population older than 65. Parkinson's disease is a huge burden on not only the patient, but also the primary caretakers because of loss of income and the emotional issues that often affect PD patients. This is especially true if the patient develops dementia as part of the disease. Finally, treatment in less developed countries is often sub-optimal due to lack of trained medical workers and cost of medication.

Parkinson's disease is spread around the world with an incidence rate[1] per year of 4.5-19 per 100 000 population. There is wide variation in incidence rate, which reflects differences in diagnostic methodology and age distribution among world's population. A

[1]Number of new diagnosed patients per yer per 100 000 people

more realistic incidence rate provided after age adjustment is 9.7-13.8. Prevalence[2] is much higher than incidence due to the prolonged nature of the disease. Large differences in prevalence rates are observed across the world (18-328 per 100 000 persons). Age-adjusted rate is in a narrower range of 72-258.8 per 100 000 persons. It is assumed that differences in prevalence rates are related to environmental risks and genetic background, while the higher rates of incidences are caused by increased awareness and improved and earlier recognition of the disease.

There are no clear risk factors for Parkinson's disease other than age and gender. Most people with PD develop the disease at around 60 and with age the incidence rate increases. Some 5 to 10 percent of cases are from people under 50, which is often referred to as "early-onset" PD. incidence rate among men is higher than among women (in relation of almost 2:1). There is a genetic component to PD, especially for early-onset cases, but most researchers now believe that PD is caused by a combination of genetic and environmental factors [7]. Treatment for PD is mainly based around taking drugs that increase the level of - or mimic the effect of dopamine in the brain. The most important of these drugs is levodopa, which crosses the blood-brain barrier to increase the level of dopamine in the brain. Unfortunately, levadopa is known to cause a number of side-effects, including hallucinations, which are a sub-symptom of psychosis.

The main symptoms of Parkinson's disease are tremors, bradykinesia[3], stiff limbs and impaired balance and coordination [7]. These symptoms are typically referred to as motor symptoms. In addition, PD patients may have symptoms such as depression and other emotional issues, sleep behavior disorders, loss of smell and increasing cognitive impairment. These symptoms are usually referred to as non-motor symptoms.

Symptoms of Parkinson's disease usually begin slowly and are almost imperceptible at the start. They then gradually become worse as the disease progresses until they finally become debilitating for the patient. Symptoms of PD and the rate of which the disease progresses can however vary greatly from patient to patient. There are no definitive biological or imaging markers for Parkinson's disease and diagnosis is currently made based on a clinical test taking into account the patient's medical history and results from neurological tests. An improvement of symptoms after starting medication is also an important diagnostic marker for PD. It is estimated that in about 25% of patients the disease is diagnosed incorrectly [8].

Parkinson's disease is caused by the death or impairment of nerve cells in a part of the brain that broadly controls movement. These neurons usually produce the important hormone and neurotransmitter dopamine and the motor symptoms experienced by

[2]Number of registered patients in population at given time
[3]Slow movement and an inability to move the body swiftly on command

PD patients are caused by a decrease in availability of dopamine as a result of this. PD patients also lose nerve endings that produce the neurotransmitter norepinephrine. Norepinephrine is a main component in many of the autonomic functions of the body such as heart rate or blood pressure. This is thought to be reason behind some of the non-motor symptoms of PD such as fatigue and problems with blood pressure. Clumps of protein called Lewy bodies are often found in the brains of PD patients. It is believed that these Lewy bodies are the main cause of nerve cell death in Parkinson's disease.

Psychosis is a common symptom of Parkinson's disease and over half of PD patients eventually develop psychosis as part of their disease progression [1]. Psychosis in PD patients most often presents itself as hallucinations and/or delusions. Hallucinations involve seeing, hearing or sensing things that are not really present. Delusions are wrongly held beliefs that do not correspond with reality. Hallucinations and delusions can often be combined, i.e the patient experiences something which is not real and is later unable to recognize this fact. The cause of psychosis in PD patients is not known, but some believe it to be a side effect of the dopaminergic drugs [4] commonly taken as first-line treatment for Parkinson's disease.

2.2 PPMI

The Parkinson's Progression Marker Initiative (PPMI) is an observational clinical study that seeks to find biomarkers[5] for Parkinson's disease. The study describes itself as *"A landmark observational clinical study to comprehensively evaluate cohorts of significant interest using advanced imaging, biologic sampling and clinical and behavioral assessments to identify biomarkers of Parkinson's disease progression"* [9]. Currently available biomarkers for PD are either non-optimal or lack verification. The study is sponsored by the Michael J. Fox Foundation for Parkinson's Research.

In essence, PPMI is a longitudinal study of patients with Parkinson's disease, along with additional statistical cohorts[6] such as healthy control groups and patients genetically predisposed to Parkinson's disease. PMMI defines six such cohorts in its study [11]:

1. De Novo PD Subjects - *"Subjects with a diagnosis of PD for two years or less who are not taking PD medications."*
2. Control Subjects - *"Control Subjects without PD who are 30 years or older and who do not have a first degree blood relative with PD."*

[4] Drugs that are related to dopamine
[5] A measureable biological indicator of biological state in a patient
[6] a group of individuals having a statistical factor in common (such as age or class membership) [10]

3. Subjects with Scans without Evidence of a Dopaminergic Deficit (SWEDD) - *"Subjects consented as PD subjects who have DaTscans that do not show evidence of a dopaminergic deficit."*

4. Prodromal Subjects - *"Subjects without Parkinson's disease who have a diagnosis of hyposmia or REM sleep behavior disorder (RBD)."*

5. Genetic Cohort Subjects - *"Subjects with and without Parkinson's disease who have a genetic mutation in LRRK2, GBA, or SNCA."*

6. Genetic Registry Subjects - *"Subjects with and without Parkinson's disease who have a genetic mutation in LRRK2, GBA, or SNCA or a first-degree relative with a LRRK2, GBA, or SNCA mutation who are evaluated at less frequent intervals to augment and broaden the follow-up of PD subjects and family members with PD associated mutations."*

Of these cohorts, the De Novo PD and Genetic Cohort are of interest to this thesis because these cohorts consist of subjects with confirmed Parkinson's disease.

Subjects in all cohorts are tested, assessed and have biological samples, such as blood or cerebrospinal fluid, taken from them at regular intervals. The type of data collected and how often it is collected is the same within a cohort, but can differ between cohorts. The subjects in the De Novo and Genetic cohort are followed for a minimum of 5 years (unless an early withdrawal) and a maximum of 13 years. Data collected from subjects can for example be the result of cognitive/motor-function tests, questionnaires, MRI imaging or biological data like blood samples. Subjects in different cohorts have different eligibility criteria for entry into the study. This must be taken into consideration when using data from different cohorts together.

2.3 Related Work

This project is partly based on the work of Barret et al (2018) [3] and Ffytche et al (2017) [5]. Both authors analyse PPMI data. Barret tries to identify baseline risk factors for PD psychosis (which includes hallucinations) using multivariate logistic regression and found that greater autonomic symptoms and higher sleep behavior scores (RBD, EDS) are associated with higher incidences of psychosis. Ffytche tries to determine baseline predictors for future psychosis in PD patients using various statistical methods and found that prior to the development of hallucinations, PD patients shows olfactory impairment, increased depression and increased sleep behaviour scores, among other predictors. Both authors found additional risk factors based on structural images of

brain and level of biomarkers in cerebrospinal fluid, but we omit aforementioned factors due to our intention to develop prediction models on a subset of data that do not include brain scans or biomarkers due to them being relatively time consuming and expensive to collect. By looking at the feature selection done in these works we benefit from the expert knowledge that the authors of these papers possess. We base much of our feature selection in this project on cognitive and motor features that are chosen by Barret and/or Ffytche, while excluding structural images and results from cerebrospinal fluid samples. This project is a partial continuation on both authors findings and an attempt to extend existing knowledge about risk factors to prediction of psychosis in PD patients.

Chapter 3

Introduction to Methodology and Theoretical Framework

In this chapter we present necessary theoretical background for the work done in this thesis. We first introduce basic ML theory and data set design, after which we move on to preprocessing techniques such as missing value imputation and feature reduction. We then present various classical ML models such as logistic regression, support vector machines (SVM), decision trees, random forest and boosted trees. Additionally, we present concepts from deep learning, particularly focusing on recurrent neural networks (RNN) and a subtype of RNNs called long short term memory networks (LSTM). Finally, the chapter ends with theory related to model validation and tuning procedures.

3.1 Fundamentals

3.1.1 Supervised vs Unsupervised Machine Learning

Machine learning (ML) is a field at the intersection of statistics, artificial intelligence and computer science and is about extracting knowledge from data. Most machine learning tasks can be divided into two categories: supervised and unsupervised. Supervised machine learning is a setting in which an algorithm makes a decision generalizing on previously known or seen examples. The algorithm is provided with a set of inputs and corresponding outputs. In return, the algorithm predicts an output given an input previously unknown. It is said that the algorithm learns from the data in a supervised manner. In unsupervised machine learning, the algorithms are given a set of inputs, but no corresponding outputs to learn from. This makes prediction impossible. In an unsupervised setting we are limited to learning about the relationship between inputs

through techniques such as clustering. Supervised machine learning can be further divided into regression and classification. Our output variable, often called the response, can be either quantitative or qualitative in nature. A quantitative variable takes on numerical values and typical examples are age, weight, blood pressure, etc. A qualitative variable takes on a value which corresponds to one of K classes, e.g "happy" from the 3 classes "sad", "normal", "happy". If the response is quantitative we are dealing with a regression task and if it is qualitative we are dealing with a classification task. This distinction is important because some machine learning algorithms only work on regression tasks and not classification tasks, and vice versa.

In this project we are dealing with a supervised machine learning problem as we have an output on which to predict on, i.e psychosis in PD patients. Our output is further qualitative in nature as psychosis in a patient is a state of being and not a numerical measurement. Psychosis in a patient can be classified as either present or not present, i.e into one of two classes. We can therefore say that this project is about developing models for a binary classification task. PD Psychosis can also be graded according to the severity of its symptoms, from mild hallucinations to more serious delusions. This project could therefore potentially also be a multiclass classification task. The reasons why we choose to keep it as a binary classification task is explored later in 4.2.2 - *Dichotomizing our Response* and 6.2 - *Future Work*.

3.1.2 Objective Function and Regularization

The objective function is the fundamental concept in machine learning. ML is a type of optimization problem and the objective function is the function that should be optimized (minimized or maximized) by taking data and the right combination of model parameters (hyperparameters) as arguments. In general the objective function can be written as 3.1

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta), \tag{3.1}$$

where $\Theta = \{w_j | j = 1, ..., d\}$ is a set of parameters that will be learned from the data set $X = \{x_1, x_2, ..., x_n\}$, $X \epsilon \mathbb{R}^{n*p}$ where $x_i = (x_{i1}, x_{i2}, ..., x_{ip}) \epsilon \mathbb{R}^p$, $i = 1, 2, ..., n$.

$L(\Theta)$ measures the training loss, i.e. how well model fit the data, while regularization $\Omega(\Theta)$ is an additional penalty term that controls model complexity. By minimizing $L(\Theta)$ we try to make our model predict the training data well, but risk overfitting the training data. By minimizing $\Omega(\Theta)$ we keep the model simple, but we risk making the model too simple to predict anything. If we have in mind the bias-variance decomposition of the objective function, then in terms of the bias-variance trade-off, $L(\Theta)$ decreases bias and

$\Omega(\Theta)$ decreases variance [12]. By minimizing the sum of both quantities we hopefully find the right bias-variance trade-off. According to [13], regularization is *"any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error"*. Thus, regularization is used to train models that generalize better on unseen data, by preventing the algorithm from overfitting. There are many regularization strategies, here we present only two later used in tuning some of our models.

L1 regularization is based on the sum of the absolute values of the parameters and then Ω has a form:

$$\Omega(\Theta) = \lambda \sum_{j=1}^{d} |w_j| \tag{3.2}$$

L2 regularization is based on the sum of the squared values of the parameters and then Ω has a form:

$$\Omega(\Theta) = \lambda \sum_{j=1}^{d} w_j^2 \tag{3.3}$$

Through the parameter λ we control the impact of the regularization term. Higher values lead to smaller parameters, but too high values for λ can lead to underfitting.

3.1.3 Prediction vs Inference

There is a certain trade-off between a model's ability to predict accurately and model's interpretability, i.e inference or knowledge about the underlying data mechanisms that can be extracted based on the model. According to [14], model accuracy and interpretability can be thought of as a trade-off. A decision tree is an example of a relatively simple model that is quite easy to interpret, but not usually as accurate as some more complex, but less interpretable models such as random forest, SVM or neural networks.

The primary goal of this project is prediction, i.e. to use data to predict an outcome - psychosis in PD patients. Inference or interpretability is a secondary objective of this work. Some inference results are presented later in this report for decision trees, random forest and boosted trees. Those results can be used to check whether a model makes sense from a clinical perspective.

3.2 Data Source

The first step of any machine learning project is to design a data set that will be used for developing the chosen ML models. This data set might consist of data specifically collected for the project or data can be taken from pre-existing databases collected by others. In our case we are constructing a data set based on data collected by the Parkinson's Progression Markers Initiative (PPMI) study. We will only be able to use data collected on certain cohorts with confirmed PD, limiting the amount of observations we have available.

High dimensional data is a term that refers to data sets where the number of input variables, also called *features*, is large. With more features it becomes harder to analyze, visualize and organize a data set. A large amount of features might also make machine learning algorithms more prone to overfitting. This is often referred to as the curse of dimensionality. In addition, if the number of features p exceeds the number of observations n, a number of machine learning algorithms become hard or impossible to train. In order to minimize the curse of dimensionality and avoid the $p > n$ problem, we will use a subset of all features available in the PPMI database to construct our data set. The subset of features we will use will be heavily dependant on the work done by Barret [3] and Ffytche [5].

3.3 Preprocessing

When we have designed a data set on which to train our models on, the next step is to process this data set so that it can be fed into machine learning algorithms. This is called preprocessing the data. Some preprocessing steps such as handling missing values are essential, as most machine learning algorithms do not work if any data points are missing. Other steps such as data standardization or feature reduction are done in order to optimize the data so that the machine learning algorithms provide better and more robust results.

3.3.1 Missing Values

Missing value handling is an essential preprocessing step as most machine learning algorithms are not designed to automatically deal with missing values.

The simplest form of handling missing values is to remove observations or features with missing values until the data set has no missing values left. If missing values are randomly

distributed across data set or across some subset of the data set then such observations can be removed. Features that contain a lot of missing values, (in practice that can be 25%, 30% or 50%, depending on the problem and the feature), could be dropped entirely. This leaves us with complete data, but has the downside of removing observations and/or features that could have made our machine learning models better and can lead to bias in the estimation of the model parameters. When removing observations one should take into consideration the size of the data set and the proportion of the missing values in the data set. According to some sources, if 5% of the data contains missing values, then removing could be a save option [15], while others state that the proportion of the missing values should not automatically lead to removal [16].

Missing value imputation is another way of handling missing values. Single imputation methods replace a missing value by a value defined by a certain rule. There are many of such methods, such as mean/mode/median imputation, interpolation and extrapolation (works in longitudinal data), imputation based on similarity, etc. The common problem in single imputation is to replace an unknown missing value by a single value and then treat it as if it were a true value, according to [17]. As a result, the single imputation ignores uncertainty and almost always underestimates the variance.

Multiple imputation overcomes this problem, by replacing the missing value with several simulated values [17]. That leads to several completed data sets that are further put through a processing pipeline. The results obtained from each completed data set are combined into a single multiple-imputation.

Because of heterogeneous types of variables in the data set we replace missing values in two ways: using mean/mode or using similarity based on Heterogeneous Euclidean-Overlap Metric (HEOM) as is described in [18] and finally choose the imputation method that produces the best results in terms of considered performance measures.

Mean/mode imputation is a simple method that replaces missing values of a categorical variable with its mode, while replaces missing values of ordinal and numerical variables with their mean. We use this method for its simplicity and speed.

In addition we wish to use possible similarities between PD patients in missing values imputation. We implement HEOM which can handle heterogeneous types of variables and is combination of overlap measure for categorical variables and normalized euclidean measure for non-categorical variables. HEOM is defined as 3.4

$$HEOM(x, y) = \sqrt{\sum_{i=1}^{n} d(x_i, y_i)^2} \tag{3.4}$$

where n is the number of features, including outcome variable, and $d(x_i, y_i)$ is the difference between ith feature values of x and y. If A_i is the ith feature then $d(x_i, y_i)$ is defined with 3.5

$$d(x_i, y_i) = \begin{cases} 1, & \text{if } x_i \text{ or } y_i \text{ are missing values} \\ \text{overlap}(x_i, y_i), & \text{if } A_i \text{ categorical: 0, if } x_i = y_i \text{; 1 otherwise} \\ \dfrac{|x_i - y_i|}{range_{A_i}}, & \text{if } A_i \text{ numerical with } range_{A_i} = max_{A_i} - min_{A_i} \end{cases} \tag{3.5}$$

For each observation with missing values, the most similar observation in term of HEOM is found and the missing values are replaced with the corresponding values from this observation.

3.3.2 Standardization and Categorical Encdoing

Standardization

Standardizing a data set is an optional preprocessing step, but some models, that depend on distance measures among the features, should be trained on standardized data. Standardizing means to rescale data to have a mean of zero and a standard deviation of one. Precisely, for given data set $X = \{x_1, x_2, ..., x_n\}$, $X \epsilon \mathbb{R}^{n*p}$ containing n training observations $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$, each with p features we standardize data with 3.6

$$z_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \tag{3.6}$$

where mean μ_j and sample standard deviation σ_j of *jth* feature are defined with 3.7

$$\begin{aligned} \mu_j &= \frac{1}{n} \sum_{i=1}^{n} x_{ij} \\ \sigma_j &= \sqrt{\frac{\sum_{i=1}^{n} (x_{ij} - \mu_j)^2}{n-1}} \end{aligned} \tag{3.7}$$

Standardization is important when features are measured on different scales or features differ a lot in magnitude. A feature with a larger range will typically outweigh a feature with a smaller range if the features are not standardized. Some of machine learning algorithms are more robust against differences in feature scales then others.

Among algorithms used in this project, tree based algorithms are less sensitive to non-standardized data, while standardization is an important preprocessing step preceding SVM or neural networks. SVM maximizes the distance between separation boundary and support vectors, according to [12]. All features should be of the same scale so that one feature does not dominate over the others when calculating distance. Similarly, neural network will during training adjust weights for larger scaled features much more than for others, i.e. neural network will prefer larger features over the smaller, according to Richard O. Duda and Stork [19], which will slow convergence to global minima. Although standardization is not a prerequisite for tree based algorithms or logistic regression, we applied it as a preprocessing step in all cases for this project as it reduces the complexity of the processing pipeline.

Categorical Encoding

Many machine learning algorithms cannot handle categorical variables so they should be converted into numerical values. Categorical variables that are ordinal, i.e. there is an intrinsic order between categories (values of the variable), are converted into numerical by assigning an integer value to each category, considering and preserving their intrinsic order by a similar ordering of integers. Categorical variables which are nominal, i.e. there is not an intrinsic order between categories, are converted into numerical by making use of binary variables. For each category one binary variable is added, such that the single original categorical variable is replaced with set of binary variables. Replacing a categorical variable with a set of equivalent binary variables is often called one-hot encoding. Due to the fact that some of variables are nominal or ordinal in this project, we use categorical encoding as a preprocessing steps for all ML models. We also use it before tree based algorithms. Although this is not strictly necessary, it reduces the complexity of our processing pipeline.

3.3.3 Dimensionality Reduction

Feature Construction and Dichotomization

Feature construction is the process of building a new set of features from an original feature set. The goal of feature construction is to reduce the amount of data, while at the same time improving the quality of the data and consequently the performance of the machine learning algorithms, according to [20]. Various approaches to feature construction can be taken. One of them is the knowledge based approach that applies existing knowledge about data and domain knowledge. We use domain knowledge

documented in PPMI project to calculate new features (total scores) from numerous single original features. When the new features are calculated, we remove the original features. In this way we reduce the number of features in our data set.

Dichotomization of a continuous (or quantitative) variable is the procedure of finding a threshold that will divide all observed values of the variable into two groups and replacing continuous values in each group with one of the two distinct values. Dichotomization of continuous variables is frequently used in medical applications. We use domain knowledge from the same PPMI source, available in Appendix B, to dichotomize several variables.

Principal Component Analysis

Principal component analysis (PCA) is an unsupervised approach to dimensionality reduction that is often used in exploratory data analysis and to make predictive models. PCA is a feature extraction approach which aims to transform data from p-dimensional space into a lower d-dimensional space. Transformed data are orthogonal projections of the original higher dimensional data, such that the variance of the projected data is maximized. Data transformation from higher dimensional space into lower dimensional space is done through several steps [21], [22].

A data set $X = \{x_1, x_2, ..., x_n\}$, containing n training observations $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$, each with p features can be represented by matrix $\mathbf{X} \epsilon \mathbb{R}^{n*p}$ formed of observations placed in rows. Firstly, the data is centered around the mean by subtracting the empirical mean vector from each row of the data matrix $\mathbf{X}$

$$\mathbf{B} = \mathbf{X} - \mathbf{1}\boldsymbol{\mu}^{\mathrm{T}}, \tag{3.8}$$

where $\mathbf{1} \epsilon \mathbb{R}^n$ is a vector of ones and $\boldsymbol{\mu} \epsilon \mathbb{R}^p$ is an empirical mean vector with elements

$$\mu_j = \frac{1}{n} \sum_{i=1}^{n} X_{ij}, j = 1, 2, ..., p. \tag{3.9}$$

Then the covariance matrix $\mathbf{C} \epsilon \mathbb{R}^{p*p}$ from the centered data matrix $\mathbf{B} \epsilon \mathbb{R}^{n*p}$, eigenvectors $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p$ and the corresponding eigenvalues $\lambda_1, \lambda_2, ..., \lambda_p$ are found

$$\begin{aligned} \mathbf{C} &= \frac{1}{n-1} \mathbf{B}^{\mathrm{T}} \mathbf{V} \\ \mathbf{V}^{-1} \mathbf{C} \mathbf{V} &= \mathbf{D} \end{aligned} \tag{3.10}$$

where $\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_p$ are columns of the matrix $\mathbf{V}\epsilon\mathbb{R}^{p*p}$ and $\lambda_1, \lambda_2, ..., \lambda_p$ are diagonal elements of the diagonal matrix $\mathbf{D}\epsilon\mathbb{R}^{p*p}$. Eigenvectors are sorted in descending order by their corresponding eigenvalues. First d eigenvectors are called *principal components*. Principal components are linear combinations of original correlated features from a higher dimensional space. How many principal components will be used to the transform data from a higher p-dimensional space into a lower d-dimensional space depends on how much of variability (statistical information) one will keep in the transformed data. Finally, the original data matrix $\mathbf{X}\epsilon\mathbb{R}^{n*p}$ is transformed into a matrix $\mathbf{Z}\epsilon\mathbb{R}^{d*n}$ by

$$\mathbf{Z} = \mathbf{V}_d^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}, \tag{3.11}$$

where $\mathbf{V}_d\epsilon\mathbb{R}^{p*d}$ is a matrix formed of first d eigenvectors as columns. Transformed observations are in columns of $\mathbf{Z}\epsilon\mathbb{R}^{d*n}$.

A downside of PCA is a certain information loss depending on how much of variability one will keep. In addition, there is a loss in interpretability of the transformed data compared to the original features. PCA is based on an assumption that the directions in which the original data vary the most are directions also associated with the response. There is no guarantee that the assumption is always true, but it is a good approximation and PCA often gives good results, according to [22].

Ordinal data

PCA is applicable on numeric data that represent continuous features. If some of features are represented with ordinal data it is questionable whether PCA can be used and how. Based on [23], [24], [25], [26] we choose to treat ordinal data as continuous for PCA.

3.3.4 Longitudinal approach

PPMI is a longitudinal cohort study, i.e. a study in which data is gathered for the same subjects repeatedly over several years and subjects are grouped in separate cohorts (groups of interest) by properties defined in the study design. Cohort studies are common in medicine, psychology and sociology, where they allow researchers to observe changes over time. Repeated measurements of the same subject collected over time are correlated, which is a specific trait of longitudinal studies. Several statistical techniques have been developed for longitudinal data analysis. One of them is derived variable analysis. Derived variable analysis refers to a method that takes a collection of measurements and collapses them into a single meaningful summary feature [27]. According to the same source, the

most common summaries are the average response and the time slope. We decided to use the average, slope, minimum and maximum of repeated measurements as derived statistics in our project.

We faced a problem caused by attrition, i.e. the variability of the number of measurements across subjects. Attrition in longitudinal studies is quite common and thus derived variable method is often more challenging to apply than one would expect. In this project we developed a way to deal with subjects that were participating in the study shorter or longer than others. This method is detailed in 4.2.8 - *Derived Approach*.

3.4 Machine Learning algorithms

Models presented below are based on theory from [22], [28] and other explicitly stated sources. Logistic regresion is chosen as a simple model that often provides good results. Decision tree is used because it is easy to interpret for inference. SVM, random forest and boosted trees are requirements of the project. LSTM is our chosen deep learning model which the inclusion of is one of the requirements of the project.

3.4.1 Logistic Regression

Logistic regression, despite its name, is a classification algorithm, usually applied on a binary class setting, but can be extended to a multiclass setting. Logistic regression models the probability that an observation $x = (x_1, x_2, ..., x_p)$ belongs to one of two classes. Probability $p(x)$ is modeled by means of *logit function* 3.12

$$p(x) = \frac{e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}}{1 + e^{\beta_0+\beta_1 x_1+...+\beta_p x_p}}, \tag{3.12}$$

where coefficients $\beta_0, \beta_1, ..., \beta_p$ are estimated using *maximum likelihood* estimation from the training observations $x_1, x_2, ..., x_n$ and their responding class labels $y_1, y_2, ..., y_n \in \{0, 1\}$. Likelihood is the probability of getting the observed data given the model. When training observations are assumed independent, the likelihood for logistic regression is given with 3.13

$$L(\beta) = \prod_{i=1}^{n} L_i(\beta) = \prod_{i=1}^{n} {p_i}^{y_i}(1 - p_i)^{1-y_i}, \tag{3.13}$$

where $p_i = p(x_i)$, as in 3.12

When coefficients $\beta_i, i = 0, 1, ...p$ are estimated by maximizing the likelihood 3.13 any new, unseen observation $x = (x_1, x_2, ..., x_p)$ is classified by calculating 3.12.

3.4.2 Support Vector Machine

Support vector machine (SVM) is a well known approach for classification that performs well on a variety of classification problems. SVM is used to classify between two classes, although it is possible to accommodate SVM for multiclass classification. SVM is a generalization of maximal margin classifiers which is simple and intuitive, but only applicable on problems where the classes are separable by a linear boundary.

Maximal Margin Classifier

Maximal margin classifier is the hyperplane that perfectly separates training observations in two classes and has the farthest minimal distance from the training observations (i.e. maximal margin) among all other separating hyperplanes, according to [22]. If the data set contains training observations with p features than we talk about p-dimensional feature space and a $(p-1)$-dimensional separating hyperplane, define by 3.14

$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p = 0, \tag{3.14}$$

where $x = (x_1, x_2, ..., x_p)$ is a training observation from p-dimensional space. Training observations from one class will be on one side of the hyperplane and will satisfy 3.15

$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p > 0, \tag{3.15}$$

while on the other side there will be training observations from the second class that satisfy 3.16

$$\beta_0 + \beta_1 x_1 + ... + \beta_p x_p < 0. \tag{3.16}$$

The margin is the minimal perpendicular distance between training observations and the separating hyperplane. If the training observations are separable into two classes then there are infinite many such separating hyperplanes (constructed by shifting or rotating the already existing one) as can be seen in Figure 3.1. The one with the maximal margin is the maximal margin classifier, as seen in Figure 3.1 on the right. The three observations that lie on the dashed line are *support vectors*. They are the closest training

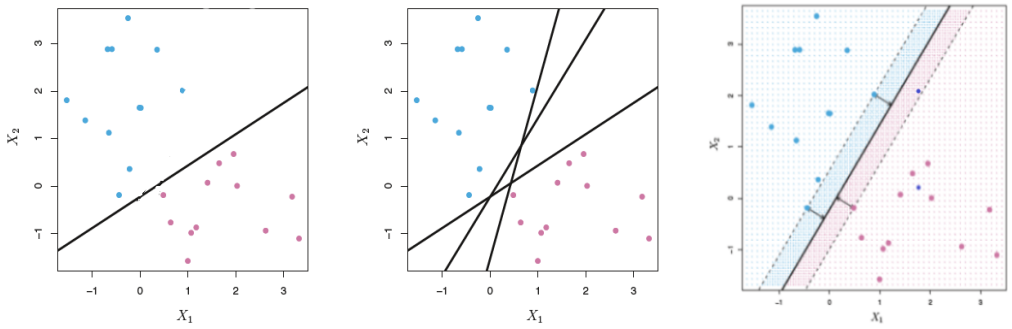


Figure 3.1: Left: Separating hyperplane between to classes. Middle: Several separating hyperplanes between to classes. Right: Maximal margin classifier.

Source: An Introduction to Statistical Learning by James G., Witten D., Hastie T., Tibshirani R.

observations to the separating hyperplane and only they affect the hyperplane. If they move, the hyperplane must accommodate (move). Maximal margin classifier is limited to separable cases. If classes cannot be separated perfectly then the maximal margin classifier cannot be found. Even when the classes are separable, it is not always favorable to find a maximal margin classifier because it can be very sensitive to a change in a single training observation, which suggests overfitting.

Support Vector Classifier

Due to these limitations, the maximal margin classifier is extended to a support vector classifier that is applicable to problems where classes are almost separable by using *a soft margin* which is not so sensitive to changes in the presence of a small number of training observations. A support vector classifier allows for some training observations to be misclassified, i.e. a few observations can be on the wrong side of the margin (dashed line on Figure 3.1) or even on the wrong side of the hyperplane. Margin is called soft because it can be violated by a few training observations. The support vector classifier is the solution to the optimization problem in 3.17

$$\begin{aligned} \max_{\substack{\beta_0,\beta_1,\ldots,\beta_p \\ \epsilon_1,\ldots,\epsilon_n}} M, \text{ subject to} \sum_{j=1}^{p} \beta_j^2 = 1, \\ y_i(\beta_0 + \beta_{i1} + \ldots + \beta_{ip}) \geq M(1 - \epsilon_i) \\ \epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C, \end{aligned} \tag{3.17}$$

where $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$ is the i-th observation, C is a tuning parameter and M is the width of the margin. The tuning parameter C determines the number and the severity of violations - how many training observations can be on the wrong side of the margin or hyperplane. C controls bias-variance trade-off. A small C means the narrow margin is rarely violated, i.e. a classifier that highly fits to the data, which may have low bias and high variance. On the other hand, a larger C means a wider margin that can be violated more often, i.e. the classifier may have higher bias and lower variance. Only training observations that lie on the margin or violate it affect the hyperplane. These observations are called *support vectors*. Larger C leads to larger margin and to more support vectors that affect the classifier as is shown in Figure 3.2.

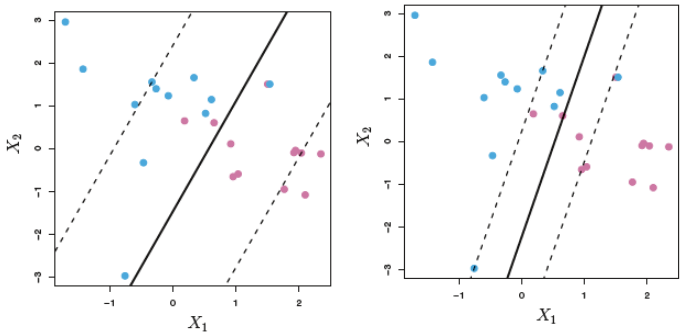


Figure 3.2: Left: Larger C - more support vectors. Right: Smaller C - fewer support vectors.

Source: An Introduction to Statistical Learning by James G., Witten D., Hastie T., Tibshirani R.

The support vector classifier is a natural choice if the boundary between two classes is linear. But, that is often not the case, thus the support vector classifier is further developed into support vector machines.

Support Vector Machine

The support vector machine (SVM) is an extension of the support vector classifier that is based on the idea of "lifting" observations from original feature space to an enlarged, higher dimensional feature space. That "lift" is done by applying appropriate non-linear mapping (transformation) between original, lower and enlarged, higher dimensional feature space. In such an enlarged feature space, observations are separable into two classes by a linear boundary. In the original feature space the boundary between classes is non-linear.

Support vector classifier can be represented as 3.18

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i \langle x, x_i \rangle, \tag{3.18}$$

where α_i are parameters, one for each support vector and $\langle x, x_i \rangle$ is the inner product between $x = (x_1, x_2, ..., x_p)$ and observation x_i given by 3.19

$$\langle x, x_i \rangle = \sum_{j=1}^{p} x_j x_{ij}. \tag{3.19}$$

SVM is the support vector classifier extended by a non-linear *kernel function* or *kernel* $K(x, x_i)$ instead of using an inner product. SVM then has a form 3.20

$$f(x) = \beta_0 + \sum_{i \in S} K(x, x_i). \tag{3.20}$$

The support vectors are the training observations that define the optimal separating hyperplane, i.e. the one with the maximum distance from the nearest training observations. An important benefit of the SVM is that the complexity of the resulting classifier depends on the number of support vectors, but not on the dimensionality of the enlarged feature space.

Many non-linear kernels are popular, e.g. a *polynomial kernel* of degree d 4.4

$$K(x, x_i) = (1 + \sum_{j}^{p} x_j x_{ij})^d, \tag{3.21}$$

or one of the most used, a *radial kernel* 3.22

$$K(x, x_i) = exp(-\gamma \sum_{j}^{p} (x_j - x_{ij})^2), \tag{3.22}$$

where γ is a positive constant.

To train a SVM on a set of training observation means to find coefficients α_i and β_0 and hyperparameters depending on the chosen kernel. For a polynomial kernel that would be a degree d, for a radial kernel it would be γ. Lower d will lead to a boundary closer to linear, just like a smaller γ will lead to a smoother boundary.

3.4.3 Tree Based Algorithms

Another approach to classification that is used in this thesis are tree based algorithms. Such algorithms can be applied to both regression and classification problems, but we will focus on how they are applied in a classification setting. Tree based algorithms are all based around the idea of a segmenting a feature space into multiple simpler regions. A prediction is then made on the basis of this segmented feature space. Commonly, the mean (regression) or mode (classification) of the training observations within a region is used to predict the response of a new observation that falls within this region. The set of splitting rules used to segment the feature space can be summarized in a tree structure. This is why we refer to this family of algorithms as tree based algorithms.

The simplest form of tree based algorithms are decision trees. Decision trees are easy and intuitive to interpret, but they lack the prediction power of more sophisticated models. Combining the predictive power of many such decision trees lead to the more competitive tree based algorithms such as random forest and boosted trees according to [22].

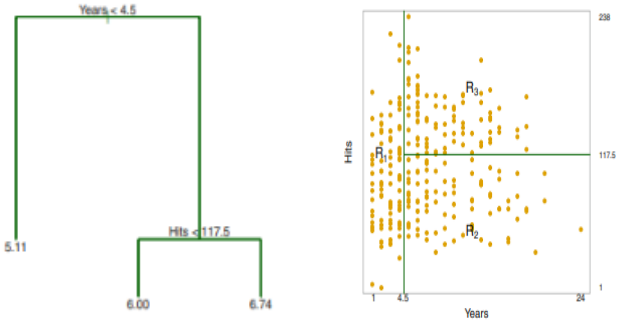


Figure 3.3: Left: Splitting rules summarized in a tree structure. Right: A feature space segmented into three sub-regions.

Source: An Introduction to Statistical Learning by James G., Witten D., Hastie T., Tibshirani R.

Decision Trees

Decision trees are the basic building blocks of tree based algorithms. A single such tree segments a feature space into sub-regions which can then be used to predict on. The segmentation of the feature space is done by a series of splits. A split is done based on a single rule that acts upon a single feature and splits the feature space somewhere along that feature's axis. After the first initial split, subsequent splits are performed within sub-regions of the already split feature space. Each split produces two new sub-regions R_{i_1}, R_{i_2}. Because splits are never performed across two different sub-regions, none of the sub-regions $R_1, R_2, ..., R_n$ overlap with each other.

For each split performed in a classification tree the purity of the resulting sub-regions is maximised. Purity is a measure of the class distribution in a sub-region. A high purity means that a sub-region contains many observations of class $\mathbf{x}$ and few observations of any other class. A low purity means that a sub-regions has a very mixed distribution of classes. Most commonly, purity is maximised by minimizing either the Gini index or the entropy of the two sub-regions resulting from a split.

The Gini index for a sub-region is defined by:

$$G = \sum_{k=1}^{K} p_k(1 - p_k) \tag{3.23}$$

The entropy for a sub-region is defined by:

$$D = -\sum_{k=1}^{K} p_k \log p_k \tag{3.24}$$

In both equations, K is the number of classes.

Both of these quantities take on low values if all p_k are either close to one or zero which can only happen if the sub-region has a high purity. An alternative to maximising the purity of sub-regions is to minimize the classification error rate. The classification error rate is simply defined as the percentage of observations in a region that do not belong to the majority class. In practice this measurement has been shown to not produce good results for trees and therefore maximising purity is preferred [22].

The decision tree algorithm is a greedy algorithm that maximises the purity of the sub-regions at each individual split, never looking ahead to see how this affects purity in subsequent splits. The result can be a non-optimal tree; a better overall solution can be overlooked. Although not always optimal, the decision tree is always reproducible. Given the same data, a set of rules follow from the algorithm in the same logical and non-random fashion.

The bias-variance trade-off is controlled in a decision tree by the amount of splits performed, i.e. by the complexity of a decision tree. There are two common approaches to controlling the complexity of a decision tree. The first approach is to stop growing the tree before all sub-regions have perfect purity. This can be done by either setting the maximum depth of the tree or by only performing splits that result in a gain in purity that exceeds some threshold. The second approach is called tree-pruning and is usually preferred over the first approach. A fully complex tree is first grown and then splits are removed from this fully grown tree in a nested fashion.

Tree-pruning is usually implemented as cost complexity pruning. In cost complexity pruning a tuning parameter α is used to prune the tree. A large α punishes complex trees and therefore as α is increased, more branches are pruned off the tree. The best value of α can be chosen through cross validation.

The advantages of using decision trees over other classification algorithms lie mainly in their ability to provide easy-to-understand model inference. The rules of a decision tree can be summarized in a tree-structure that can be displayed graphically. Unfortunately, their power of inference is not matched by their predictive power. A single decision tree will most likely perform worse than other classification algorithms such as LR or SVM.

Bagging and Random Forest

The fundamental issue of DT is that they individually suffer from a high variance compared to other classification algorithms. It is well known that the variance of a model can be reduced by constructing an ensemble of such models and averaging their predictions together to form a final prediction. In the case of classification, one would not average predictions, but rather take a majority vote. Unfortunately one do not usually possess multiple training sets with which multiple decision trees can be trained. It is however possible to approximate using different training sets by bootstrapping from a single training set to produce multiple bootstrapped training sets. Each bootstrapped training set is produced by repeatedly sampling with replacement from the original training set. N models are then trained from N bootstrapped training sets and their predictions are taken in a majority vote to produce a final classification. Constructing an ensemble of decision trees in this manner is called bagging. Decision trees grown in bagging are usually grown fully and not pruned afterwards. This is because their individual complexity is balanced by their use in an ensemble which reduces their collective variance. Bagging has been demonstrated to give marked improvements in predictive power over a single decision tree [22].

Random forest is an extension of bagging that usually delivers better performance due to a single tweak. As in bagging, decision trees are built from bootstrapped training sets and are then used in an ensemble. The difference is that in a random forest, individual splits in the decision trees are only allowed to consider a random subset of size m out of the p predictors in the training set. Because each split only has subset of predictors to consider, decision trees in random forest tend to grow more random and varied than in bagging. The effect is that decision trees in random forest tend to be less correlated with each other than the decision trees in bagging. Averaging the predictive power of many uncorrelated models produce a stronger decrease in variance than doing the same with

more correlated models. This is why random forest usually delivers better performance than bagging for most data sets. The size m of the subset of predictors that each split is allowed to consider, is a hyperparameter than can be tuned by, for example, cross validation.

Any algorithm that builds an ensemble of models from bootstrapped training sets benefit from an important side effect of using bootstrapped samples. It can be shown that on average, each bootstrapped training set samples around 2/3 of all observations in the original data set. Consequently, this means that each observation in the original training set is not used in approximately 1/3 of all models. If one averages the prediction of all the models that do not make use of an observation to predict upon that observation, one gets a reliable approximation of test error, while not having to fit the model more than once. This is called Out-of-Bag estimation error (OOB). It can be shown that if the number of trees grown in the ensemble is sufficiently large, OOB is equivalent to the leave-one-out cross validation error [22].

Using an ensemble of decision trees increases the predictive power of models compared to using a single decision tree. Unfortunately, the use of such an ensemble reduces the amount of inference a model provides. While a single decision tree can be shown graphically, there is no easy way to graphically represent an ensemble of such trees. However, an overall summary of predictor importance can be obtained from ensemble methods. In the case of classification, the importance of a predictor can be measured by summing up the gain in purity in a tree for all splits considering this predictor and then averaging this value over all trees in the ensemble.

3.4.4 Boosted Trees

Boosted trees is another ensemble method that combines multiple decision trees in a serial manner. The method uses pre-pruning and combines shallow trees, thus producing a model that is very efficient when it comes to memory usage and prediction speed.

While decision trees, bagging and random forest can be explained quite well using non-mathematical terms because they are somewhat heuristical in nature, boosted trees are more rooted in mathematics. It is therefore required that we take a step back and review some basic fundamentals before we proceed. The following explanation is heavily based on these slides by Tianqi Chen [29].

A core concept of supervised learning is the objective function that should be optimized (usually minimized). As we already stated in 3.1.2, an objective function has a form:

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta), \tag{3.25}$$

where $\Theta = \{w_j | j = 1, ..., d\}$ is a set of parameters that will be learned from the data set $X = \{x_1, x_2, ..., x_n\}$, $X \epsilon \mathbb{R}^{n*p}$ where $x_i = (x_{i1}, x_{i2}, ..., x_{ip}) \epsilon \mathbb{R}^p$, $i = 1, 2, ..., n$. $L(\Theta)$ measures the training loss, while $\Omega(\Theta)$ is a regularization term.

Assuming we use an ensemble of trees, we can think of an individual tree as a function mapping a set of input to a certain output. Then the prediction for i-th observation using K trees can be formulated as 3.26:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \epsilon F \tag{3.26}$$

where f_k is a tree/function and F is a space containing all such trees/functions.

The objective function for such a tree ensemble can be defined as 3.27

$$Obj(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k), \tag{3.27}$$

where $\sum_{i=1}^{n} l(y_i, \hat{y}_i)$ measures the training loss for all n observations used to train the model and $\sum_{k=1}^{K} \Omega(f_k)$ measures the complexity of the trees in the ensemble.

We can not use gradient boosting to solve the optimization problem as we need to find f_k which are trees and not numerical vectors. Instead we use something called *additive training*, also called boosting.

We start with a model that returns a constant prediction for all n observations:

$$\hat{y}_i^{(0)} = 0, i = 1, 2, .., n \tag{3.28}$$

and continue building upon that model in iterations called boosting rounds such that:

$$\begin{aligned} \hat{y}_i^{(1)} &= y_i^{(0)} + f_1(x_i) \\ \hat{y}_i^{(2)} = y_i^{(1)} + f_2(x_i) &= f_1(x_i) + f_2(x_i) \\ &.... \\ \hat{y}_i^{(t)} = y_i^{(t-1)} + f_t(x_i) &= \sum_{k=1}^{t} f_k(x_i) \end{aligned} \tag{3.29}$$

In each boosting round, the f_t that is added to the ensemble is the one that minimizes the objective function which is now:

$$Obj(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{t-1} + f_t(x_i)) + \Omega(f_t) \tag{3.30}$$

The magnitude of the training loss part $\sum_{i=1}^{n} l(y_i, \hat{y_i}^{t-1} + f_t(x_i))$ is determined by predictions where the model in the previous boosting round has performed poorly. This essentially means that for each boosting round, a tree is added to the model that is built to perform well on predictions that previous boosting rounds have not performed well on. This is the main strength of boosted trees compared to random forest. Boosted trees is an iterative learning process where the model will hopefully converge to an optimal solution.

3.4.5 Deep Learning - Artificial Neural Networks

Artificial neural networks (ANNs) is a family of algorithms inspired by biological neural networks that constitute a human or animal brain. ANN is built of nodes that are considered to be neurons in the brain, while connections between the nodes correspond to neuron-to-neuron communication through synapses. Nodes are connected in a hierarchical structure, layer over layer where each connection is weighted. ANNs are powerful in finding non-linear predictive models and their structure can be tailored to make them applicable in different areas. Here we start by presenting a simple multilayer architecture and continue with architectures constructed to find patterns in sequences of data, e.g longitudinal data.

Multilayer Perceptron

A multilayer perceptron (MLP) is a network with a simple architecture shown in Figure 3.4. Nodes are interconnected in a feed-forward way, i.e. connections are directed from input layer towards output layer and do not form cycles. Bias nodes are always set to 1 and are added to the network to increase its flexibility. Network is fully connected if each node is connected to every node in the subsequent layer.

MLP is a supervised learning algorithm that when trained on the data set $X = \{x_1, x_2, ..., x_n\}$, $X \epsilon \mathbb{R}^{n*p}$ where $x_i = (x_{i1}, x_{i2}, ..., x_{ip}) \epsilon \mathbb{R}^p$, $i = 1, 2, ..., n$ learns a non-linear approximator f for either classification or regression, such that $f(x_i) = y_i$, where y_i is an outcome corresponding to an observation x_i.

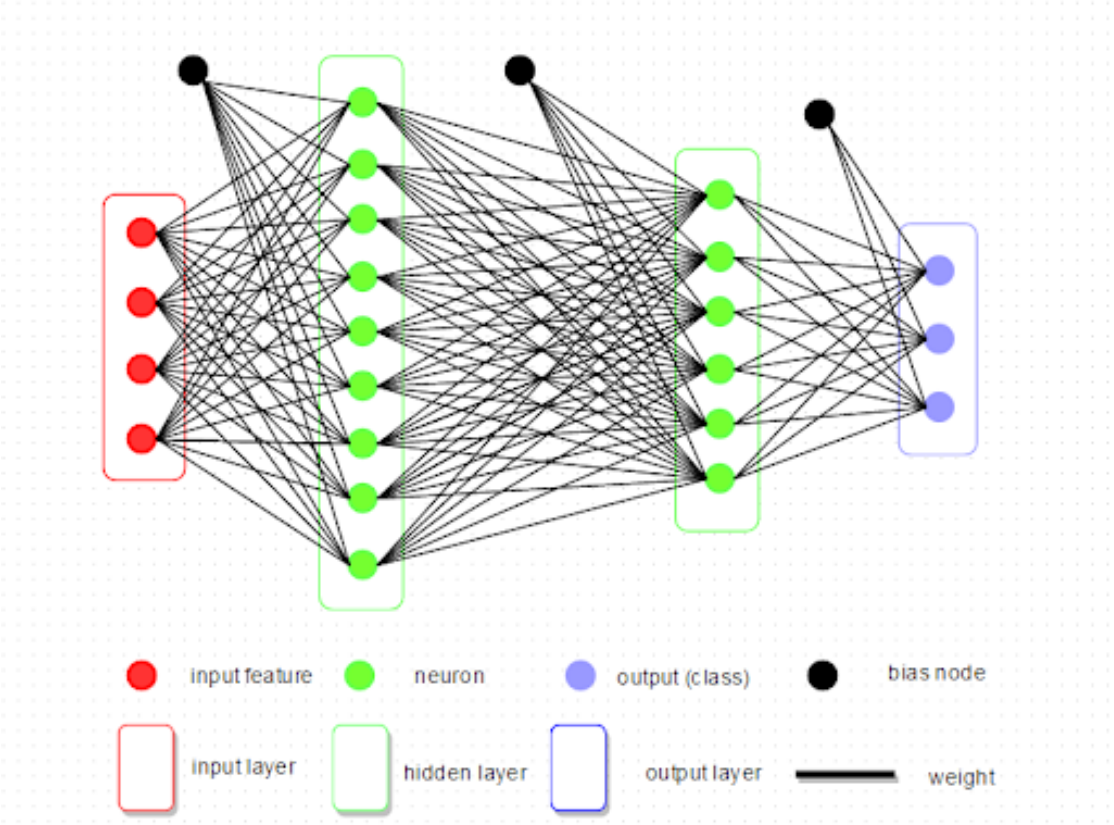


Figure 3.4: Multilayer perceptron

Source: R for Deep Learning (I): Build Fully Connected Neural Network from Scratch - parallelr.com

Input nodes form an input layer and represent input features. There are as many input nodes as there are input features. All connections are weighted and input values from the input layer are transformed in the first hidden layer such that their weighted sum is computed. Further, the weighted sum servers as argument for a non-linear activation function that is activated at each node in the first hidden layer. An activation function is a non-linear function that introduces non-linearity into the network. It is attached to each hidden or output node in the network and determines, based on the input value, whether the node will be activated. The result is that output from each node in the first hidden layer serves as input in subsequent hidden layers where the process continues. Finally, the output nodes receive values from the last hidden layer and transform them into output values z_i.

Training a MPL means to find weights for all connections between nodes. While activation function is the same for all nodes (or at least nodes in one layer) and chosen among many possible non-linear differential functions in the network design phase, weights must be learned from training data. Training starts with a random initialization of weights and

feed-forward computation. Output values z_i differ from target values y_i. Total training error is then 3.31, according to [19]

$$J(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - z_i)^2 \tag{3.31}$$

which should be minimized by updating w, a vector of all weights in the network. Back-propagation algorithm is based on gradient descent: the weights are changed in the direction that will reduce the error $J(w)$

$$\Delta(w) = -\eta\frac{\delta J}{\delta w}, \tag{3.32}$$

where η is *learning rate*, a small number that controls how quickly the model learns, i.e. the amount of change in response to the estimated error each time the model weights are updated.

That gives an iterative procedure to update weights:

$$w(m+1) = w(m) + \Delta(w(m)). \tag{3.33}$$

Thus, training of the MLP is an iterative process that starts by presenting an observation for which the feed-forward computation is done, then backward computation follows to find the change that minimizes $J(w)$, which is then used to update the weights. The iterative process continues with other observations from the training set until the change in training error is under a certain threshold. There are three common training protocols in terms of the way in which observations are presented to the network:

- stochastic training where training observations are presented randomly and the weights are updated accordingly. It is possible that some observations are presented more than once,
- batch training, where each update is based on the average gradient over the presentation of all observations in the data set. If only a subset of the data set is used for each batch, it is called a mini batch,
- online training where each observation is presented only once.

There are only two hidden layers shown in Figure 3.4 for simplicity, but the number of layers, and the number of nodes in each layer, can vary with the nature of the problem at hand. Each hidden layer trains on the previous layer's output, hence it aggregates

and recombine features from the previous layer. More layers means that the network can recognize more complex features, but that can also cause overfitting. Neural networks are also sensitive to feature scaling, i.e. weights from larger features are adjusted more than others. In real life application that means that training data should be standardized before training. Hyper-parameters like number of layers, number of nodes, activation function choice (often sigmoid, hyperbolic tangent, rectified, etc. functions), learning rate, batch size, etc. should be chosen carefully. Hyperparameters are often tuned by hand or using grid or random search.

3.4.6 Recurrent Neural Networks

Recurrent neural network (RNN) differs from a feed-forward network, such as MLP, in a way that it allows connections between neurons in the same or previous layers, i.e. nodes can be connected in directed cycles. Each hidden node is connected to both itself and other hidden nodes. RNN has a temporal dimension and is designed to recognize patterns in sequence data. Computations derived from earlier input are fed back into the network, which gives RNNs a kind of memory. It means that output depends not only on the present inputs but also on the previous step's neuron state. RNN takes the current observation and the values from the hidden nodes from the previous step, applies matrix operation on them and the result serves as argument to an activation function. In each step the same matrix is applied.

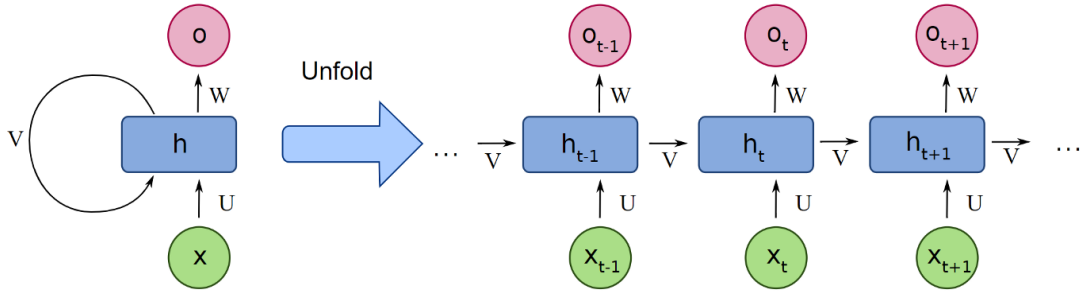


Figure 3.5: Folded and unfolded representation of RNN

Source: Deep Learning: Recurrent Neural Networks by Pedro Torres Perez - deeplearningbrasilia

From the unfolded representation it is easier to understand what is happening at each step. The output from the activation function in the hidden layer is both sent onwards to the output layer and forwarded on to the next iteration of the RNN. Equations 3.34

based on [30] describes the RNN mechanism:

$$\begin{aligned} h_t &= f_h(Ux_t + Vh_{t-1} + b_h) \\ o_t &= f_o(Wh_t + b_o), \end{aligned} \tag{3.34}$$

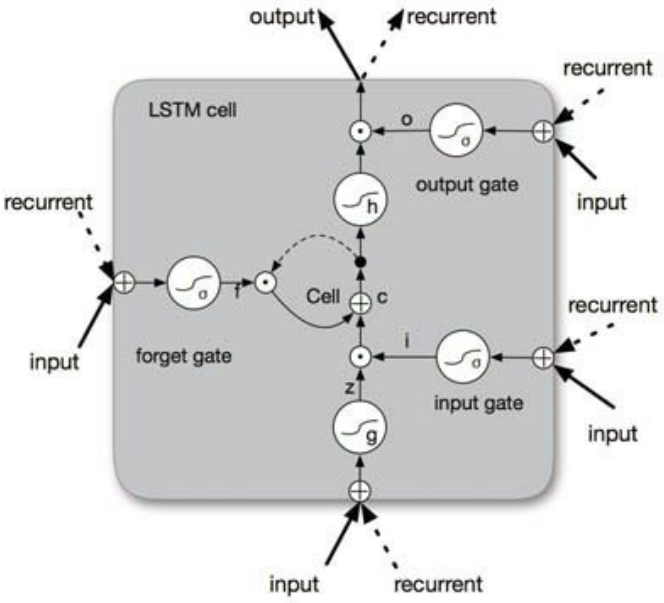
where U, V and W are matrices of weights, same at each time step. Bias terms b_h and b_o are not explicitly visible in Figure 3.5, but they are assumed to be in it. Activation functions are f_h and f_o, while x_t and o_t are input and output at the t-th step. The hidden units from the previous step h_{t-1} influence the computation of h_t which gives a sort of memory to the network.

Using the same matrix at each time step, over many steps, can lead to two problems known as exploding and vanishing gradients. The problems can be experienced with ANNs, due to the depth of the network. Gradients with matrix multiplication can become vary large which means large weights updates during training. This can make networks unstable and leave them unable to learn from the training data. Exploding gradients can be mitigated by regularization to encouraging the weights to be small or by gradient clipping, i.e. when the gradient exceeds some threshold it is scaled down. For the opposite problem of vanishing gradients, one solution is *long short-term memory networks* presented in the next section, 3.4.7.

3.4.7 Long Short-Term Memory Networks

Because of the vanishing gradient problem RNNs tend to suffer from short-term memory and do not learn from dependencies that span across long periods of time. Long short-term memory (LSTM) networks are constructed to solve that problem. Their building blocks are LSTM units, which have the same function as hidden nodes in RNN, commonly built of a *cell* and three multiplicative *gates*. The cell is the memory part of the unit and keeps track of the dependencies in inputs over time, while three gates, *input, output* and *forget* gate control the proportions of information to forget and to pass on to the next time step [31]. Several unit architecture are in use, only a basic one is shown in Figure 3.6 taken from [31].

Formally, according to the same source [31], an LSTM unit update mechanism at time t is described by 3.35



"The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic sigmoid function.", from [32].

Figure 3.6: LSTM unit schematic

Source: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF by X. Ma and E. Hovy [31]

$$\begin{aligned} i_t &= \sigma(W_i h_{t-1} + U_i x_t + b_i) \\ f_t &= \sigma(W_f h_{t-1} + U_f x_t + b_f) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_c h_{t-1} + U_c x_t + b_c) \\ o_t &= \sigma(W_o h_{t-1} + U_o x_t + b_o) \\ h_t &= o_t \odot \tanh(c_t), \end{aligned} \tag{3.35}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. x_t is the input vector at time t, and h_t is the hidden state vector storing all the useful information at (and before) time t. U_i, U_f, U_c, U_o denote the weight matrices of different gates for input x_t, and W_i, W_f, W_c, W_o are the weight matrices for hidden state h_t. b_i, b_f, b_c, b_o denote the bias vectors.

3.5 Model Validation and Tuning

3.5.1 Performance Measures

Performance of a binary classifier can be measured in many ways. Here, we only present the measures (metrics) that will be used later in the project. They are derived from *the*

confusion matrix for a two class problem. The confusion matrix for a two class problem is a two rows by two columns table that reports the number of true positives, false positives, false negatives and true negatives predictions, as shown in Table 3.1.

	Actual positive	Actual negative
Predicted positive	True positives TP	False positives FP
Predicted negative	False negatives FN	True negatives TN

Table 3.1: Confusion matrix

True positives (TP) and true negatives (TN) are the correct predictions, while false negatives (FN) and false positives (FP) are the incorrect predictions. Some popular measures that are used in this project are: *accuracy, precision, sensitivity, specificity* and *F1 score*. Their definitions are listed below:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.36}$$

$$precision = \frac{TP}{TP + FP} \tag{3.37}$$

$$sensitivity = \frac{TP}{TP + FN} \tag{3.38}$$

$$specificity = \frac{TN}{TN + FP} \tag{3.39}$$

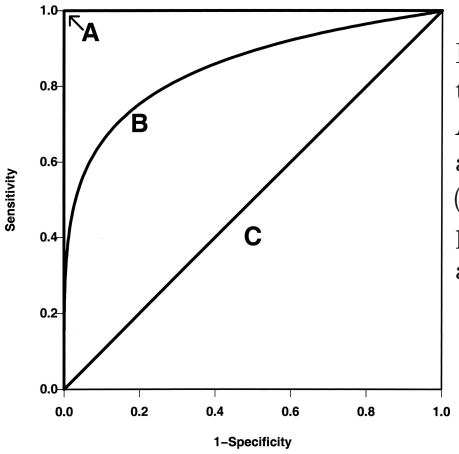
$$F_1 = \frac{2TP}{2TP + FP + FN} \tag{3.40}$$

For a reader more familiar with other terms in use: precision can be called positive predictive value, while sensitivity is also known as recall.

Sensitivity and specificity are often used as measures in a clinical setting (e.g. performance of a medical test). Sensitivity measures how often a test recognizes patients with the condition that is tested for. Specificity measures the ability of a test to recognize patients without the condition that is tested for. Generally, there is trade-off between sensitivity and specificity [33], i.e. an increase in sensitivity is often accompanied by decrease in specificity and vice versa.

Another popular performance metric that we use is area under the ROC curve. ROC (receiver operating characteristic) curve compares sensitivity versus specificity across

various threshold settings. The ROC curve is created by plotting the sensitivity against (1 - specificity) at various probability thresholds. Area under the ROC curve is another measure of a classifier's performance. The larger the area, the better the performance, as is shown in Figure 3.7



Hypothetical ROC curves on Figure 3.7 representing the diagnostic accuracy of the gold standard (lines A; AUC=1), a typical ROC curve (curve B; AUC=0.85), and a diagonal line corresponding to random chance (line C; AUC=0.5). As diagnostic test accuracy improves, the ROC curve moves toward A, and the AUC approaches 1.

Source: Circulation, Vol 115, No 5, Feb. 6 2007, Receiver-Operating Characteristic Analysis for Evaluating Diagnostic Tests and Predictive Models by Kelly H. Zou, A. James O'Malley and Laura Mauri

Figure 3.7: ROC curves

The last performance measure we use in this project is Matthews correlation coefficient (MCC) because it has some advantages over F1 score and accuracy on imbalanced data sets, according to [34]. MCC can be directly calculated from the confusion matrix using 3.41

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}. \tag{3.41}$$

3.5.2 Hyperparameter Tuning

Hyperparameter tuning is a process that consists of finding a set of optimal hyperparameters for a particular learning algorithm. A hyperparameter is a parameter that defines the model architecture and has to be chosen before the learning process begins. Some examples of hyperparameters are: depth of a decision tree, number of trees in random forest, number of neurons and layers in a neural network, misclassification penalty in SVM, etc. Model parameters values are learned through the learning (training) process when the loss function is optimized, but hyperparameters cannot be optimized from the data. Hyperparameters have a strong influence on the algorithm's prediction accuracy (performance) and often differ for different data sets. Several methods have been developed for hyperparameter tuning, some of them are automated. Here, we only present methods that will be used later in the project.

Grid Search

Grid search, usually in combination with manual search [35], is a traditional hyperparameter tuning method. Its idea is simple: chose a grid of values for hyperparameters, based on experience or guessing and evaluate model performance in each of the grid points to find the best hyperparameter combination. The result depends on the initial choice of the minimum and maximum grid points which are usually set by manual search. This method is computationally expensive in terms of total number of calculations, because it checks all grid points, but it can usually be parallelized.

Random Search

Random search [36] is a variation of the grid search idea. Random search does not search over the entire grid, but over a random sample of points on the grid. Thus it is much cheaper than grid search, but mostly performs as well as grid search. According to [37] "if at least 5% of the points on the grid yield a close-to-optimal solution, then random search with 60 trials will find that region with high probability", where high probability means 0.95. Random search is easy to parallelize, takes fewer validation trials, thus it is more efficient, if the condition from the citation is satisfied.

3.5.3 Cross Validation

Cross-validation is a method for model performance evaluation, i.e. for assessing how the model will perform on an unseen data set (e.g. test set). The most commonly used version is *k-fold cross-validation*, where k is a chosen number, often 5 or 10. The training data set is split in k partitions and the model is trained k times. Each time the model is trained on $k-1$ partitions that are used as a training set, while the remaining k-th partition is used as a test set. Each time the $k-1$ partitions are chosen differently from the initial k partitions, thus each time a different partition is used as a test set. For each of the k training runs, performance measures are computed. Finally, the average performance is calculated for the k training runs. K-fold cross-validation, for k=5 is shown in Figure 3.8

Cross-validation allows the model to train and validate on the same subset of data. When the training data set is small, this is especially useful, because there is no need for splitting the training data further in order to have a validation set. In addition, k-fold cross-validation gives k performance evaluations, which can be similar or not and this can lead to some conclusions about the data set or the model. These performance evaluation

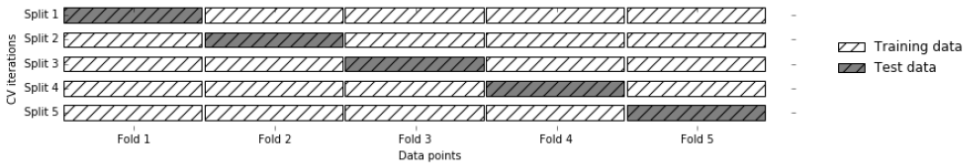


Figure 3.8: 5-fold cross-validation

Source: Introduction to Machine Learning with Python by Andreas C. Müller and Sarah Guido

results can show if the model is sensitive to changes in the data set and roughly what would be the performance of the model in the worst and best case.

Nested Cross Validation

Nested cross-validation is a procedure for estimating the generalization error of the model and its corresponding tuning procedure. When cross-validation is used for both tuning hyperparameters and model evaluation, the same test folds from k-fold cross-validation is used in both procedures. That causes, according to [38], information "leak" into the model and overfitting, i.e. model evaluation will be overly-optimistic. The size of the problem depends on the size of the data set and model stability. To avoid this problem, nested cross-validation uses a series of data set splits and two nested cross-validation loops. The inner loop is for hyperparameter tuning (model selection), while the outer loop is for unbiased model evaluation. A sketch of the algorithm is presented in Figure 3.9

The inner loop contains cross-validation and random or grid search implementation. The outer loop provides training set to the inner loop, while test sub-set from the outer loop is held back. Although authors claim in [38] that, if the model has limited number of hyperparameters, nested cross-validation is probably not needed, we decided to use it in this project to get a better estimate of the generalization error of our models.

3.5.4 Train Test Split

Machine learning model training begins with a given data set. A model should be trained, its performance evaluated, and finally tested. For that purpose, a data set is usually split into training, validation and test sets. The role of each of them is as follows: the training set is used to train the model, the validation set is used to tune the hyperparameters of the model and the test set is used to assess the performance of the model. When the original data set is small (as it is in this project) cross-validations methods are often used

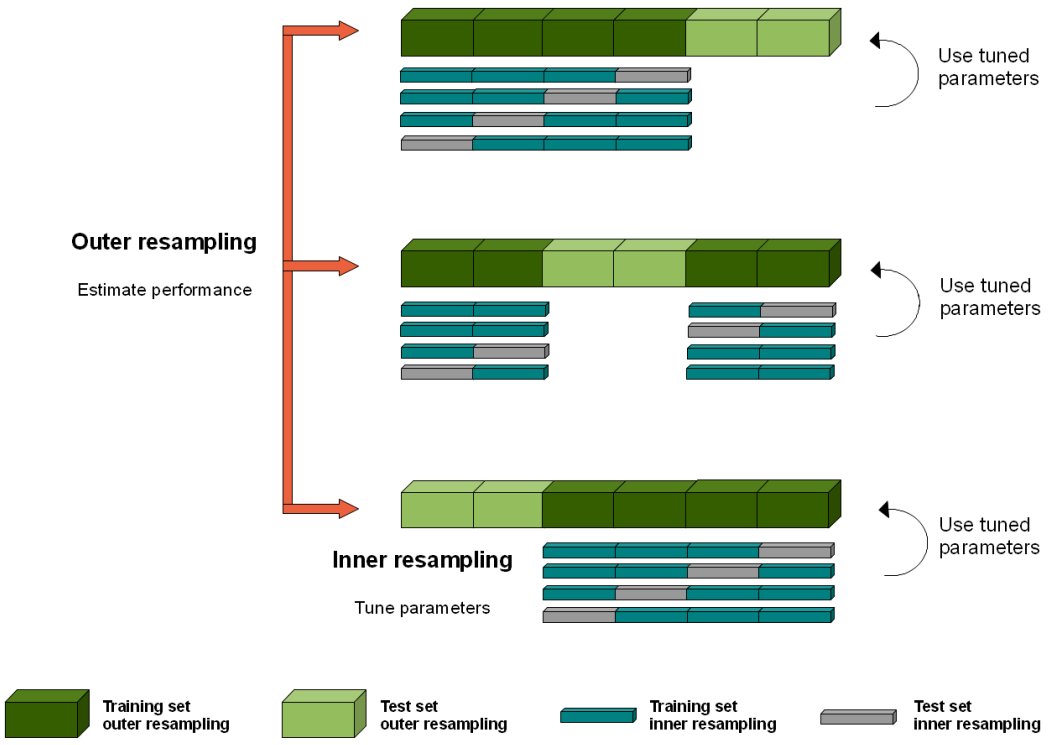


Figure 3.9: Nested cross-validation

Source: https://i.stack.imgur.com/vh1sZ.png or Nested cross validation explained by Weina Jin

and the validation set is omitted. There is not a specific rule to select the proportions of the original data set that should be used for the training and test sets. It depends on the size of the data set and the algorithm that will be trained. A common choice is a 70/30 or 80/20 split. We use a 70/30 split in order to have a large enough training set, but at the same time get stable results from our test set.

Chapter 4

Methods and Materials

This chapter goes through the set up and execution of the project. We start off with our data set, explaining our cohort and feature selection. We then go on to describe our preprocessing pipeline and the different prediction approaches used. We finish the chapter by explaining how we use our models, as well as our tuning and model validation procedures.

4.1 Data Set

In this section we go through the steps taken to collect our data set from the PPMI database.

4.1.1 Cohort Selection

Our data set is constructed from data downloaded in December 2019 from the PPMI database, as briefly mentioned in 3.2. PPMI has data available for various cohorts. An overview of PPMI and the various cohorts in the study can be found in 2.2. Of these cohorts, we can only use data from the cohorts *De Novo PD Subjects* and *Genetic Cohort Subject*. Subjects in these cohorts are the only subjects in the study with confirmed Parkinson's disease. By selecting subjects from these two cohorts only, we reduce the amount of subjects available to us from 2200 to 683. Of these 683 subjects, 423 are from the De Novo Cohort and 260 are from the Genetic Cohort. The eligibility criteria of these cohorts are different. The genetic cohort is for example, less strict in terms of disease length and progression, than the De Novo cohort. The criteria can be viewed in Appendix A. In this appendix, the criteria highlighted in green is the same between the two cohorts and therefore poses no problem. The rest of the criteria are different. After

consulting and discussing with an MD, the differences between criteria highlighted in red were deemed problematic, while the differences between non-highlighted criteria were deemed non-problematic.

Subjects from the genetic cohort were therefore filtered so only subjects that fulfill the following criteria are used:

- Subject is older than 30
- Subject has had a PD diagnosis for no more than two years
- Subject is at Hoehn and Yahr stage 2 or less
- Subject is currently not using PD medication

Selection process is illustrated in Figure 4.1.

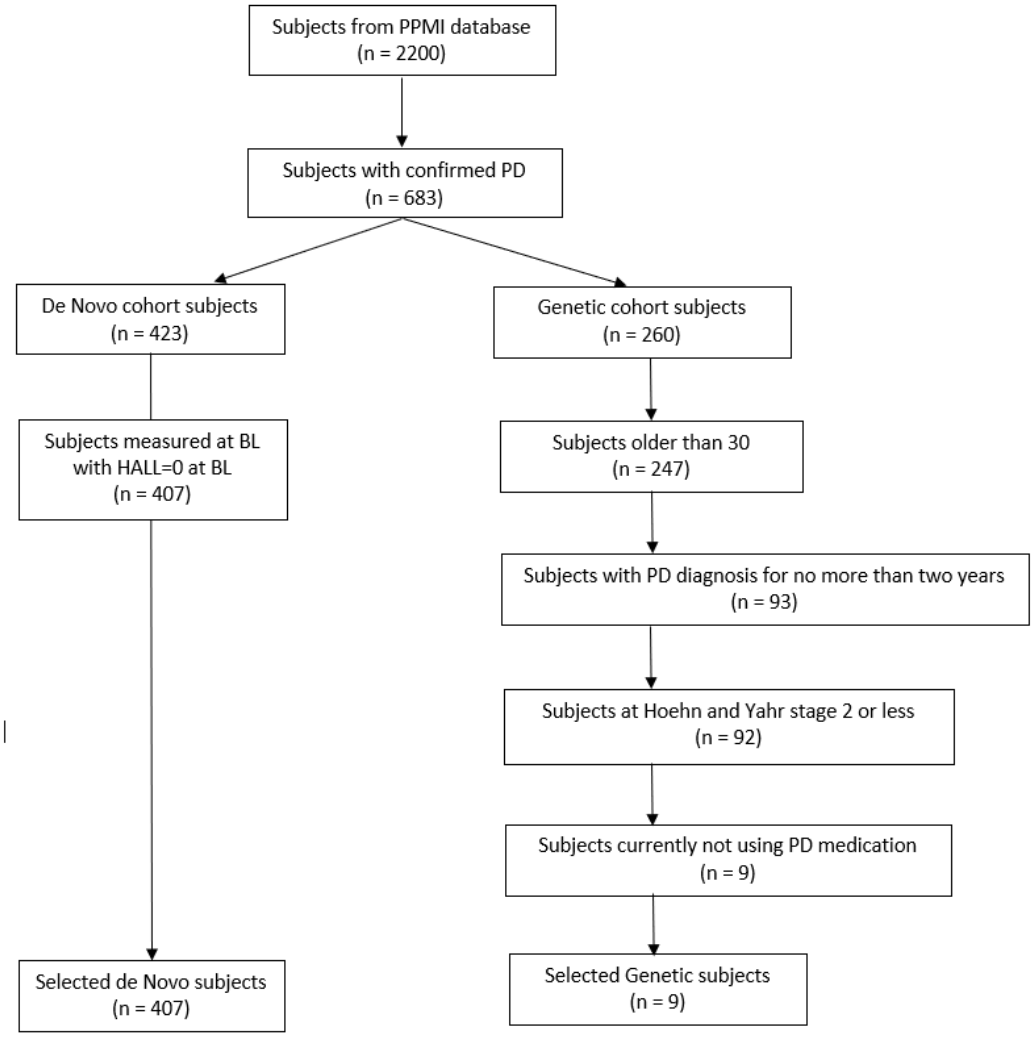


Figure 4.1: Subject selection process

After filtering by these criteria we are left with only 9 subjects from the genetic cohort, which reduces the subjects available to us from 683 to 432, with 423 subjects in the De Novo cohort and 9 subjects in the genetic cohort.

Finally we have to filter out subjects from both cohorts that do not fulfill the following criteria:

- Subject has baseline data
- Subject does not experience symptoms of psychosis at baseline

This reduces the number of subjects in the De Novo cohort from 423 to 407, leaving us with 416 eligible subjects from both cohorts.

4.1.2 Feature Subset

The PPMI database consists of multiple data tables, each containing multiple features. We limit the number of features in our data set by only collecting a specific set of features from a limited number of data tables for each subject. We perform this feature selection based mostly on features used in Barret [3] and Ffytche [5]. In addition, we include some demographic features as it is considered common practice. Two additional features are included based on the recommendation of an MD working on Parkinson's disease. Lastly, we collect our response and two additional features (NP1COG, PD_MED_USE) based on the nature of the prediction, additional comments from Barret and [39].

The collected data set does not contain any features that depend on MRI scans or biological samples (blood, cerebrospinal fluid, etc.). Such data is available in the PPMI database, but it is often expensive and time-consuming data to collect, and therefore less available. An important reasoning behind our choice of feature subset is to base ourselves on data that is more easily collected like clinical tests, questionnaires and demographic data. Such data is often collected for PD patients. This should make our models usable for a wider selection of PD patients than would have been the case if we included the less available features, like the MRI scans mentioned previously.

Whenever possible we collect component features, i.e the partial scores/answers to a test instead of a total final score. For example, a hypothetical test A has six scored tasks and a total score is calculated by summing up the partial scores. In this scenario we would collect the six partial scores and not the total score. The total score can be calculated from the partial scores later. We prefer component features because it gives us a greater freedom in how we proceed with preprocessing, giving us more options for

feature reduction, in contrast with Barret [3] and Ffytche [5] who base themselves on total scores. The formulas for calculating the total scores can be found in Appendix B.

As a note, Barret [3] and Ffytche [5] makes use of T-scores or scaled scores for many of their features. These are scaled versions of raw total scores, often scaled by population mean and population standard deviation. When the population mean and standard deviation are not known, then the sample mean and sample standard deviation are used. T-scores available in PPMI data are scores standardized to have a mean of 50 and a standard deviation of 10, as shown in 4.1.

$$T = 10(\frac{X - \mu}{\sigma}) + 50, \tag{4.1}$$

where X is a feature with mean μ and standard deviation σ. While such scores are useful in order to compare a subject's score to that of an average member of a population, it should have no bearing on machine learning algorithms. Each observation for a feature would be scaled by the same magnitude and the relative relationship between observations would stay the same. We can therefore use raw total scores as equivalent features. We scale our features as part of our preprocessing pipeline later, but this is to bring all our features into the same scale which can be helpful for many algorithms.

Based on Barret and Ffytche we include features from the following data tables containing corresponding psychometric test measurements. Every test measurements below are used in at least one of the articles:

- Montreal Cognitive Assessment (MoCA) - an assessment test for cognitive function in a patient. It is often used as a cognitive screening tool for a number of illnesses such as Parkinson's disease or Alzheimer's disease [40].
- Hopkins Verbal Learning Test - Revised (HVLT) - a test of verbal, short-term memory/new learning in a patient, primarily in populations with brain-disorders. It is useful in populations that require follow-up through neuropsychological assessment and to examine whether treatment has been effective [41].
- Benton Judgement of Line Orientation Test (BJLO) - a measure of spatial perception and orientation of a patient [42]. The test consists of 30 tasks, but only 15 of these are used for one test iteration. The test alternates between using the first 15 tasks and the last 15 tasks for testing purposes. The total raw score of a subject for one iteration is comparable between the alternative iterations.
- Letter Number Sequencing (LNS) - a measure of verbal working memory.

- Semantic Verbal Fluency (SVF) - the test entails that the patient must generate words from a given category within a preset time of 60 seconds. According to [43] it is a quick, easy-to-apply test, highly sensitive and specific in the diagnosis of dementia, which justifies its use in detection of cognitive decline.
- REM Sleep Disorder Questionnaire (RBD) - contains a set of questions that should be answered either 'yes' or 'no' in order to asses the most prominent clinical features of REM sleep disorder [44].
- Geriatric Depression Scale Short (GDS) - a test used to identify depression in the elderly [45].
- University of Pennsylvania Smell Identification Test (UPSIT) - used to test the function of an individual's olfactory system, it indicates how the patient does in accordance to his/hers age group and gender [46].
- Epworth Sleepiness Scale (ESS) - assesses daytime sleepiness in adults while performing usual activities [47].
- Scales for Outcomes in Parkinson's Disease - Autonomic Dysfunction (SCOPA-AUT) - used to evaluate autonomic[1] symptoms in patients with Parkinson's disease. The scale is a self-completed questionnaire that consists of 25 items in the following categories: "Gastrointestinal (7), Urinary (6), Cardiovascular (3), Thermoregulatory (4), Pupillomotor (1), and Sexual (2 items for men and 2 items for women)" [49].
- Symbol Digit Modalities (SDM) - screens for cognitive impairment by using a simple substitution task. The subject has 90 seconds to match numbers with geometric figures [42].

Features extracted from aforementioned tests are presented in Table 4.1, while further details about variable definitions and score calculations are contained in Appendix B.

We collect features from the following three data tables based on Barret [3] and Ffytche [5], the recommendations of an MD working on Parkinson's disease, the nature of the prediction task and [39].

- MDS-UPDRS Part 1
- MDS-UPDRS Part 2
- MDS-UPDRS Part 3

[1] "The part of the nervous system responsible for control of the bodily functions not consciously directed, such as breathing, the heartbeat, and digestive processes." [48]

Test	Extracted features
MoCA	MCAALTTM, MCACUBE, MCACLCKC, MCACLCKN, MCACLCKH, MCALION, MCARHINO, MCACAMEL, MCAFDS, MCABDS, MCAVIGIL, MCASER7, MCASNTNC, MCAVF, MCAABSTR, MCAREC1, MCAREC2, MCAREC3, MCAREC4, MCAREC5, MCADATE, MCAMONTH, MCAYR, MCADAY, MCAPLACE, MCACITY
HVLT	HVLTRT1, HVLTRT2, HVLTRT3, HVLTRDLY, HVLTREC, HVLTFPRL, HVLTFPUN
BJLO	JLO_TOTRAW
LNS	LNS1A, LNS1B, LNS1C, LNS2A, LNS2B, LNS2C, LNS3A, LNS3B, LNS3C, LNS4A, LNS4B, LNS4C, LNS5A, LNS5B, LNS5C, LNS6A, LNS6B, LNS6C, LNS7A, LNS7B, LNS7C
SVF	VLTANIM, VLTVEG, VLTFRUIT
RBD	DRMVIVID, DRMAGRAC, DRMNOCTB, SLPLMBMV, SLPINJUR, DRMVERBL, DRMFIGHT, DRMUMV, DRMOBJFL, MVAWAKEN, DRMREMEM, SLPDSTRB, STROKE, HETRA, PARKISM, RLS, NARCLPSY, DEPRS, EPILEPSY, BRNINFM, CNSOTH
GDS	"GDSSATIS", "GDSDROPD", "GDSEMPTY", "GDSBORED", "GDSGSPIR", "GDSAFRAD", "GDSHAPPY", "GDSHLPLS", "GDSHOME", "GDSMEMRY", "GDSALIVE", "GDSWRTLS", "GDSENRGY", "GDSHOPLS", "GDSBETER"
UPSIT	UPSITBK1, UPSITBK2, UPSITBK3, UPSITBK4
ESS	"ESS1", "ESS2", "ESS3", "ESS4", "ESS5", "ESS6", "ESS7", "ESS8"
SCOPA -AUT	"SCAU1", "SCAU2", "SCAU3", "SCAU4", "SCAU5", "SCAU6", "SCAU7", "SCAU8", "SCAU9", "SCAU10", "SCAU11", "SCAU12", "SCAU13", "SCAU14", "SCAU15", "SCAU16", "SCAU17", "SCAU18", "SCAU19", "SCAU20", "SCAU21","SCAU22", "SCAU23", "SCAU24", "SCAU25"
SDM	SDMTOTAL

Table 4.1: Features extracted from psychometric tests

MDS-UPDRS is Movement Disorder Society sponsored revision of Unified Parkinson Disease Rating Scale [50] used to estimate the development of PD in patients. It covers all the movement hindrances of PD and consists of 5 parts. Features extracted from the MDS-UPDRS parts are presented in Table 4.2, while further details about variable definitions and score calculations are contained in Appendix B.

NP1HALL measures whether or not a subject has experienced psychotic symptoms the week preceding the test. This is also the response used in Barret [3] and Ffytche [5]. It is important to mention that the criterion of "has experienced psychotic symptoms the week preceding the test" is a weakness inherent in our response. Technically, in extreme cases, this criterion means that a patient might experience symptoms of psychosis all weeks of the year, with the exception of the week prior to his/hers yearly test, and still be classified as non-psychotic. Unfortunately, NP1HALL is the best indicator of psychosis for a PD patient available in the PPMI database.

Test	Extracted features
MDS-UPDRS 1	NP1HALL, NP1COG
MDS-UPDRS 2	NP2TRMR, NP2WALK, NP2FREZ
MDS-UPDRS 3	NP3BRADY, NP3FACXP, NP3FRZGT, NP3FTAPL, NP3FTAPR, NP3GAIT, NP3HMOVL, NP3HMOVR, NP3KTRML, NP3KTRMR, NP3LGAGL, NP3LGAGR, NP3POSTR, NP3PRSPL, NP3PRSPR, NP3PSTBL, NP3PTRML, NP3PTRMR, NP3RIGLL, NP3RIGLU, PN3RIGRL, NP3RIGN, NP3RIGRU, NP3RISNG, NP3RTALJ, NP3RTALL, NP3RTALU, NP3RTARL, NP3RTARU, NP3RTCON, NP3SPCH, NP3TTAPL, NP3TTAPR, PD_MED_USE, NHY

Table 4.2: Features extracted from MDS-UPDRS

NP1COG measures whether or not a a subject has experienced cognitive impairment the week preceding the test. According to Thanvi et al [39], *"Old age, **cognitive impairment**, history of depression, and sleep disorders are important risk factors for the development of psychosis in PD.* Thus we collect NP1COG due to a relation between cognitive impairment and psychosis.

TD/PIGD[2] classification is one of the features recommended by an MD working on Parkinson's disease. TD/PIGD calculation is based on features from MDS-UPDRS 1 and 2 and described in Appendix B.

Lastly, PD_MED_USE feature tells us which PD medications a subject is currently taking. According to Barret [3], the relationship between PD medication and psychotic symptoms is complex, thus we believe it could be relevant to include information about the PD medication a subject takes as a feature.

MDS-UPDRS 3 is taken twice by each subject, before and after PD drugs are administered to the subject. The purpose of this is to measure the improvement in post drug-administration motor-symptoms. We only collect data from the pre drug-administration test. Barret [3] also chose to not use data from MDS-UPDRS 3 post-drug administration, as the relationship between medication and psychosis is *"not unidirectional, i.e., the occurrence of psychotic symptoms may influence medication choice."*

Demographic data is collected from four different data tables:

- Family History PD
- Socio-economics

[2]Postural Instability and Gait Disorder

- Screening Demographics
- PD Features

Features extracted from the demographic data tables are presented in Table 4.3, while further details about variable definitions and calculation of age and diagnosis length are contained in Appendix B.

Test	Extracted features
Family History PD	BIOMOMPD, BIODADPD, FULSIBPD, HAFSIBPD, MAGPARPD, PAGPARPD, MATAUPD, PATAUPD, KIDSPD
Socio-economics	EDUCYRS
Screening Demographics	BIRTHDT, GENDER
PD Features	PDDXDT

Table 4.3: Demographic features

We collect family history of PD because it might capture some of the genetic component of Parkinson's disease. The number of education years from socio-economics might have an effect on various cognitive measures. We calculate age which is an important predictor for disease development and has an effect on various test scores. We also calculate the number of years a subject has had a diagnosis of PD, which is an important indicator of psychosis and general decline in health. Gender is included because the incidence rate of PD is not equally distributed amongst the genders.

Finally, we use variables PATNO, ENROLLDT and EVENT_ID as auxiliary variables in the process of selecting subjects and merging tables. These are not considered as features and not used in training the prediction models later.

Meaning of used variables is given in Appendix C.

4.1.3 Merging the Data Tables

Our data set now consists of individual data tables containing features we have selected. We merge them using variables PATNO and EVENT_ID variables.

Some of the data tables are non-longitudinal, i.e data for a subject does not vary with time. Examples of such data tables are the demographic ones. Also included here is the University of Pennsylvania Smell Identification Test as it is only performed once per subject; at baseline. This can be seen in the activity schedule in Appendix D, in the row "Olfactory Testing (UPSIT)". We first merge the non-longitudinal data tables together

using the key PATNO, which is a unique ID for each patient. After this first merge we are left with a combined data table with one entry per subject, i.e no longitudinal data.

The rest of the data tables are longitudinal. In order to merge these data tables together with our non-longitudinal one, we have to introduce a secondary key: EVENT_ID. To understand how EVENT_ID relates to time and in general how our longitudinal data is organised, it is essential to have a quick look at the activity schedule in Appendix D. What is referred to as "Visit Number" in the activity schedule is the same as EVENT_ID. By merging on this key and PATNO, we group entries for each subject longitudinally. Non-longitudinal data is simply copied for each longitudinal entry. We can observe from the activity schedule that the data for all the tests we use is only collected at yearly intervals. This translates to Visit Number BL, V04, V06, V08, V10, V12, V13, V14 and V15. This means that we can only make use of these yearly data entries, and not the entries in between like V02, V07, etc. For each subject we therefore have a maximum of 9 longitudinal entries. The number might be lower than this for a subject due to missing entries or because the subject has dropped out of the study.

We must also mention that there is a peculiarity regarding MoCA in the schedule. MoCA is not done at BL, but rather performed at a screening no more than 45 days prior to BL. This is due to the fact that the results on MoCA are used to include/exclude subjects from the cohorts and is therefore needed at screening. According to Vogel et al [2015] [51], MoCA has a high correlation to various neuropsychological tests even when MoCA is performed up to 180 days prior to these tests. We justify our inclusion of MoCA in the baseline data based on this.

After merging the longitudinal data tables with our non-longitudinal one, we are left with one combined data table, with up to 9 longitudinal entries for each subject. Non-longitudinal data is copied for each longitudinal entry.

4.1.4 Post Merging Operations

Before we move on to missing value imputation we perform some additional post merging operations. This includes calculating age and duration of PD diagnosis for each subject for each longitudinal entry (as described in Apendix B). We remove enrollment date (ENROLLDT), birth date (BIRTHDT) and date of first PD diagnosis (PDDXDT) after deriving these features. Post merging is also technically where we filter out subjects from the genetic cohort that do not meet the eligibility criteria. Finally, we remove subjects from our data set that present with psychosis at baseline, as we are interested in predicting future psychosis and these subjects are already classified as suffering from psychosis (see Figure 4.1).

After the post merging operations we have a data set with 2250 longitudinal entries, 416 unique subjects and 185 features, including our response. If we divide our 2250 longitudinal entries by our 416 unique subjects we have on average ca. 5.4 longitudinal entries per subject. This is significantly lower than the maximum of 9 entries per subject. This indicates that we are dealing with missing entries and/or subject dropout from the study. Figure 4.6 illustrates attrition in our data set.

4.2 Preprocessing

This section details most of our preprocessing steps after data collection. Some of these steps are always performed and some are optional depending on how we want to use our data. Figure 4.2 illustrates the process.

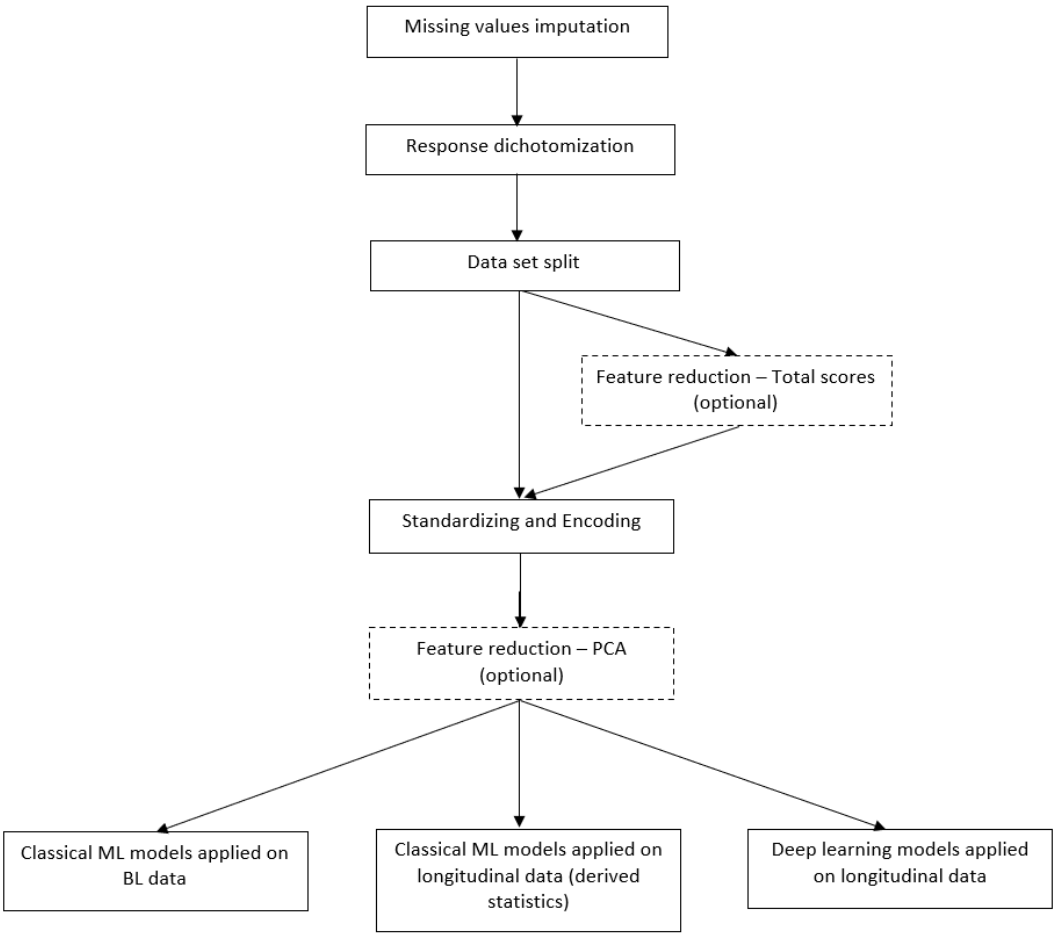
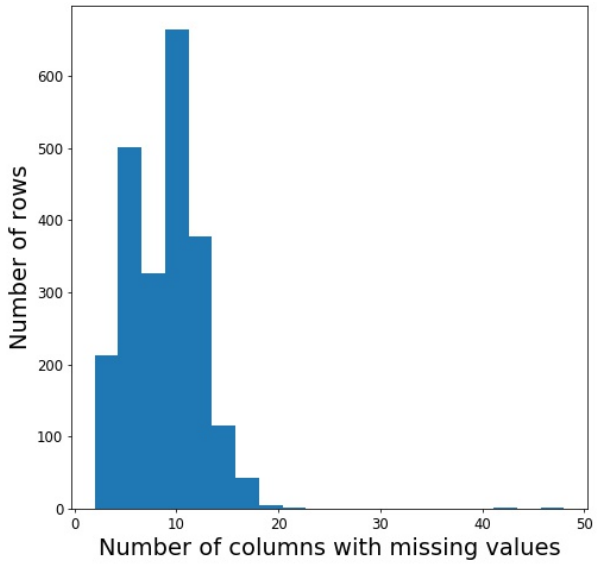


Figure 4.2: Preprocessing steps

4.2.1 Missing values

In the context of missing value imputation, it is easier to visualize missing values in terms of rows and columns instead of data entries and features. In this subsection we will therefore refer to a data entry as a row and a feature as a column.

All rows have missing values before imputation, from a minimum of 2 missing values to a maximum of 48 missing values in a row. Figure 4.3 shows the distribution of columns with missing values among rows, while Figure 4.4 shows the distribution of columns with missing values among patients.

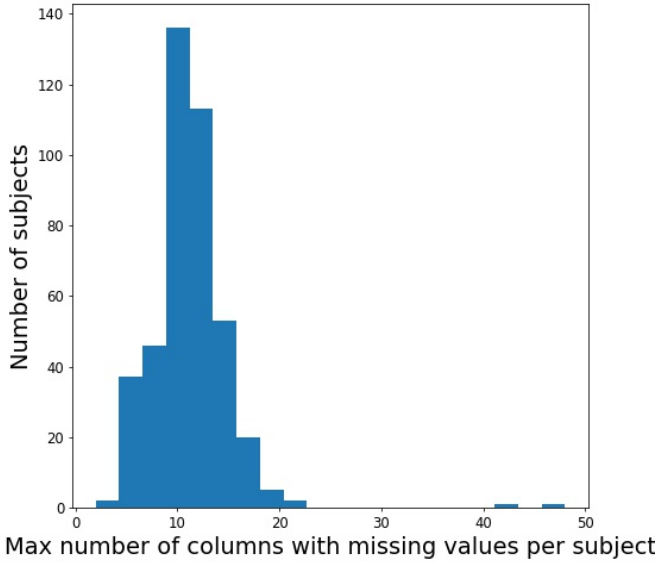


The majority of rows have few missing values and only a few rows have more than 15 missing values.

Figure 4.3: Distributions of columns with missing values per row

We use different strategies for replacing missing values depending on the feature under consideration. Columns that describe the family history of a subject is filled with zeroes. The reasoning behind this is that if such a column is left blank, the subject most likely lacks information for the column or the column is not applicable for the subject. In the first case, we can impute 0 with a high degree of certainty because the incidence rate of PD is low. In the second case we can impute 0 because the column deals with a hypothetical situation that does not exist in reality. e.g a subject without kids is asked whether or not their kids have a PD diagnosis.

We use a similar reasoning to fill the missing values in a column (CNSOTH) that describes whether or not a subject has additional diagnoses other than PD. The vast majority of non-missing values for this column is 0 and so it is unlikely that a subject with a missing value in reality has an additional diagnosis.



The majority of subjects have a maximum number of missing values between 6 and 14.

Figure 4.4: Distributions of columns with missing values per subject

Two pairs of SCOPA columns depend on a patient's gender. One pair is applicable only for men (columns SCAU22, SCAU23) and thus missing values for women are filled with zeros. The other pair is applicable only for women (columns SCAU24, SCAU25) and thus missing values for men are filled with zeros.

Lastly, we impute 0's into missing columns of the Letter Number Sequencing (LNS) test where the last three preceding columns have also been 0's. The LNS test consists of rounds consisting of three questions. By studying the LNS data table, we were able to discern that clinicians stopped the test after a subject had completely failed a round, i.e 0's in all columns of that round. The following rounds are left blank by the clinicians and we therefore impute 0's into these columns as the subject has essentially "failed" these rounds as well.

After imputing missing values for these columns we are left with 70 rows which still contain missing values. This is a reduction from 2250 rows with missing values to 70 rows with missing values. The rest of the missing values are imputed through either mode/mean imputation or by using Heterogeneous Euclidean-Overlap Metric (HEOM) as a similarity measure for imputation. These methods for missing value imputation are presented in 3.3.1. Mode/mean imputation is our standard imputation procedure, while HEOM is explored as a potential improvement (Appendix G).

4.2.2 Dichotomizing the Response

Our response NP1HALL is at this stage a categorical variable with values ranging from 0 to 4. The different values of NP1HALL correspond to the following classifications [50]:

- 0: "Normal: No hallucinations or psychotic behavior"
- 1: "Slight: Illusions or non-formed hallucinations, but patient recognizes them without loss of insight."
- 2: "Mild: Formed hallucinations independent of environmental stimuli. No loss of insight."
- 3: "Moderate: Formed hallucinations with loss of insight."
- 4: "Severe: Patient has delusions or paranoia."

Our stated project goal is to predict whether or not a patient will develop psychotic symptoms as part of the disease progression. Our goal is not to predict the severity of these psychotic symptoms. Including the severity of psychotic symptoms in our prediction would be difficult because some of the categories of NP1HALL are very rare in the data set. The distribution of NP1HALL is shown in Figure 4.5.

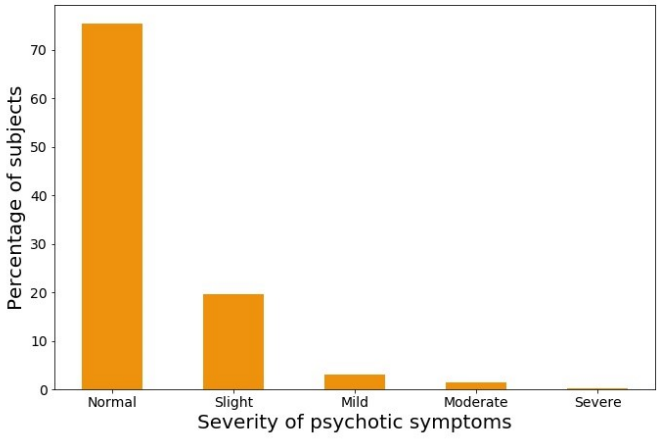


Figure 4.5: Distribution of psychotic symptoms

Around 75% of subjects in our data set never experience any psychosis, i.e all their data points have an NP1HALL of 0. Another 20% of subjects experience category 1: slight hallucinations as their worst psychotic symptoms. Finally, 3%, 1% and 0.2% of subjects experience respectively 2: slight, 3: mild and 4: moderate as their worst psychotic symptoms. No subject in our data set experiences 5: severe psychotic symptoms.

Predicting the severity of psychotic symptoms would therefore present us with a multiclass classification problem on a very imbalanced data set. We therefore dichotomize NP1HALL into a new variable, which we call HALL, defined by 4.2

$$HALL = \begin{cases} 1, & \text{if } NP1HALL > 0 \\ 0, & \text{if } NP1HALL = 0 \end{cases} \tag{4.2}$$

This leaves us with a class balance in our data set of 3:1 which is generally not considered imbalanced. According to [52], an imbalanced ratio is considered to range from 1:4 and upwards.

4.2.3 Splitting the Data

As described in 3.5.4, a common strategy for training and evaluating ML models is to split the data set into three parts: (1) training, (2) validation and (3) test set. Our data set is quite small and so we choose to omit a validation set. Our implementation of the train-test split of the data set is done in a stratified manner. This means that the class ratio of 3:1 from 4.2.2 is preserved in the new train and test subsets. For example, in the case of a 70/30 train-test split, our training subset contains 70% of all subjects, or 291 of 416 subjects in the data set. The class distribution of the response, HALL, is 220:71, which is almost equal to 3:1; the class distribution of the original data set. Our test subset contains 30% of all subjects; the remaining 125 subjects. The class distribution of the response HALL is 92:33, which is very close to the original 3:1 distribution. As is evident from the training set, it is difficult to make the distribution exactly the same sometimes, but it is similar enough to be considered preserved.

4.2.4 Features Reduction - Total Scores

We have 185 features in our data set after missing value imputation. Most of these features are component scores, e.g single answers on multiple question tests. With a large number of features (185) and small number of patients (416) we face the curse of dimensionality, i.e. the feature space is large enough to became sparse. It is important to reduce the number of features in our data set to avoid overfitting. Further, this makes the data set easier to visualize and reduces the amount of parameters that needs to be estimated by our models. One way of reducing the number of features in our data set is to combine the component scores for one test into a corresponding total score for that test. This is how the performance of a subject is most often measured by a test. In addition, Barret and Ffytche use total scores as their features. The reason why we do not collect these total scores at the beginning is that by collecting component scores instead, we give ourselves the ability to perform other forms of feature reduction such as PCA, which is detailed later in 4.2.6.

The total scores are calculated from their component scores as described in Appendix B. The total scores in Appendix B are calculated for: *Letter Number Sequencing (LNS), HVLT Immediate/Total Recall, HVLT Discrimination Recognition, HVLT Retention, MoCA Total Score, Semantic Fluency (SFT), REM Sleep Behaviour Disorder (RBD),*

GDS Raw Score, UPSIT Raw Score, Epsworth Sleepiness Scale, SCOPA-AUT Total Score, MDS-UPDRS Part III, TD/PIGD classification, Family History of PD. Some of these these total scores are further dichotomized as shown in Appendix B. This concerns the features *REM Sleep Behaviour Disorder (RBD) (RBD Positive/Negative), GDS Raw Score (Depressed/Not Depressed), Epsworth Sleepiness Scale (Sleepy/Not Sleepy), TD/PIGD classification (TD,PIGD,Indeterminate), Family History of PD (Yes/No).*

Note that this is an optional preprocessing step and can be replaced by other methods of feature reduction. After applying the aforementioned method we ended up with a reduction from 185 to 26 features, including the response. The resulting features are a mix of categorical, ordinal and numeric variables.

4.2.5 Standardizing and One-Hot Encoding

We standardize our data set and one-hot encode our categorical features after performing feature reduction by using total scores, but before we apply PCA. If we do not perform feature reduction by using total scores, we standardize and one-hot encode after splitting the data into training and test sets.

We standardize our data set according to formula 3.6 in 3.3.2. This standardization is only applied to numerical features. Ordinal and categorical features are not standardized.

In addition, we one-hot encode our categorical features as described in 3.3.2. Ordinal variables come categorically encoded from the initial PPMI database and are therefore left alone. Ordinal variables are not standarized as according to Jajuga et al. [53], standardization of ordinal variables is not necessary.

4.2.6 Features Reduction - Principal Component Analysis (PCA)

The theory behind PCA is described in 3.3.3. PCA allows us to apply a different form of feature reduction on our data set. PCA can either be applied before or after feature reduction by total scores. The number of features we have after performing PCA is dependant on how much variability is kept in the transformed features. Note that if PCA is performed, the data must be standardized and one-hot encoded beforehand as described in 4.2.5. As with the feature reduction by using total scores, this is an optional preprocessing step.

We use total scores as our standard feature reduction method. Feature reduction using PCA is explored in Appendix H.

4.2.7 Response Distribution During the Study

To better understand the three different prediction approaches detailed in 4.2.8 it is important to study the distribution of our response HALL with respect to time, where time is measured in years. Looking at these distributions also gives us an idea of the level of attrition[3] in the study. Attrition is a well-known phenomena in longitudinal studies. In this project we are not interested in the reasons for attrition, but we are still interested in the absolute numbers of subjects that drop out during the study.

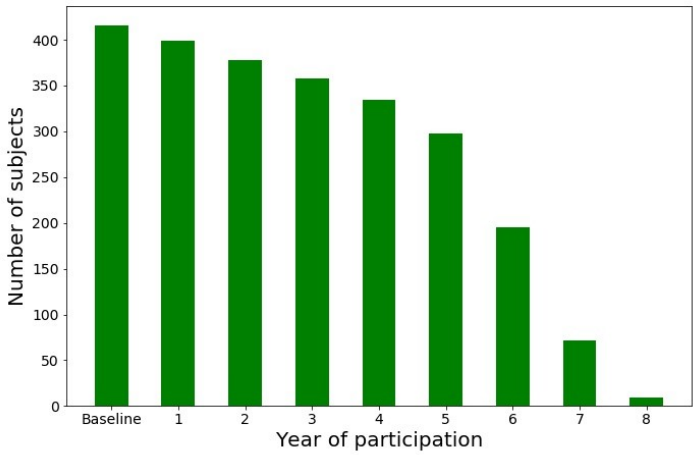


Figure 4.6: Number of subjects with regards to the length of their participation in the study

Figure 4.6 shows attrition combined with the progressive enrollment into the study. First subjects were enrolled into one of the study's cohorts in August 2010, while the last ones were enrolled in December 2018. Plot shows only subjects in our data set.

In figure 4.6, a subject that is a study participant for e.g. four years (i.e. subject's last measurement is from year 4) is counted in all of the first 5 columns (Baseline, 1, ..., 4), even if some of the intermediate yearly measurements are missing.

[3]subject leaving during a study due different reasons, e.g. death, consent withdrawal, pregnancy, protocol violation, institutionalization, primary care physician decision, etc.

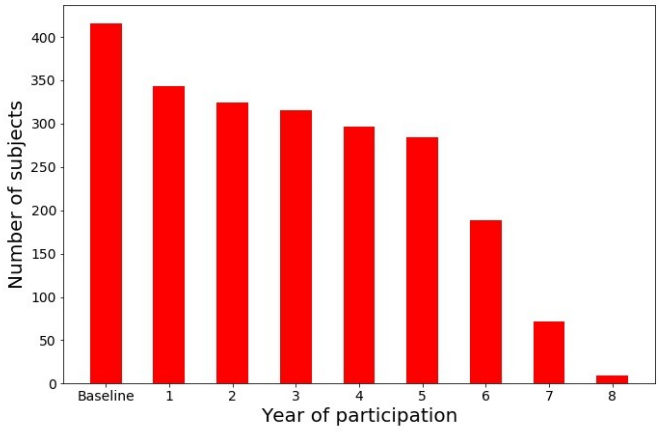


Figure 4.7: Number of subjects tested after n-years of participation

Figure 4.7 shows how many subjects had been tested at particular yearly visit. It is possible that some subjects are counted in column 3, but not in column 1 or 2. In other words: there are missing longitudinal data points for some subjects. Each column shows number of existing yearly measurement in the data set.

It is interesting to show how many participants have incomplete set of temporal data, i.e. they missed one or more of the yearly tests.

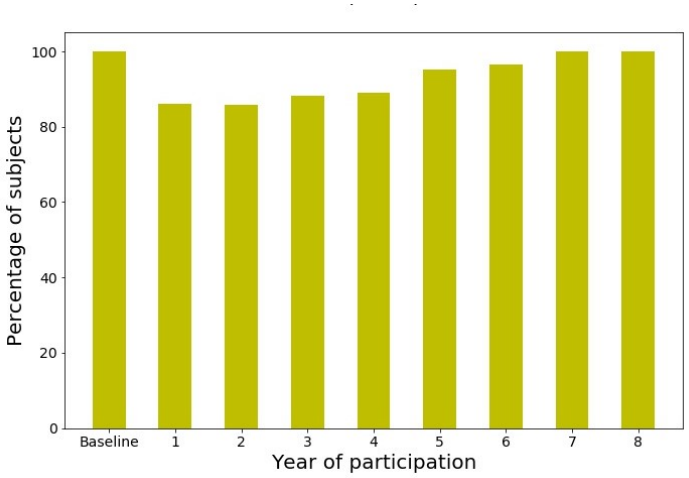


Figure 4.8: Percentage of subjects tested after n-years of participation

Figure 4.8 shows that not all participants had been tested. Evasion is most visible at years 1, 2, 3 and 4. This plot is result of yearly numbers from Figure 4.7 divided by corresponding numbers from Figure 4.6.

Patients who reported psychosis in a particular year did not necessarily keep reporting psychosis in later years.

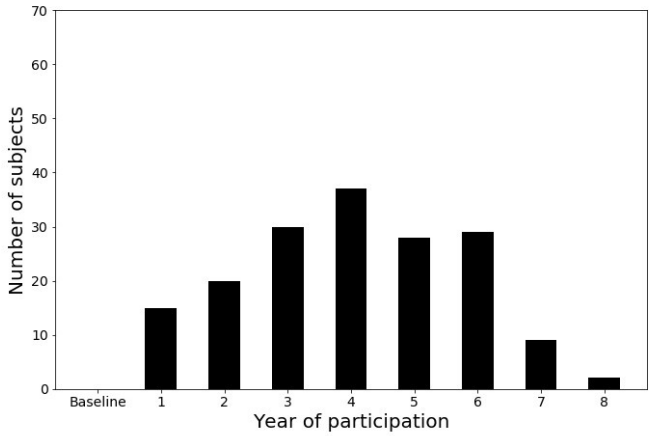


Figure 4.9: Number of subjects that reported psychosis after n-years of participation

Figure 4.9 shows a relatively low number of patients with developed psychosis. That might be because of the criterion for reporting psychosis. Namely, a patient only reports psychosis present if the patient experienced symptoms of psychosis the week preceding the report.

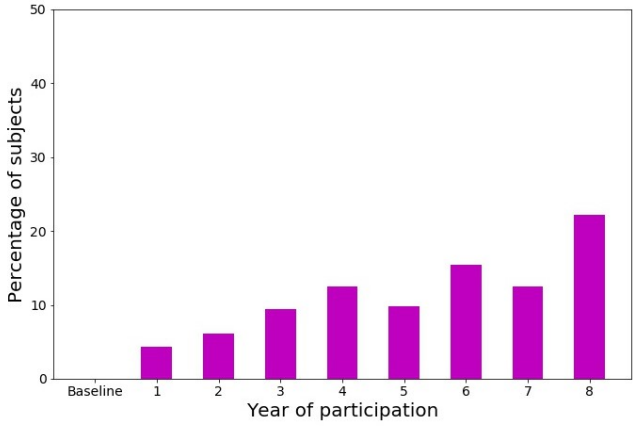


Figure 4.10: Percentage of subjects that reported psychosis after n-years of participation

Figure 4.10 shows an ascending trend in reported psychosis with years of participation, i.e. with the development of PD. This plot is the result of yearly numbers from Figure 4.9 divided by corresponding numbers from Figure 4.6.

4.2.8 Prediction Approach

Our stated project goal is to develop models that are capable of predicting psychosis in PD patients. It is important that these models are applicable in real-life scenarios. We envision two practical scenarios for our models to be applied in. In the first scenario we must try to predict psychosis for a subject which only has one data point available, i.e baseline data. In the second scenario, multiple observations are available for a patient and we must try to accommodate these additional observations into our classical ML models. By classical ML models we refer to logistic regression, support vector machine and tree based algorithms used in this project.

First Scenario - Baseline Data

This subsection details our approach to the first scenario using classical ML models. We assume that in real life, a subject will not suffer from psychosis at the baseline observation, as predicting future psychosis for a subject that already has symptoms of psychosis is of minimal interest. This is also why we remove all patients from our data set who suffer from psychosis at baseline, in one of our preprocessing steps.

For this scenario we must train our classical ML models on baseline data to predict a single response which indicates whether or not the subject will experience future psychosis. It is possible to put a constraint on this response, e.g the subject will experience psychosis within 2 years, 3 years, etc. Looking at the Figures 4.9 and 4.10 we can see that putting a constraint on the response for this data set would leave us with few subjects with a positive response, especially if the constraint is only 1 or 2 years. We therefore choose to not put such a constraint on the response and as long as the subject experiences psychosis during the lifetime of the study, the subject is predicted positive.

To process our data set for this prediction approach we first transform our response HALL into another response HALL_EVER by 4.3

$$HALL\_EVER = \begin{cases} 1, & \text{if } HALL = 1 \text{ for any longitudinal observation of a subject} \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

After transforming our response in this manner, we remove all non-baseline observations from the data set. We are then left with a baseline observation for each subject with a corresponding HALL_EVER response. Note that this processing step removes any subjects in our data set that has only baseline data available. This is because it is not possible to produce a valid response unless the subject has at least one "future" observation. In a real-life scenario, where the model is applied, not trained or validated, one observation is enough because the model would not need a "future" observation of psychotic symptoms.

Second Scenario - Derived Statistics

This subsection details our approach to the second scenario using classical ML models. In this scenario we have multiple observations available for each subject, but only one response. This is problematic because classical ML models depend on a 1:1 ratio between observations and responses. This means that in order to apply these models on the

second scenario, we will need to condense the information of multiple observations into one observation. A way of doing this is to derive statistics from the multiple observations that can be summarized into one single observation.

To develop models that can benefit from information carried by multiple longitudinal observations we make the assumption that in real life, applying models to a patient which has already had psychosis makes little sense. In other words, using observations where the subject has a positive HALL response to train the model does not reflect reality. Therefore, we derive statistics based on the first $i-1$ temporal observations of a subject, where i is either the first temporal observation with HALL $= 1$ or the last observation of a subject (in the case where a subject never experiences psychosis). This idea leads to a model that is never trained on any observation where HALL is positive, as is illustrated in Figure 4.11.

Patient 1

	EVENT_ID	HALL
18	BL	0
19	V04	0
21	V06	0
23	V08	0
20	V10	0
22	V12	0
24	V13	0
25	V14	0
26	V15	0

Patient 2

	EVENT_ID	HALL
41	BL	0
40	V04	0
43	V06	1
44	V08	1
42	V10	1
45	V12	1
46	V13	0
47	V14	1
48	V15	0

Patient 3

	EVENT_ID	HALL
28	BL	0
27	V04	0
30	V06	0
29	V08	0
31	V12	0

Figure 4.11: Different i's, number of observations taken into consideration

Patients 1, 2 and 3 illustrate how this prediction approach works in practice. As a reminder, EVENT_ID is our longitudinal variable and is sorted.

- Patient 1 has no psychosis during the study and so we derive statistic based on all but the last observation he/she has in the study ($i = 9$). The last (red) observation contains the response HALL $= 0$.
- Patient 2 has psychosis relatively early and we derive statistic based on the first 2 observations ($i = 3$). The 3rd (red) observation contains the response HALL $= 1$.

- Patient 3 has no psychosis and has missing observations (V10). We derive statistic based on his/hers available observations, with the exception of the last available observation ($i = 5$). The last (red) observation contains the response HALL = 0.

Our response is the value of HALL at observation i, where $i \geq 3$, so that we have at least two observations to derive statistics from and one observation to draw a response from. In a real-life scenario, we would need a minimum of two observations to predict the presence of psychosis because the model is applied and would no longer need to be trained or validated. Note that this approach removes, from the data set, all subjects that have less than 2 observations available before observation i.

Our observations are a mix of categorical, ordinal and numerical features. For numerical and ordinal features we derive the following statistics from the first $i - 1$ temporal observations of a subject:

- Average
- Intercept and slope of regression line
- Minimum
- Maximum

According to [27], these are commonly used in longitudinal analysis. These statistics are derived from ordinal features with the underlying assumption that the distance between successive values of an ordinal features is constant. None of these statistics can be derived from categorical features. The only statistics we derive from categorical features is therefore the mode.

Deep Learning: LSTM

Classical ML models are not conceived for establishing true longitudinal relationship between observations. In our case, the best we can do is to summarize these relationships into derived statistics and apply classical ML models as in 4.2.8. Certain deep learning models on the other hand are conceived specifically to extract temporal relationships. Recurrent Neural Networks (RNNs) and Long-Short Term Memory (LSTM) are examples of such deep learning models and are presented in 3.4.5. Using such models means that we no longer have to adhere to the 1:1 ratio between observations and responses, as in our classical ML models. These models are capable of predicting one response for multiple longitudinal observations.

To process our data set for these DL models we find i as in 4.2.8. The first $i-1$ observations are processed by the model and the value of HALL in i is used as the response. Unlike when we use derived statistics, data from a subject can be used as long as it has more than 1 observation. This makes such deep learning models capable of operating on not only longitudinal data, but also on just baseline data. Subjects that only have a baseline observation available are removed from the data set because they lack a "future" observation of psychotic symptoms.

4.3 Models

Every model presented in this section will make use of some variant of 'balanced' class weights to counteract our 3:1 class distribution. This means that the minority class, i.e. subjects with psychosis, are given more weight in the model compared to subjects of the majority class. Different models use different methods to give more weights to some samples, but the final result is the same. The model is penalized more heavily for misclassifying a minority sample than a majority one. In the case of our 3:1 class distribution, a model would be penalized 3 times as much for misclassifying a subject with psychosis than a subject without psychosis. In sklearn, 'balanced' class weights means that the model uses the values of the response to automatically adjusts weights inversely proportional to class frequencies in the input data [54].

4.3.1 Classical Machine Learning Models

Logistic Regression

LR is explained in 3.4.1 and implemented using the scikit-learn python package [55] with the class *LogisticRegression* [56]. For LR we are interested in exploring the effects of regularization on the model as explained in 3.1.2.

We will use the 'liblinear' scikit-learn solver [56] as it supports both L1 and L2 norm for regularization penalty. For our logistic regression models we want to tune these hyper-parameters:

- Regularization penalty: possible values are [L1-Norm, L2-Norm]
- Regularization strength: range $[0, \infty]$.

We use 'balanced' class weights in order to give our models better performance on the minority class.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is implemented using the scikit-learn python package [55] with the class *Svm* [57]. We will try three different kernels that are described in 3.4.2. These are linear, polynomial and radial kernels. In addition, we are interested in the effects of regularization. Scikit-learn does not support anything other than L2-norm regularization penalty for SVM. We use 'balanced' class weights in order to give our models better performance on the minority class.

The linear kernel is defined in the scikit-learn implementation as described in 3.4.2. The parameter to tune is therefore the strength of regularization.

We want to tune the following hyper-parameters for a linear kernel:

- Regularization strength: possible values are [0, ∞]

The polynomial kernel in the scikit-learn implementation is slightly different from the one detailed in 3.4.2. Instead of the definition used of some authors (e.g. Tibshirani [22], Bishop [12])

$$K(x, x_i) = (1 + \sum_{j}^{p} x_j x_{ij})^d, \tag{4.4}$$

the polynomial kernel in scikit-learn, [58], is defined more generally as

$$K(x, x_i) = (r + \gamma \sum_{j}^{p} x_j x_{ij})^d, \tag{4.5}$$

where r is a constant that allows to trade off the influence of the higher order and lower order terms.

For the polynomial kernel we therefore want to tune following hyper-parameters:

- Regularization strength: range [0, ∞]
- Gamma scaling, γ: range (0, ∞]
- Polynomial degree, d: range [1, ∞]
- Independent term, r: range [0, ∞]

The radial kernel is defined in the scikit-learn implementation as described in 3.4.2. The parameter to tune is therefore the gamma scaling and the strength of regularization.

For the radial kernel we therefore want to tune the following hyper-parameters:

- Regularization strength: range $[0, \infty]$
- Gamma scaling, γ: range $(0, \infty]$

Decision Tree

Our decision tree model is implemented through the scikit-learn python package [55] with the class *DecisionTreeClassifier* [59]. We will build our decision trees by using cost complexity pruning as explained in 3.4.3. We will choose to minimize GINI in our splits as it is the standard criterion for decision trees in scikit-learn. Using GINI over entropy or vice versa usually does not affect results too a large degree.

We want to tune the following hyper-parameters for decision trees:

- Complexity parameter, α: range $[0, \infty]$

For decision trees we can also perform some inference by visualizing the tree structure of the tuned model.

We use 'balanced' class weights in order to give our models better performance on the minority class.

Random Forest

Our random forest model is implemented through the scikit-learn python package [55] with the class *RandomForestClassifier* [60]. We will build our random forest models using the theory in 3.4.3. The hyper-parameters we want to tune are the number of trees in a forest and the fraction of features considered at each split. We use 'balanced' class weights in order to give our models better performance on the minority class.

The theory in 3.4.3 mentions that trees in random forest are usually not pruned and often fully grown. After preliminary testing it was evident that our random forest models required some kind of tree regularization, due to overfitting when just tuning the two initial hyper-parameters. It was therefore decided to introduce another hyper-parameter

that controls the max depth of individual trees to prevent overfitting. We use this hyper-parameter instead of cost complexity pruning to reduce the complexity of individual trees.

We therefore want to tune the following hyper-parameters for random forest:

- Number of trees in the forest, n: range $[1, \infty]$
- Fraction of features considered at each split, m: range $[0, 1]$, if $m = 0$ then minimum of 1 feature considered at each split.
- Maximum depth of trees in the forest, d: range $[1, \infty]$

For random forest we can also estimate generalization error with Out-Of-Bag error (OOB-error) as explained in 3.4.3. We'll present OOB in addition to nested cross validation for random forest. We can perform some interference from random forest by plotting a feature importance summary as explained in 3.4.3.

Boosted Trees

We implement a boosted trees model through the XGBoost package for python [61]. A brief overview of boosted trees can be found in 3.4.4. The boosted trees model in XGBoost has a lot of hyper-parameters available, which makes tuning complex. We will therefore base our tuning on the XGBoost document *"Notes on Parameter Tuning"* [62]. The effect and range of the different hyper-parameters in XGBoost is described in another XGBoost document, *XGBoost Parameters* [63]. We will start tuning relatively simply by just tuning the number of boosting rounds and leaving the rest of the hyper-parameters on default values. We will also scale the weights of the positive class so that the class weights are balanced.

Preliminary testing shows that the resulting models are overfitting the training data. According to *"Notes on Parameter Tuning"* [62], overfitting in XGBoost can be controlled by introducing these hyper-parameters:

Hyper-parameters that directly reduces the complexity of a model:

- max_depth: Stops each boosted tree from growing beyond a maximum depth. Default value: 6
- min_child_weight: Stops each boosted tree from performing splits if this results in child nodes where the sum of sample weights is less than this threshold. Default value: 1

- gamma: Stops each boosted tree from performing a split if the loss reduction is below this threshold. Default value: 0

Hyper-parameters that introduce randomness. This makes the model more robust to noise [62]:

- subsample: Each boosted tree is built on a subsample of all the observations in the training set. Default value: 1.
- colsample_bytree: Each boosted tree is only allowed to use a subsample of all features to split on. Default value: 1
- eta: Shrinks the new feature weights on each boosting round. Similar to learning rate in other boosting algorithms. Default value: 0.3

We will try to tune all these hyper-parameters in addition to the number of boosting rounds. These are just some of the hyper-parameters available in XGBoost and we could, for example, also tune the regularization parameter for both L1-norm and L2-norm. We will focus on the listed hyper-parameters because they are highlighted in the *"Notes on Parameter Tuning"* [62]. Preliminary testing shows that they are able to control overfitting.

We want to tune the following hyper-parameters for boosted trees:

- number of boosting rounds, n: range $[1,\infty]$
- max_depth, d: range $[1,\infty]$
- min_child_weight, w: range $[0,\infty]$
- gamma, γ: range $[0,\infty]$
- subsample, $s1$: range $(0,1]$
- colsample_bytree, $s2$: range$(0,1]$
- eta, η: range$[0,1]$

We can perform some interference from boosted trees by plotting a feature importance summary as explained in 3.4.3.

4.3.2 Deep Learning: LSTM

We implement one deep learning model for this project. The model we chose is long-short term memory (LSTM), a variant of RNN. RNN and LSTM networks are explained in 3.4.5. LSTM networks are well suited for longitudinal data/time series and are usually preferred over RNN networks. The LSTM network is implemented using the Keras API for Python [64].

Validation Set

Before we implement the model itself we have to do some additional preprocessing specific for deep learning. The classical ML models in this project rely on cross validation in order to tune hyper-parameters. Cross validation is not commonly used for deep learning and does not have native support in Keras. We will therefore opt to use a validation set for our LSTM model.

We start with a training and test set. In order to create an additional validation set, we must split the training set. We want both the test set and validation set to be of equal size. The correct split ratio for the validation set is therefore calculated as

$$n_{test}/n_{train}$$

where n_{test} is the number of samples in the test set and n_{train} is the number of samples in the training set.

For example, if we consider a data set with a 100 samples and we first perform a 80/20 split, $n_{test}/n_{train} = 80/20 = 0.25$, splitting the training set with this ratio results in $0.75 * 80 = 60$ and $0.25 * 80 = 20$ samples for the training and validation set. The resulting training, test and validation sets are respectively 60/20/20.

Creating Batches

When a neural network is trained, it usually does not update its weights for each prediction it makes. Instead a collection of training observations are processed in what is called a batch and weights are updated on the basis of all the predictions in such a batch. A neural network is usually trained on multiple such batches. Batches are used because they save computational effort and memory. They can also help give more stable estimates of the gradient, according to [65].

A limitation of batches in RNN or variants of RNN, (such as LSTM), is that they require all time series in a batch to have the same amount of observations. Due to the way we process the data for deep learning (see 4.2.8), each subject, (which corresponds to a time series), has a variable amount of observations. This makes it hard to create large batches with a balanced number of subjects. We therefore chose to use mini-batches of one subject per batch, which is called stochastic gradient descent, according to [65]. This means that our LSTM network is fed one subject at a time and will update its weights after each subject.

Number of Epochs

A neural network is usually allowed to train on a single data set multiple times in order to find the right weights. The number of times a neural network is allowed to train on a single data set is usually referred to as epochs. The number of epochs can be a predefined number.

Another way of determining the number of epochs is to use early stopping. If a chosen performance measure stops improving on a validation set after a certain number of epochs, we can assume that the model will not improve by additional epochs. We can therefore stop training the model at this point. This is called early stopping.

The last model before training is halted due to early stopping might not be the best performing model of all models trained by epochs. This is due to how sometimes a gradient function can overshoot a local minimum, according to [66]. It is therefore common to combine early stopping with continually saving the best performing model after each epoch. When early stopping comes into effect, the best performing model from all the epochs trained is returned. This is called a model checkpoint in Keras.

We implement early stopping and a model checkpoint for our LSTM network. We do early stopping and select the best model checkpoint by looking at sensitivity as our performance measure.

Tuning the LSTM Network

It is difficult to comprehensively tune all hyper-parameters in a neural network. Tuning is often done heuristically or based on previous working examples. There is no foolproof recipe for tuning because tuning is largely dependant on the problem and data [67]. We have not found relevant work which we can build upon when it comes to LSTM networks. Therefore we will adapt a strategy of building a simple LSTM network and then gradually trying to add complexity.

We start with a simple LSTM network with one input layer, one LSTM layer and one binary output layer. The input layer is automatically handled by Keras. The output layer uses "sigmoid" as the final activation function and the LSTM layer uses the default activation function of "tanh".

We will first tune the number of neurons in the LSTM layer. After determining the number of neurons to use, we will try different optimizers and learning rates. This should result in a relatively simple tuned LSTM network. We can then try adding additional layers to the network. One option is adding a dropout layer. This is relevant if we see that our model struggles with overfitting. Another option is to add more LSTM layers to the network as according to Goldberg, Yoav [68]: *"While it is not theoretically clear what is the additional power gained by the deeper architecture, it was observed empirically that deep RNNs work better than shallower ones on some tasks"*.

It is important to mention that, as in all the other models in this project, we will use 'balanced' class weights to train our LSTM network.

4.4 Model Tuning and Validation Strategy

4.4.1 Performance Measures

We will measure the performance of our models using the performance metrics presented in 3.5.1. All metrics are implemented using the scikit-learn python package [55], module *sklearn.metric* [69]. We pay the most attention to sensitivity as a performance measure due to the recommendation by an MD working on Parkinson's disease.

4.4.2 Model Tuning and Validation

Machine Learning Models

We tune our ML models using a combination of random search and grid search which are explained in 3.5.2. In general, we use random search in combination with cross validation to perform a first exploratory search through the hyper-parameters of a model. We then use the results of this search to narrow down the range of the hyper-parameters for a grid search. This grid search is accompanied by nested cross validation in order to better estimate the generalization error of a model. Finally, we estimate the performance of the tuned model on a test set.

Cross validation and nested cross validation are explained in 3.5.3. We generally use 10 folds for both nested and regular cross validation. Random search, grid search and cross validation are implemented through the scikit-learn python package [55], module *sklearn.model_selection* [70]. Our models are tuned to provide the best performance on sensitivity as this is considered our most important metric due to recommendation by an MD working on Parkinson's disease.

Deep Learning Models

As mentioned in 4.3.2 we make use of a validation set for the LSTM network. Hyper-parameters are selected by the aggregated result of multiple grid searches on this validation set. We choose the hyper-parameter that gives the best average result on multiple grid searches due to the randomness inherent in a neural network. The initial weights set in a neural network is just one example of a random element that has a large effect on a final model. We use the average result of 5 grid searches for tuning. On the same note, we will present our results from deep learning as the average performance of 10 neural networks on the test set; all models built with the same hyper-parameters.

Chapter 5

Results and Discussion

In this chapter we present our results from the three different prediction approaches used in this thesis. We show our tuning process for each model and then show that model's performance on various metrics. When applicable, we also show inference for a model. ROC curves and confusion matrices for all models are available in Appendix E. Finally, we analyze and discuss our results.

5.1 Results

5.1.1 Baseline Approach

This section presents the results from classical ML models trained on the "baseline approach" as detailed in 4.2.8. The following preprocessing steps are applied on the data set for this approach:

- Mode/mean missing value imputation
- Feature reduction through total scores
- Train/test set ratio of 70/30
- Prediction based on baseline data

The processed data set contains 399 unique subjects with 279 subjects in the train set and 120 subjects in the test set. This is a reduction from the 416 unique subjects in the data set due to the removal of subjects that do not have at least one additional observation other than the baseline observation for this approach. Class proportion is

approximately 3:1 in both the training and test sets in favor of the negative class, i.e. the class of PD patients that did not experience psychotic symptoms over the course of the study. We use 10 folds for both cross validation and nested cross validation. Models are tuned to give the best sensitivity possible.

Confusion matrices and ROC curves for the models presented in this section can be found in Appendix E - *Baseline Approach*.

Logistic Regression

As stated in 4.3.1, the hyper-parameters we want to tune for this model are:

- Regularization penalty, L: possible values are [L1-norm, L2-norm]
- Regularization strength, C: range $[0, \infty]$

Our scikit-learn solver is 'liblinear' and we use 'balanced' class weights. Note that a lower value of C gives a stronger regularization.

The tuning phases used to find the final tuned model are summarized in Table 5.1.

Tuning Phase	Random Search 1	Random Search 2	Grid search 1	Grid search 2
Parameters	C,L	C,L	C,L	C,L
Range/Grid C	[0,10]	[0,10]	0.35,0.01	0.0001,0.001, 0.01,01
Range/Grid L	L1	L2	L1,L2	L2
Chosen C	0.35	0.01	0.01	0.01
Chosen L	L1	L2	L2	L2

Table 5.1: Baseline Approach: Logistic Regression Tuning.

We first perform a random search with 1000 iterations to find an estimate of the optimal regularization strength for L1-norm regularization penalty. Results indicate a regularization strength of around 0.35 for L1-norm.

Secondly, we perform a random search with 1000 iterations to find an estimate of the optimal regularization strength for L2-norm regularization penalty. Results indicate a regularization strength of around 0.01 for L2-norm.

Using the results from the first two searches we perform a grid search with possible values for regularization penalty being L1 or L2 and regularization strength being 0.35 or 0.01. Results indicate that L2-penalty performs better than L1. A last grid search determines the optimal strength of regularization.

The chosen hyper-parameters are L2-norm for regularization penalty and 0.01 for regularization strength. The results from nested cross validation, training and test set are summarized in Table 5.2.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.620	0.350	0.512	0.657	0.411	0.585	0.157
Nested CV - std	0.136	0.176	0.235	0.120	0.190	0.160	0.297
Training results	0.695	0.431	0.620	0.721	0.509	0.670	0.308
Test results	0.658	0.407	0.710	0.640	0.518	0.675	0.308

Table 5.2: Baseline Approach: Logistic Regression Results

Support Vector Machine (SVM)

As stated in 4.3.1 we want to try different kernels for our SVM models.

For a linear kernel we want to tune the following hyper-parameter:

- Regularization strength, C: range $[0, \infty]$

We use 'balanced' class weights. Note that a lower value of C gives a stronger regularization.

The tuning phases used to find the final tuned model for a linear kernel are summarized in Table 5.3.

Tuning Phase	Random Search 1	Grid search 1
Parameters	C	C
Range/Grid C	[0,10]	[1,12], SS = 0.1
Chosen C	6.46	3.3

Table 5.3: Baseline Approach: SVM Linear Kernel Tuning. SS = Step Size

We start by performing a random search with 1000 iterations to find an estimate for the optimal regularization strength. Results indicate a regularization strength of around 6.46 for SVM with a linear kernel.

Using this result we perform a final grid search. The chosen regularization strength is 3.3.

The results from nested cross validation, training and test set are summarized in Table 5.4.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.627	0.354	0.562	0.649	0.430	0.606	0.190
Nested CV - std	0.068	0.069	0.164	0.098	0.091	0.077	0.135
Training results	0.724	0.470	0.662	0.745	0.550	0.704	0.370
Test results	0.683	0.422	0.613	0.708	0.500	0.660	0.290

Table 5.4: Baseline Approach: SVM Linear Kernel Results

For a polynomial kernel we want to tune following hyper-parameters:

- Regularization strength, C: range $[0, \infty]$
- Gamma scaling γ: range $(0, \infty]$
- Polynomial degree, d: range $[1, \infty]$
- Independent term, r: range $[0, \infty]$

The tuning phases used to find the final tuned model for a polynomial kernel are summarized in Table 5.5.

Tuning Phase	Random Search 1	Grid search 1	Grid search 2	Grid search final
Parameters	C,γ,d,r	C,γ,d,r	C,γ,d,r	C, γ,d,r
Range/Grid C	[0,10]	[0.14,0.18], SS = 0.01	[0.1,0.14], SS = 0.01	[0.01,0.08], SS = 0.01
Range/Grid γ	(0,10]	[0.2,0.4], SS = 0.05	[0.4,0.6], SS = 0.05	[0.50,0.75], SS = 0.05
Range/Grid d	[1,5]	[1.4,1.8], SS = 0.1	[1.0,1.4], SS = 0.1	[1.0,1.4], SS = 0.1
Range/grid r	[0,10]	[4.0,4.8], SS = 0.2	[3.2,4.0], SS = 0.2	[0.0,0.8], SS = 0.2
Chosen C	0.16	0.14	0.1	0.02
Chosen γ	0.3	0.4	0.55	0.70
Chosen d	1.6	1.4	1.0	1.0
Chosen r	4.4	4.0	3.2	0.0

Table 5.5: Baseline Approach: SVM Polynomial Kernel Tuning.
SS = Step Size

We start by performing a random search with 1000 iterations to find estimates of the optimal values for regularization strength, gamma scalling, polynomial degree and independent term. Results indicate a regularization strength of around 0.16, gamma scaling of around 0.3, degree of around 1.6 and independent term of around 4.4.

Using these results we perform a grid search. The chosen hyper-parameters from this grid search are as follows:

- Regularization strength: 0.14
- Gamma scaling: 0.4
- Polynomial degree: 1.4
- Independent term: 4.0

The chosen hyper-parameters are all at the minimum or maximum allowed by the grid search. This suggests that we should change the range of our search. We therefore perform a second grid search with a different grid. The chosen hyper-parameters from this second grid search are as follows:

- Regularization strength: 0.1
- Gamma scaling: 0.55
- Polynomial degree: 1.0
- Independent term: 3.2

The chosen hyper-parameters from this second grid search are also at minimum or maximum values allowed by the grid, with the exception of gamma scaling. Polynomial degree is at the minimum allowed for the parameter. We need to find better ranges for the regularization strength and independent term. We therefore perform a series of grid searches until we find the right ranges. The chosen hyper-parameters from the final grid search are:

- Regularization strength: 0.02
- Gamma scaling: 0.70
- Polynomial degree: 1.0
- Independent term: 0.0

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.652	0.371	0.480	0.711	0.410	0.596	0.181
Nested CV - std	0.077	0.118	0.174	0.084	0.117	0.096	0.179
Training results	0.728	0.470	0.549	0.788	0.506	0.669	0.322
Test results	0.683	0.419	0.581	0.719	0.486	0.650	0.274

Table 5.6: Baseline Approach: SVM Polynomial Kernel Results

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table 5.6.

For a radial kernel we want to tune the following hyper-parameters:

- Regularization strength, C: range $[0, \infty]$
- Gamma scaling, γ: range $(0, \infty]$

The tuning phases used to find the final tuned model for a radial kernel are summarized in Table 5.7.

Tuning Phase	Random Search 1	Random Search 2	Grid search 1
Parameters	C,γ	C,γ	C,γ
Range/Grid C	[0,10]	[5,15]	[6,12], SS = 1
Range/Grid γ	[0,10]	[0,1]	[0.001,0.01], SS = 0.001
Chosen C	0.01	9	7
Chosen γ	9.1	0.002	0.004

Table 5.7: Baseline Approach: SVM Radial Kernel Tuning.
SS = Step Size

We start by performing a random search with 1000 iterations to find estimates of the optimal values for regularization strength and gamma scaling. Results indicate a regularization strength of 0.01 and gamma scaling of 9.1.

Results from cross validation show that all samples are being classified as positive with the chosen hyper-parameters. This indicates a biased model which does not capture any of the complexity of the data. As explained in this document from the scikit-learn developers [71], a model based on a radial kernel might be very biased if gamma scaling is too high or if regularization is too strong.

We perform a second random search. This time we increase the range for regularization strength and lower the range for gamma scaling. Results no longer appear highly biased. The results from this random search indicate a regularization strength of around 9 and gamma scaling of around 0.002.

Based on these results we perform a final grid search to find the optimal hyper-parameters. The chosen hyper-parameters are a regularization strength of 7.0 and a gamma scaling of 0.004.

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table 5.8.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.638	0.360	0.480	0.692	0.400	0.586	0.165
Nested CV - std	0.073	0.102	0.162	0.106	0.089	0.075	0.144
Training results	0.735	0.483	0.592	0.784	0.532	0.688	0.353
Test results	0.658	0.375	0.484	0.719	0.423	0.601	0.188

Table 5.8: Baseline Approach: SVM Radial Kernel Results

Decision Tree

As stated in 4.3.1, the hyper-parameter we want to tune for this model is:

- Complexity parameter, α: range $[0, \infty]$

We use 'balanced' class weights.

The tuning phases used to find the final tuned model for a decision tree are summarized in Table 5.9.

Tuning Phase	Random Search 1	Random Search 2	Grid search 1
Parameters	α	α	α
Range/Grid α	[0,0.1]	[0,0.01]	[0,001.02], SS = 0.001
Chosen α	0.06	0.008	0.016

Table 5.9: Baseline Approach: Decision Tree Tuning.
SS = Step Size

We start by performing a random search with 1000 iterations to find an estimate of the optimal value of α. The chosen value for α from this search is around 0.06. Results show that all samples are classified as positive with the chosen hyper-parameter. This indicates a biased model as it did for the radial SVM kernel in 5.1.1. To decrease bias we must increase the complexity of our model. We can do this by lowering the range of α.

We therefore perform a second random search. This time we use a lower range for α. The chosen value for α from this search is around 0.008. Results no longer indicate a very high bias. Using this results we perform a final grid search to find an optimal value. The optimal value for α is found to be 0.016

The results from nested cross validation, training and test set for this hyper-parameter are summarized in Table 5.10.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.451	0.223	0.439	0.457	0.289	0.448	-0.098
Nested CV - std	0.132	0.078	0.166	0.178	0.098	0.111	0.202
Training results	0.699	0.451	0.845	0.649	0.588	0.747	0.431
Test results	0.600	0.356	0.677	0.573	0.467	0.625	0.219

Table 5.10: Baseline Approach: Decision Tree Results

We can produce a visualization of this decision tree. We limit this visualization to a tree structure of maximum 3 nodes in depth to limit the size of the corresponding image. The visualization can be viewed in Figure 5.1.

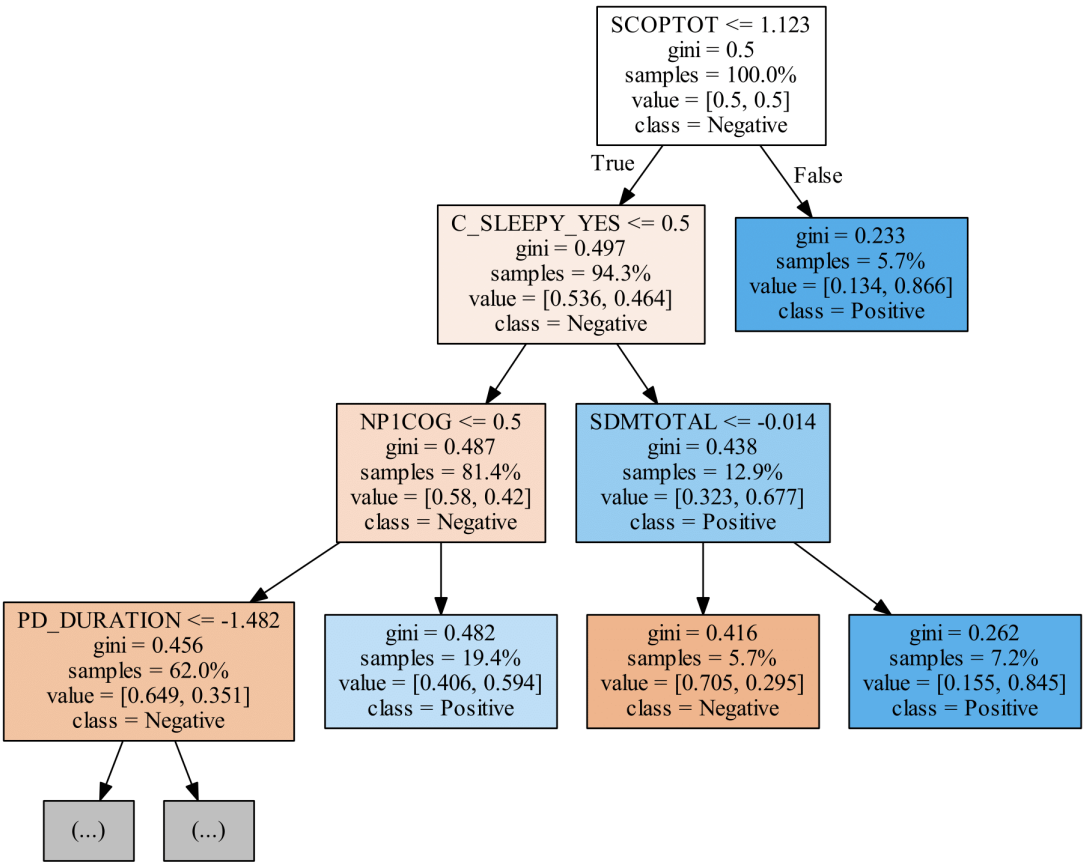


Figure 5.1: Visualization of the decision tree structure for the baseline approach, max tree depth of 3

Random Forest

As stated in 4.3.1, the hyper-parameters we want to tune for this model are:

- Number of trees in the forest, n: range $[1, \infty]$
- Fraction of features considered at each split, m: range $[0, 1]$, if $m = 0$, then a minimum of 1 feature is considered at each split
- Maximum depth of trees in the forest, d: range $[1, \infty]$

We use 'balanced' class weights.

The tuning phases used to find the final tuned model for random forest are summarized in Table 5.11.

Tuning Phase	Random Search 1	Grid search 1
Parameters	n,m,d	n,m,d
Range/Grid n	[10,100]	[18,30], SS = 2
Range/Grid m	[0.5,0.9]	[0.55,0.75], SS = 0.05
Range/Grid d	[1,20]	[1,4], SS = 1
Chosen n	24	22
Chosen m	0.65	0.70
Chosen d	1	1

Table 5.11: Baseline Approach: Random Forest Tuning.
SS = Step Size

We start by performing a random search with 1000 iterations to find estimates for the optimal values of n, m and d. The chosen values are $n = 24$, $m = 0.65$ and $d = 1$. Using these results we perform a grid search. The chosen hyper-parameters are $n = 22$, $m = 0.70$ and $d = 1$.

The results from nested cross validation, OOB, training and test set for this set of hyper-parameters are summarized in Table 5.12.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.612	0.298	0.355	0.701	0.314	0.528	0.057
Nested CV - std	0.102	0.190	0.196	0.111	0.171	0.117	0.239
OOB	0.599	0.275	0.352	0.683	0.309	0.517	0.032
Training results	0.692	0.416	0.521	0.750	0.462	0.636	0.253
Test results	0.725	0.471	0.516	0.798	0.492	0.657	0.305

Table 5.12: Baseline Approach: Random Forest Results

We can plot the importance of the features used in the random forest model. The higher value a feature has, the more important it is for the model. The set of feature importance values sum to 1. The plot can be viewed in Figure 5.2.

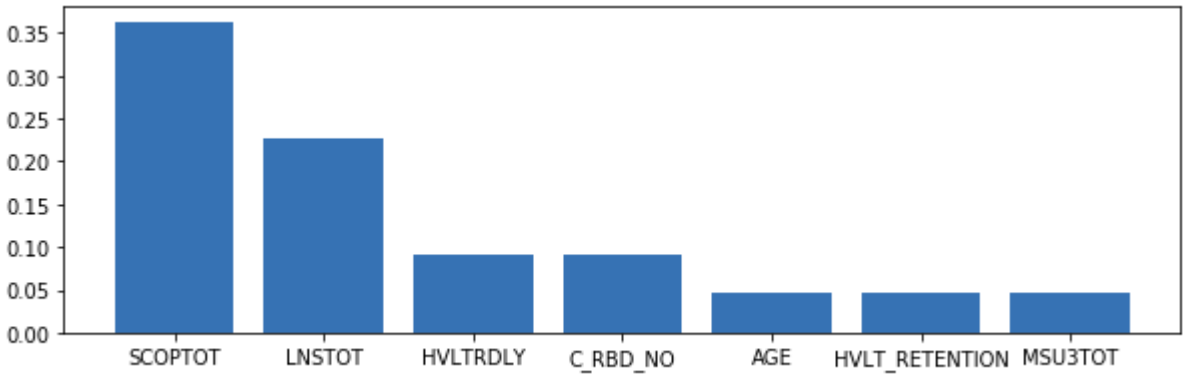


Figure 5.2: Feature importance values plotted as a bar plot. Only the most important features are shown.

Boosted Trees

As stated in 4.3.1, the hyper-parameters we want to tune for this model are:

- number of boosting rounds, n: range $[1, \infty]$
- max_depth, d: range $[1, \infty]$
- min_child_weight, w: range $[0, \infty]$
- gamma, γ: range $[0, \infty]$
- subsample, $s1$: range $(0, 1]$
- colsample_bytree, $s2$: range $(0, 1]$
- eta, η: range $[0, 1]$

The tuning phases used to find the final tuned model for boosted trees are summarized in Table 5.13.

We start by performing a random search with 1000 iterations to find estimates for the optimal values of the hyper-parameters. Results indicate these values:

- number of boosting rounds, n: 10
- max_depth, d: 2
- min_child_weight, w: 2.5
- gamma, γ: 2.4

Tuning Phase	Random search 1	Grid search 1	Grid search 2	Random search final
Parameters	n,d,w,$s1$,$s2$,η	n,d,w,$s1$,$s2$,η	n,d,w,$s1$,$s2$,η	n,d,w,$s1$,$s2$,η
Range/Grid n	[5,20]	[9,11], SS = 1	[7,10], SS = 1	[5,15]
Range/Grid d	[1,10]	[1,3], SS = 1	[1,2], SS = 1	[0,3]
Range/Grid w	[1,10]	[2,3], SS = 1	[1,2], SS = 1	[0,2]
Range/Grid γ	[0,5]	[2,3], SS = 1	[1,2], SS = 1	[0.1,2.0]
Range/Grid $s1$	[0.5,1]	[0.4,0.6], SS = 0.1	[0.4,0.6], SS = 0.1	[0.4,1.0]
Range/Grid $s2$	[0.5,1]	[0.9,1], SS = 0.1	[0.9,1], SS = 0.1	[0.5,1.0]
Range/Grid η	[0.3,1]	[0.8,1], SS = 0.1	[0.7,1], SS = 0.1	[0.3,2.0]
Chosen n	10	9	8	12
Chosen d	2	1	1	2
Chosen w	2.5	2	1	1.23
Chosen γ	2.4	2	1	1.44
Chosen $s1$	0.5	0.5	0.6	0.42
Chosen $s2$	1	1	1	0.80
Chosen η	0.9	0.8	0.9	1.76

Table 5.13: Baseline Approach: Boosted Trees Tuning.
SS = Step Size

- subsample, $s1$: 0.5
- colsample_bytree, $s2$: 1
- eta, η: 0.9

Using these results we perform a series of grid searches. After a number of grid searches it became apparent that we would not get a stable set of hyper-parameters using grid search. This is most likely due to the low amount of values we can try for each parameter for each grid search. For example, grid search 1 and 2 uses only two or three different values for each parameter in order to keep the number of permutations computationally feasible. Even if we only increase the number of values for each parameter from three to four, we increase the number of permutations from $3^7 = 2187$ to $4^7 = 16384$. We therefore decided to estimate the optimal hyper-parameters using a random search with 10000 iterations and use the results to build our model. The chosen hyper-parameters from this final random search are as follows:

- number of boosting rounds, n: 12

- max_depth, d: 2
- min_child_weight, w: 1.23
- gamma, γ: 1.44
- subsample, $s1$: 0.42
- colsample_bytree, $s2$: 0.80
- eta, η: 1.76

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table 5.14.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.533	0.255	0.527	0.538	0.319	0.532	0.058
Nested CV - std	0.156	0.114	0.328	0.297	0.159	0.083	0.151
Training results	0.620	0.358	0.620	0.620	0.454	0.620	0.210
Test results	0.667	0.395	0.548	0.708	0.459	0.628	0.234

Table 5.14: Baseline Approach: Boosted Trees Results

We can plot the feature importance of the features used in the boosted trees model. The feature importance is measured in the total information gain for each split using this feature in the model. The higher value a feature has, the more important it is for the model. The plot can be viewed in Figure 5.3.

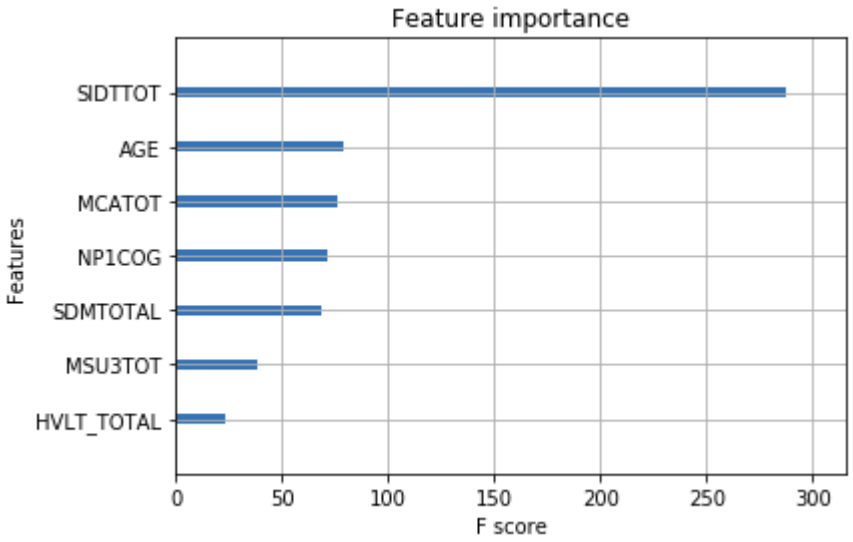


Figure 5.3: Plot of feature importance for boosted trees, only the most important features are shown

5.1.2 Derived Statistics Approach

This section presents the results from classical ML models trained on the "derived statistics approach" as detailed in 4.2.8. The following preproccesing steps are applied on the data set for this approach:

- Mode/mean missing value imputation
- Feature reduction through total scores
- Train/test set ratio of 70/30
- Prediction based on derived statistics

The processed data set contains 356 unique subjects with 248 subjects in the training set and 106 subjects in the test set. This a reduction from the 399 unique subjects in the baseline approach 5.1.1 because we require at least three observations for a subject in this approach, (compared to at least two in the baseline approach). Class proportion is approximately 3:1 in both training and test sets in favor of the negative class, i.e. the class of PD patients that did not experience psychotic symptoms over the course of the study. We use 10 folds for both cross validation and nested cross validation. Models are tuned to give the best sensitivity possible.

Confusion matrices and ROC curves for the models presented in this section can be found in Appendix E - *Derived Statistics Approach*.

Logistic Regression

As stated in 4.3.1, the hyper-parameters we want to tune for this model are:

- Regularization penalty, L: possible values are [L1, L2]
- Regularization strength, C: range [0, ∞]

Our scikit-learn solver is 'liblinear' and we use 'balanced' class weights. Note that a lower value of C gives a stronger regularization.

The tuning phases used to find the final tuned model are summarized in Table 5.15.

Tuning Phase	Random Search 1	Random Search 2	Grid search 1	Grid search 2
Parameters	C,L	C,L	C,L	C,L
Range/Grid C	[0,10]	[0,10]	0.16,0.01	[0.1,0.5], SS = 0.01
Range/Grid L	L1	L2	L1,L2	L1
Chosen C	0.16	0.01	0.16	0.22
Chosen L	L1	L2	L1	L1

Table 5.15: Derived Statistics Approach: Logistic Regression Tuning. SS = Step Size

We first perform a random search with 1000 iterations to find an estimate of the optimal regularization strength for L1-norm regularization penalty. Results indicate a regularization strength of around 0.16 for L1-norm.

Secondly, we perform a random search with 1000 iterations to find an estimate of the optimal regularization strength for L2-norm regularization penalty. Results indicate a regularization strength of around 0.01 for L2-norm.

Using the results from the first two searches we perform a grid search with possible values for regularization penalty being L1 or L2 and regularization strength being 0.16 or 0.01. Results indicate that L1-penalty performs better than L2. A last grid search determines the optimal strength of regularization.

The chosen hyper-parameters from grid search is L1 for regularization penalty and 0.22 for regularization strength. The results from nested cross validation, training and test set are summarized in Table 5.16.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.698	0.375	0.587	0.732	0.454	0.659	0.276
Nested CV - std	0.070	0.155	0.263	0.083	0.188	0.122	0.229
Training results	0.754	0.483	0.724	0.763	0.579	0.744	0.432
Test results	0.708	0.425	0.680	0.716	0.523	0.698	0.347

Table 5.16: Derived Statistics Approach: Logistic Regression Results

Support Vector Machines (SVM)

As stated in 4.3.1 we want to try different kernels for our SVM models.

For a linear kernel we want to tune the following hyper-parameter:

- Regularization strength, C: range $[0, \infty]$

We use 'balanced' class weights. Note that a lower value of C gives a stronger regularization.

The tuning phases used to find the final tuned model for a linear kernel are summarized in Table 5.17.

Tuning Phase	Random Search 1	Grid search 1
Parameters	C	C
Range/Grid C	[0,10]	[1.0,4.0], SS = 0.1
Chosen C	2.3	2.5

Table 5.17: Derived Statistics Approach: SVM Linear Kernel Tuning. SS = Step Size

We start by performing a random search with 1000 iterations on a linear kernel to find an estimate of the optimal regularization strength. Results indicate a regularization strength of around 2.3.

Using this result we perform a grid search. The chosen hyper-parameter is a regularization strength of 2,5 for a linear kernel.

The results from nested cross validation, training and test set are summarized in Table 5.18.

For a polynomial kernel we want to tune following hyper-parameters:

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.646	0.327	0.417	0.716	0.349	0.566	0.127
Nested CV - std	0.104	0.138	0.192	0.129	0.147	0.109	0.200
Training results	0.899	0.714	0.948	0.884	0.815	0.916	0.762
Test results	0.830	0.621	0.720	0.864	0.667	0.792	0.556

Table 5.18: Derived Statistics Approach: SVM Linear Kernel Results

- Regularization strength, C: range $[0, \infty]$
- Gamma scaling γ: range $(0, \infty]$
- Polynomial degree, d: range $[1, \infty]$
- Independent term, r: range $[0, \infty]$

The tuning phases used to find the final tuned model for a polynomial kernel are summarized in Table 5.19.

Tuning Phase	Random Search 1	Grid search 1
Parameters	C,γ,d,r	Cγ,d,r
Range/Grid C	[0,10]	[0.05,0.5], SS = 0.05
Range/Grid γ	(0,10]	[1.0,1.4], SS = 0.1
Range/Grid d	[1,5]	[1,5], SS = 1
Range/grid r	[0,10]	[0.0,2.0], SS = 0.5
Chosen C	0.3	0.1
Chosen γ	1.2	1.3
Chosen d	3	1
Chosen r	1.3	0

Table 5.19: Derived Statistics Approach: SVM Polynomial Kernel Tuning. SS = Step Size

We start by performing a random search with 1000 iterations on a polynomial kernel to find estimates for the optimal values of regularization strength, gamma scaling, polynomial degree and independent term. Results indicate a regularization strength of around 0.3, gamma scaling of around 1.2, polynomial degree of around 3 and and independent term of around 1.3

Using these result we perform a grid search. The chosen hyper-parameters from this grid search are as follows:

- Regularization strength: 0.1
- Gamma scaling: 1.3
- Polynomial degree: 1
- Independent term: 0

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table 5.20.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.634	0.290	0.437	0.695	0.344	0.566	0.112
Nested CV - std	0.050	0.084	0.205	0.057	0.125	0.095	0.163
Training results	0.806	0.561	0.793	0.811	0.657	0.802	0.543
Test results	0.745	0.475	0.760	0.741	0.585	0.750	0.439

Table 5.20: Derived Statistics Approach: SVM Polynomial Kernel Results

For a radial kernel we want to tune the following hyper-parameters:

- Regularization strength, C: range $[0, \infty]$
- Gamma scaling, γ: range $(0, \infty]$

The tuning phases used to find the final tuned model for a radial kernel are summarized in Table 5.21.

Tuning Phase	Random Search 1	Grid search 1
Parameters	C,γ	C,γ
Range/Grid C	[1,10]	[2,10], SS = 0.1
Range/Grid γ	[0,1]	0.0001,0.001,0.01,0.1
Chosen C	6.9	2.9
Chosen γ	0.0001	0.001

Table 5.21: Derived Statistics Approach: SVM Radial Kernel Tuning. SS = Step Size

We start by performing a random search with 1000 iterations on a radial kernel to find estimates for the optimal values of regularization strength and gamma scaling. We choose

value ranges to avoid a highly biased radial SVM model as happened in 5.1.1. The chosen hyper-parameters are a regularization strength of 6.9 and gamma scaling of 0.0001.

Based on these results we perform a grid search. The chosen hyper-parameters are a regularization strength of 2.9 and gamma scaling of 0.001.

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table 5.22.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.630	0.351	0.570	0.653	0.417	0.611	0.201
Nested CV - std	0.183	0.202	0.305	0.243	0.219	0.164	0.302
Training results	0.762	0.494	0.707	0.779	0.582	0.743	0.436
Test results	0.736	0.463	0.760	0.728	0.576	0.744	0.426

Table 5.22: Derived Statistics Approach: SVM Radial Kernel Results

Decision Tree

As stated in 4.3.1, the hyper-parameter we want to tune for this model is:

- Complexity parameter, α: range $[0, \infty]$

We use 'balanced' class weights.

The tuning phases used to find the final tuned model for a decision tree are summarized in Table 5.23.

Tuning Phase	Random Search 1	Grid search 1
Parameters	α	α
Range/Grid α	[0,0.05]	[0.0,0.05], SS = 0.001
Chosen α	0.03	0.025

Table 5.23: Derived Statistics Approach: Decision Tree Tuning. SS = Step Size

We start by performing a random search with 1000 iterations on a decision tree to find an estimate for the optimal value of α. The chosen value for α is around 0.03.

Using this result we perform a grid search. The chosen α is 0.025.

The results from nested cross validation, training and test set for this hyper-parameter are summarized in Table 5.24.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.621	0.350	0.640	0.616	0.444	0.628	0.227
Nested CV - std	0.124	0.121	0.227	0.145	0.145	0.135	0.235
Training results	0.754	0.485	0.828	0.732	0.611	0.780	0.483
Test results	0.670	0.361	0.520	0.716	0.426	0.618	0.212

Table 5.24: Derived Statistics Approach: Decision Tree Results

We can produce a visualization of this decision tree. As the chosen decision tree only has a depth of 2, the entire tree is visualized. The visualization can be viewed in Figure 5.4.

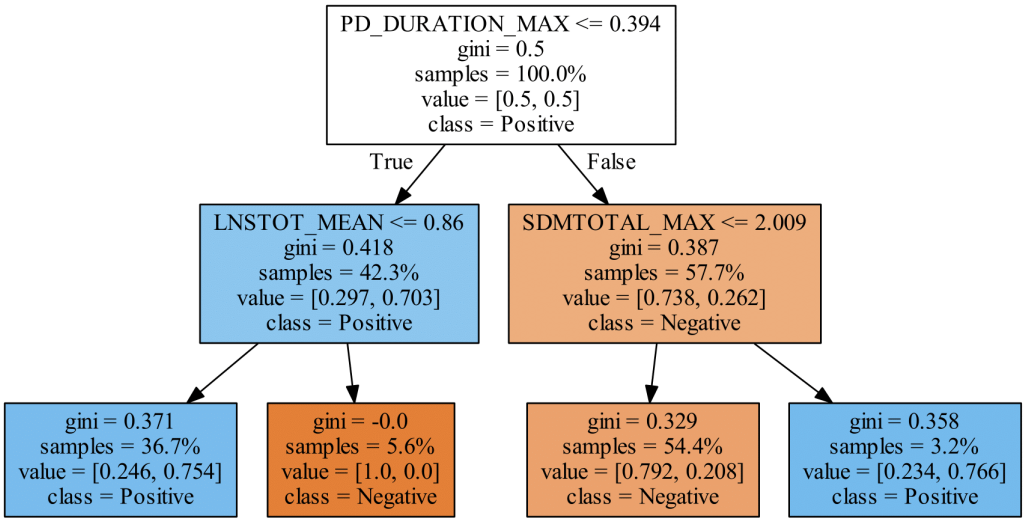


Figure 5.4: Visualization of the tree structure for the derived statistics approach, all of the tree is visible

Random Forest

As stated in 4.3.1, the hyper-parameters we want to tune for this model are:

- Number of trees in the forest, n: range $[1, \infty]$

- Fraction of features considered at each split, m: range [0, 1], if $m = 0$, then the minimum of 1 feature is considered at each split.
- Maximum depth of trees in the forest, d: range [1, ∞].

We use 'balanced' class weights.

The tuning phases used to find the final tuned model for a decision tree are summarized in Table 5.25.

Tuning Phase	Random Search 1	Grid search 1
Parameters	n,m,d	n,m,d
Range/Grid n	[10,100]	[80,120], SS = 2
Range/Grid m	[0.5,0.9]	[0.50,0.90], SS = 0.05
Range/Grid d	[1,20]	[1,2], SS = 1
Chosen n	97	114
Chosen m	0.70	0.60
Chosen d	1	1

Table 5.25: Derived Statistics Approach: Random Forest Tuning. SS = Step Size

We start by performing a random search with 1000 iterations on a random forest to find estimates for the optimal values of n, m and d. The chosen values are $n = 97$, $m = 0.7$ and $d = 1$.

Using these results we perform a grid search. The chosen hyper-parameters are $n = 114$, $m = 0.60$ and $d = 1$.

The results from nested cross validation, OOB, training and test set for this set of hyper-parameters are summarized in Table 5.26.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.702	0.430	0.713	0.700	0.529	0.707	0.363
Nested CV - std	0.122	0.138	0.246	0.120	0.170	0.152	0.262
OOB	0.718	0.439	0.741	0.711	0.551	0.726	0.391
Training results	0.726	0.448	0.741	0.721	0.558	0.731	0.402
Test results	0.698	0.410	0.640	0.716	0.500	0.678	0.313

Table 5.26: Derived Statistics Approach: Random Forest Results

We can plot the importance of the features used in the random forest model. The higher value a feature has, the more important it is for the model. The set of feature importance values sum to 1. The plot can be viewed in Figure 5.5.

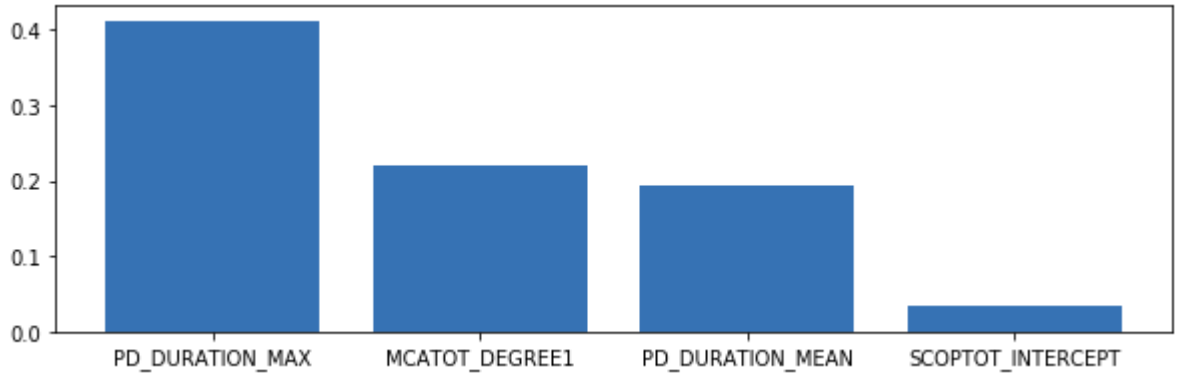


Figure 5.5: Feature importance values plotted as a bar plot, only the 4 most important features are shown

Boosted Trees

As stated in 4.3.1, the hyper-parameters we want to tune for this model are:

- number of boosting rounds, n: range $[1, \infty]$
- max_depth, d: range $[1, \infty]$
- min_child_weight, w: range $[0, \infty]$
- gamma, γ: range $[0, \infty]$
- subsample, $s1$: range $(0, 1]$
- colsample_bytree, $s2$: range $(0, 1]$
- eta, η: range $[0, 1]$

The tuning phases used to find the final tuned model for boosted trees are summarized in Table 5.27.

We start by performing a random search with 1000 iterations to find estimates for the optimal values of the hyper-parameters. Results indicate these values:

- number of boosting rounds, n: 8
- max_depth, d: 2

Tuning Phase	Random Search 1	Grid search 1	Grid search 2	Random search final
Parameters	n,d,w,$s1$,$s2$,η	n,d,w,$s1$,$s2$,η	n,d,w,$s1$,$s2$,η	n,d,w,$s1$,$s2$,η
Range/Grid n	[5,20]	[6,10], SS = 2	[10,14], SS = 2	[5,20]
Range/Grid d	[1,10]	[1,3], SS = 1	[1,2], SS = 1	[1,4]
Range/Grid w	[1,10]	[8,12], SS = 2	[9,11], SS = 1	[5,15]
Range/Grid γ	[0,5]	[2,4], SS = 1	[1.5,2.5], SS = 0.5	[0,5]
Range/Grid $s1$	[0.5,1]	[0.6,0.8], SS = 0.1	[0.9,1.0], SS = 0.1	[0.8,1.0]
Range/Grid $s2$	[0.5,1]	[0.4,0.6], SS = 0.1	[0.5,0.7], SS = 0.1	[0.5,1.0]
Range/Grid η	[0.3,1]	[0.5,0.7], SS = 0.1	[0.4,0.6], SS = 0.05	[0.3,2.0]
Chosen n	8	10	14	5
Chosen d	2	1	1	1
Chosen w	9.7	10	9	6.46
Chosen γ	2.7	2	1.5	3.14
Chosen $s1$	0.91	1.0	1.0	0.88
Chosen $s2$	0.74	0.6	0.5	0.89
Chosen η	0.62	0.5	0.4	0.45

Table 5.27: Derived Statistics Approach: Boosted Trees Tuning.
SS = Step Size

- min_child_weight, w: 9.7
- gamma, γ: 2,7
- subsample, $s1$: 0.91
- colsample_bytree, $s2$: 0.74
- eta, η: 0.62

Using these results we perform a series of grid searches. After a number of grid searches it became apparent that we would not get a stable set of hyper-parameters using grid search. This is the same problem we experienced with boosted trees in the baseline approach, (see 5.1.1). We therefore decided to estimate the optimal hyper-parameters using a random search with 10000 iterations and use the results to build our model. The chosen hyper-parameters from this final random search are as follows:

- number of boosting rounds, n: 5

- max_depth, d: 1
- min_child_weight, w: 6.46
- gamma, γ: 3.14
- subsample, $s1$: 0.88
- colsample_bytree, $s2$: 0.89
- eta, η: 0.45

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table 5.28.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.694	0.408	0.637	0.711	0.489	0.674	0.310
Nested CV - std	0.112	0.161	0.258	0.109	0.183	0.149	0.266
Training results	0.831	0.593	0.879	0.816	0.708	0.848	0.618
Test results	0.670	0.375	0.600	0.691	0.462	0.646	0.255

Table 5.28: Derived Statistics Approach: Boosted Trees Results

We can plot the importance of the features used in the boosted trees model. The feature importance is measured in the total information gain for each split using this feature in the model. The higher value a feature has, the more important it is for the model. The plot can be viewed in Figure 5.6. All features used are plotted due to the low amount of boosted trees (5) with a maximum depth of 1.

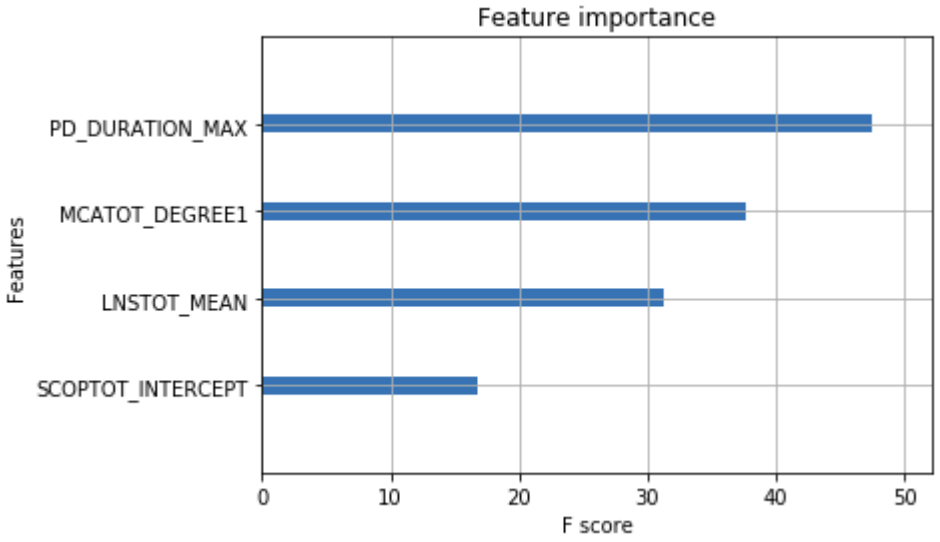


Figure 5.6: Plot of feature importance values for boosted trees. All features are shown.

5.1.3 Deep Learning Approach

This section presents the results from our LSTM network trained on the "deep learning approach" as detailed in 4.2.8. The following preproccesing steps are applied on the data set for this approach:

- Mode/mean missing value imputation
- Feature reduction through total scores
- Train/validation/test set ratio of 60/20/20
- Prediction based on deep learning

The processed data set contains 399 unique subjects with 239 subjects in the train set, 80 subjects in the validation set and 80 subjects in the test set. This is the same amount of unique subjects as in the baseline approach, 5.1.1. Class proportion is approximately 3:1 in both training, validation and test sets in favor of the negative class, i.e. the class of PD patients that did not experience psychotic symptoms over the course of the study. The LSTM network is tuned to give the best sensitivity possible.

We will follow the tuning strategy detailed in 4.3.2. The tuning process is summarized in Table 5.29.

We start off with a simple LSTM network with one LSTM layer and one binary output layer. We use binary cross-entropy as our loss function and RMSprop as our optimizer.

Tuning Phase	1	2	3	4
Parameter	n	o, η	d	l
Grid	[2,12], SS = 2	[Adam, RMSprop, SGD], [0.001,0,01,0.1]	[0.0,0.5], SS = 0.1	[1,2,3]
Chosen value	6	Adam, 0.1	0.0	1

Table 5.29: Deep Learning Approach: LSTM Tuning.
SS = Step Size
Parameters:
n = number of neurons in the lstm layer
o = optimizer used to train the network
η = learning rate
d = dropout rate
l = number of LSTM layers

Learning rate is set to default. We first want to tune the number of neurons in the LSTM layer. We perform a grid search with the following grid for the amount of neurons in the LSTM layer: [2, 4, 6, 8, 10, 12]. The grid contains a relatively low amount of neurons due to results from preliminary testing. Results indicate that a LSTM layer with 6 neurons performs the best for sensitivity.

Next we want to tune the optimizer used and the associated learning rates. We perform a grid search with the following grid for optimizers ["Adam", "RMSprop", "SGD"] and the following grid for learning rates [0.001, 0.01, 0.1]. Results indicate that using "Adam" as the optimizer with a learning rate of 0.1 gives the best performance for sensitivity.

Next we take a look at training results and compare them with validation results to see if the current model is overfitting. Results indicates that the current model overfits the training data slightly. We can try to decrease overfitting by adding a dropout layer between the input layer and the LSTM layer. We add a dropout layer to the model and perform a grid search with the following grid for dropout rates [0, 0.1, 0.2, 0.3, 0.4, 0.5]. Note that a dropout rate of 0 makes the dropout layer inactive, so results from this value should be equivalent to not having a dropout layer in the model. Results indicate that the model does not benefit from a dropout layer.

Finally, we will try adding more LSTM layers to the model. We will try stacking one and two LSTM layers on top of our original one. Results indicate that adding additional LSTM layers to the model is not beneficial.

Results from 10 different LSTM networks trained with these hyper-parameters are presented in Table 5.30. The table shows the mean performance of these models on different metrics as well as the standard deviation. Results are shown for the training, validation and test sets.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Train - mean	0.831	0.674	0.678	0.883	0.669	0.781	0.562
Train - std	0.044	0.1	0.13	0.049	0.092	0.065	0.119
Validation - mean	0.735	0.48	0.577	0.788	0.52	0.683	0.346
Validation - std	0.024	0.044	0.084	0.045	0.042	0.031	0.052
Test - mean	0.681	0.402	0.411	0.777	0.401	0.594	0.188
Test - std	0.047	0.08	0.1	0.063	0.077	0.052	0.103

Table 5.30: Deep Learning Approach: LSTM Results

A Confusion matrix and ROC curve for a representative LSTM model can be found in Appendix E - *Deep Learning Approach.*

5.2 Analysis and Discussion

5.2.1 Baseline Approach

This subsection contains our analysis and discussion relating to the results presented in 5.1.1 *Results - Baseline Approach.*

From Tables 5.31 and 5.32 we can see that logistic regression is our top performer on sensitivity and on metrics that emphasis a balanced classification (f1, roc_auc, MCC). Second place could arguably go to either random forest or SVM-linear. SVM-linear performs well, but slightly worse on most metrics compared to logistic regression. Random forest has the best performance of all models on some metrics, but is not able to classify most of the minority class correctly and therefore performs poorly on sensitivity and as a result, suffers on f1, roc_auc and MCC.

Our worst performing models are boosted trees, SVM-radial and the decision tree. SVM-radial and boosted trees models do not perform particularly well on neither the minority class or the majority class. This is reflected in their poor placement on the balanced

classification metrics (f1, roc_auc, MCC). Our decision tree performs the worst on metrics such as accuracy, precision and specificity, but is in 2nd place for sensitivity. Despite this, it performs badly on the balanced classification metrics.

SVM-polynomial is an average performer. It is important to notice that our SVM-polynomial is actually a linear model as the polynomial degree is 1. The difference in results compared to SVM-linear is explained by gamma scaling still being applied to this "linear" SVM-polynomial model.

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
LR	0.658	0.407	**0.710**	0.640	**0.518**	**0.675**	**0.308**
SVM - Linear	0.683	0.422	0.613	0.708	0.500	0.660	0.290
SVM - Poly	0.683	0.419	0.581	0.719	0.486	0.650	0.274
SVM - Radial	0.658	0.375	0.484	0.719	0.423	0.601	0.188
DT	0.600	0.356	0.677	0.573	0.467	0.625	0.219
RF	**0.725**	**0.471**	0.516	**0.798**	0.492	0.657	0.305
BT	0.667	0.395	0.548	0.708	0.459	0.628	0.234

Table 5.31: Baseline Approach: Test Results
LR = Logistic Regression
DT = Decision Tree
RF = Random Forest
BT = Boosted Trees

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
LR	5/6th	4th	**1st**	6th	**1st**	**1st**	**1st**
SVM - Linear	2/3rd	2nd	3rd	4/5th	2nd	2nd	3rd
SVM - Poly	2/3rd	3rd	4th	2/3rd	4th	4th	4th
SVM - Radial	5/6th	6th	7th	2/3rd	7th	7th	7th
DT	7th	7th	2nd	7th	5th	6th	6th
RF	**1st**	**1st**	6th	**1st**	3rd	3rd	2nd
BT	4th	5th	5th	4/5th	6th	5th	5th

Table 5.32: Baseline Approach: Test Results Ranked
LR = Logistic Regression
DT = Decision Tree
RF = Random Forest
BT = Boosted Trees

There are signs that the test results are perhaps too optimistic. Table 5.33 shows the average result from all tuned models together. We decided to show the difference between nested cross validation and test results in this manner in order to provide a comparison

that addresses all models. Looking at the same Table 5.33, we can see that on average, nested cross validation gives lower estimates for all metrics measured. It is also important to note that the average standard deviation from nested cross validation is relatively high. These two observations put together might indicate that our data set has a significant amount of variance. Some hyper-parameters chosen for our models might also indicate this as they seem to favour less complex models. Some examples are:

- Logistic regression has a large amount of regularization at 0.01.
- SVM with Polynomial kernel is reduced to a less complex linear model (degree 1) with a high degree of regularization (0.02).
- Our random forest is relatively simple with 22 trees all grown with a maximum depth of 1.

There are also some models, such as SVM with a linear kernel, that do not follow this pattern with a relatively high regularization parameter of 3.3.

The relatively low complexity of some of our models might also be explained by limitations to our tuning procedures. We start tuning a model by using an initial range for the hyper-parameters. We proceed with the tuning process on the assumption that these initial ranges contain or at least border the optimal hyper-parameter values. If the initial ranges are far from the optimal values, we might artificially limit the complexity of our models by not searching for hyper-parameters in ranges that would give a higher complexity.

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - avg. mean	0.590	0.316	0.479	0.629	0.368	0.554	0.101
Nested CV - avg. std	0.106	0.121	0.204	0.142	0.131	0.103	0.192
Test Set - avg.	0.659	0.401	0.604	0.679	0.478	0.641	0.256

Table 5.33: Baseline Approach: Nested CV vs. Test

Inference

We can infer some things about the relative importance of specific features by looking at the tree visualization of the decision tree (Figure 5.1) and the plots of feature importance

from random forest and boosted trees (Figures 5.2 and 5.3). We can see from these figures that the three models do not agree entirely on what features are important. We can however see that both the decision tree and random forest emphasise the importance of SCOPTOT. SCOPTOT corresponds to the total score of a subject on SCOPA[1], which is used to evaluate autonomic symptoms in patients with Parkinson's disease. This seems to support the results in Barret [3] which found that *"greater automatic symptoms at baseline were associated with increased risk of reporting psychotic symptoms on ≥2 occasions"*.

Boosted trees emphasise the importance of SIDTTOT instead. SIDTTOT corresponds to the total score of a subject on the University of Pennsylvania Smell Identification Test (UPSIT), used to test the function of an individual's olfactory system. This seems to support the results in Ffytche [5] which associated lower olfactory results at baseline with an increased risk of psychosis.

5.2.2 Derived Statistics Approach

This subsection contains our analysis and discussion relating to the results presented in 5.1.2 *Results - Derived Statistics Approach*.

Looking at Tables 5.34 and 5.35 we see that there is a clear ranking of our models in terms of performance. The various SVM models are our best performing ones. In first place we have SVM with a linear kernel. It is by some margin our best performing model, ranked first on all metrics except for sensitivity. Second and third place go to SVM with respectively polynomial and radial kernels. These perform quite similar. It must be noted that the polynomial kernel has a degree of 1 (as was the case in the baseline approach) and is therefore essentially a linear kernel. As in the baseline approach, the difference between SVM-linear and our "linear" SVM-polynomial can be explained by the addition of gamma scaling. Logistic regression is in 4th place with a performance below that of SVM. Our various tree based models perform quite poorly in this approach. Random forest is our best performing tree based model and decision tree is the worst.

As in the baseline approach there are signs that the test results are too optimistic. Looking at Table 5.36 that shows average result from all tuned models together, we can see that on average, nested cross validation gives lower estimates for all metrics measured. Again, the average standard deviation from nested cross validation is also quite high. These observations might indicate, as in the baseline approach, that we are dealing with a data set with a significant amount of variance. Hyper-parameters chosen

[1] Scales for Outcomes in Parkinson's Disease - Autonomic Dysfunction

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
LR	0.708	0.425	0.680	0.716	0.523	0.698	0.347
SVM - Linear	**0.830**	**0.621**	0.720	**0.864**	**0.667**	**0.792**	**0.556**
SVM - Poly	0.745	0.475	**0.760**	0.741	0.585	0.750	0.439
SVM - Radial	0.736	0.463	**0.760**	0.728	0.576	0.744	0.426
DT	0.670	0.361	0.520	0.716	0.426	0.618	0.212
RF	0.698	0.410	0.640	0.716	0.500	0.678	0.313
BT	0.670	0.375	0.600	0.691	0.462	0.646	0.255

Table 5.34: Derived Statistics Approach: Test Results
LR = Logistic Regression
DT = Decision Tree
RF = Random Forest
BT = Boosted Trees

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
LR	4th	4th	4th	4/5/6th	4th	4th	4th
SVM - Linear	**1st**	**1st**	3rd	**1st 1st**	**1st**	**1st** 6	**1st**
SVM - Poly	2nd	2nd	**1/2nd**	2nd	2nd	2nd	2nd
SVM - Radial	3rd	3rd	**1/2nd**	3rd	3rd	3rd	3rd
DT	6/7th	7th	7th	4/5/6th	7th	7th	7th
RF	5th	5th	5th	4/5/6th	5th	5th	5th
BT	6/7th	6th	6th	7th	6th	6th	6th

Table 5.35: Derived Statistics Approach: Test Results Ranked
LR = Logistic Regression
DT = Decision Trees
RF = Random Forest
BT = Boosted Trees

for the models in this approach do seem however to favour more complexity than the corresponding models in the baseline approach.

- Logistic regression has a regularization parameter of 0.22 compared to 0.01 in the baseline approach.
- SVM with Polynomial kernel is of degree 1 as in the baseline approach, but has a regularization parameter of 0.1 instead of 0.02.
- Our random forest still has a maximum depth of 1, but has increased the number of trees grown; from 22 in the baseline approach to 114 in this approach.

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - avg. mean	0.661	0.362	0.572	0.689	0.432	0.630	0.231
Nested CV - avg. std	0.109	0.143	0.242	0.127	0.168	0.132	0.237
Test Set - avg.	0.725	0.450	0.669	0.742	0.536	0.705	0.368

Table 5.36: Derived Statistics Approach: Nested CV vs. Test

Inference

As in the baseline approach, we can infer some things about the relative importance of specific features by looking at the tree visualization of the decision tree (Figure 5.4) and the plots of feature importance from random forest and boosted trees (Figures 5.5 and 5.6).

Looking at these figures we can see that all models agree that PD_DURATION_MAX is the most important feature. This feature logically corresponds to how long a subject has had a a PD diagnosis at the last observation available for that subject. This "last" observation is either the last observation before the response is positive or the last observation of that subject in the study. This indicates that there is an increased risk of showing symptoms of psychosis the longer a subject has been diagnosed with PD. This is consistent with multiple sources, including the American Parkinson Disease Association [1] which says *"A number of risk factors – both internal and external- are associated with the condition. Some of these risk factors include: age,* ***duration and severity of Parkinson's disease****; and the taking of dopamine therapy"*. Another source, a research article from Thanvi et al [39] says *"Psychosis is common in Parkinson's disease (PD), particularly in its later stages."*.

Both random forest and boosted trees also place some emphasis on MCATOT_DEGREE1. This corresponds to the slope of a subject's scores on the Montreal Cognitive Assessment (MoCA). MoCA is used as an assessment test for cognitive function in a patient. This seems to indicate that a decline in cognitive function is associated with an increased risk of psychosis in PD patients. This is consistent with the source we used to justify our inclusion of a feature that measured cognitive impairment in our feature selection, 4.1.2. That source, Thanvi et al [39] says *"Old age,* ***cognitive impairment****, history of depression, and sleep disorders are important risk factors for the development of psychosis in PD.*

5.2.3 Deep Learning Approach

This subsection contains our analysis and discussion relating to the results presented in 5.1.3 *Results - Deep Learning Approach*.

The test results from our LSTM network show some of the worst performance of all the models in this thesis. The results from the validation set is better, but the model is also tuned to give a good performance for this particular subset. If we compare the results from the training set to that of the test or validation set, we do not detect a problematic amount of overfitting. We can also not see a large amount of variance in the standard deviation of the test results. We must therefore conclude that our deep learning approach did not work particularly well for this classification problem.

There might be a number of reasons why our deep learning approach was not successful. Firstly, we only explored LSTM networks with a limited set of hyper-parameters. It is fully possible that a different LSTM network built in a different way would produce good results. A second reason might be that we have a relatively small data set for a relatively complex prediction. In general it is expected that a neural network requires more data to train on than other machine learning models. According to [13] *"a rough rule of thumb is that a supervised deep learning algorithm will generally achieve acceptable performance with around 5,000 labeled examples per category"*, which is much more than our data set contains. This might explain why our classical ML models are performing better for this data set, as they are better able to handle the limited amount of data. In fact, due to the need for a validation set, our LSTM network had access to the least amount of training data of all models trained in this project.

5.2.4 Comparing the Approaches

A comparison of results from the three different prediction approaches can be found in Table 5.37, which combines results from Tables 5.36, 5.33 and 5.30. If we are strictly looking at the test results, we can rank the different approaches as:

1. Derived Statistics Approach
2. Baseline Approach
3. Deep Learning Approach

The derived statistics approach is the clear winner on all metrics. The baseline and deep learning approaches are more similar in performance, but baseline massively outperforms

deep learning on sensitivity, and as a result, also outperforms deep learning on f1, roc_auc and MCC. We can see that on average the classical ML models improve with the derived statistics approach, compared to the baseline approach. This is expected as we are able to utilize temporal information and therefore have access to more data in this approach.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
BL NCV avg.	0.590	0.316	0.479	0.629	0.368	0.554	0.101
BL Test	0.659	0.401	0.604	0.679	0.478	0.641	0.256
DE NCV avg.	0.661	0.362	0.572	0.689	0.432	0.630	0.231
DE Test	0.725	0.450	0.669	0.742	0.536	0.705	0.368
DL Val	0.735	0.48	0.577	0.788	0.52	0.683	0.346
DL Test	0.681	0.402	0.411	0.777	0.401	0.594	0.188

Table 5.37: Comparing results from all approaches
BL NCV avg. = Baseline Approach Nested Cross Validation average
BL Test = Baseline Approach Test results
DE NCV avg. = Derived Statistics Approach Nested Cross Validation average
DE Test = Derived Statistics Test results
DL Val avg. = Deep Learning Validation Results average
DL Test avg = Deep Learning Test Results average

Prediction Sensitivity vs. Years After Baseline

Plots of the sensitivity of our models vs. years after baseline can be found in Appendix F. These plots show how our models perform on subjects of the minority class that experience their first psychotic symptoms after n years.

For our models in the baseline approach we see almost no pattern in the sensitivity vs. years plots. The only model that shows any pattern is boosted trees. This model seems to show a steady decline in sensitivity after year 2. All models also seem to have in common that they perform relatively poorly or average on subjects who experience symptoms of psychosis during the first year after baseline.

Our models in the derived statistics approach show more clear patterns in their sensitivity vs. years plots. Surprisingly, all models in the derived statistics approach have a 100% sensitivity for subjects who experience their first symptoms of psychosis in year 2. In addition, 5 out of 7 models have a 100% sensitivity for subjects in year 3. Note that there are no subjects in the data set for the derived statistics approach that experience their first symptoms of psychosis during the first year. This is due to the fact that we

have to remove these subjects for this approach as we can not take meaningful statistics such as slope or mean for a subject that only has one observation. We require a subject to have at least two years of non-psychotic observations in order to derive statistics. Unfortunately, this filters out any subject who experiences symptoms of psychosis the first year for this approach.

The difficulty the baseline approach models showed with subjects who experience symptoms the first year might partly explain why the derived statistics approach is our best performing approach. By not including this difficult group, the derived statistics approach models might show better performance metrics due to an easier test set to classify. If we look at the sensitivity vs. years plot for the LSTM network we can perhaps see a confirmation of this. The LSTM network has a 100% sensitivity for year 2 and 3, but struggles with year 1.

5.2.5 Improving Results

An attempt to improve results by using HEOM instead of mode/mean for missing value imputation is explored in Appendix G. The results show that the difference between using HEOM or mode/mean for missing value imputation is minimal. Mode/mean might provide slightly better results.

An attempt to improve results by using PCA instead of total scores for feature reduction is explored in Appendix H. The results show that with a similar amount of reduced features, total scores performs better than PCA.

Chapter 6

Future Work and Conclusion

6.1 Challenges

In this section we briefly outline the major challenges we faced during the project.

At the very beginning of this project we experienced an entrance into a field that was unknown to us. Our first impression of Parkinson's disease was that most people have some knowledge about it, but when we dived deeper into relevant literature and data, we found that we lacked understanding of relevant medical terminology. That was an issue we had to cope with in order to understand the meaning of the data we used. Only when that prerequisite was met were we able to develop this project and apply machine learning techniques in an appropriate way.

With progress in understanding the data and the problem at hand, more questions arose. The most challenging was how to benefit from the longitudinal data in the PPMI database: first in classical machine learning algorithms and then in deep learning. Studying how to apply longitudinal statistics and how to use time series in neural networks took time because these techniques were not known to us beforehand.

Another question was how to handle the mixture of categorical, ordinal and numerical features in our data set. This mix of features made techniques such as data standardization or PCA harder to use. We had to think about questions such as: *Do we standardize an ordinal variable?* or *Is PCA a reliable feature reduction method when used on data with ordinal or categorical features?*.

We also struggled with a slightly imbalanced data set, where the number of subjects of the positive class were outnumbered 3:1 by subjects of the negative class. We experimented briefly with oversampling techniques, but this did not yield good results. We therefore

decided to balance class weights in our models in order to address this imbalance. We also made use of multiple performance metrics, including metrics that specifically rewards classifiers that perform well on the minority class.

Finally, we had to solve the problem of a smaller than ideal data set. Because of the size of our data set we wanted to avoid using a validation set as this would have reduced the size of our training and test sets by a significant amount. We therefore had to search for alternative model validation techniques. This search eventually lead us to using nested cross validation for model validation.

6.2 Future Work

In this section we discuss some possible ideas and/or improvements to the project that could be explored in an extended thesis or in other works that build upon this project.

One possible idea is extending the binary classification task in this project into a multiclass classification task. We limited this project to a binary classification problem due to a relatively small data set with few examples of the most severe symptoms of psychosis. We thought it would be unrealistic to try to predict the severity of psychotic symptoms with such a data set. In case of access to a larger data set or multiple data sets that can be combined, one could consider a multiclass classification task that tries to predict the severity of psychotic symptoms as well as their presence. Another possibility is to use the current data set for multi class classification, but to augment the scarce classes with synthetic observations produced by techniques such as SMOTE [72].

Another possible improvement could be the use of multiple imputation as mentioned in 3.3.1 for handling missing values in the data set. As we show in Appendix G, HEOM imputation does not bring any improvements over mean/mode imputation, but multiple imputation has the potential to improve the validity of medical research, according to [73]. Multiple imputation is more complex to implement, but could provide more realistic values, because it does not underestimate the variance in the data in the amount that single imputation methods do.

An improvement in the prediction power of our models could come from the inclusion of additional features in our data set that are biological in nature. Brain imaging scans, test results from cerebrospinal fluid (CSF) samples, test results from blood samples, etc. are all available in the PPMI database. Ffytche [5] found for example that a reduction in amyloid Aß1-42[1] in CSF samples is associated with an increased risk of early onset psychosis in PD patients. The inclusion of this feature might therefore help our models in

[1] Aggregates of various proteins [74]

their predictions. We ultimately decided to use features that are not biological in nature because we wanted our models to be applicable to a wide range of PD patients. Biological features are often more expensive and/or time consuming to collect than non-biological ones and are therefore less available.

There are several preprocessing techniques that we did not explore in this thesis that could help improve results from our models. An intriguing idea was suggested to us by our supervisors. This involved using a technique called Multiple Discriminant Analysis (MDA) [75] to perform feature reduction in our data set. In short, MDA is a feature reduction method that tries to find new dimensions from the original data set that separates the classes the most. A limitation of MDA is that it can not produce more than $K - 1$ dimensions from the original data, where K is the number of classes. This makes MDA unsuitable for reducing a data set with a large amount of features like our data set. The idea however was to use MDA on data from individual tests (eg. MoCA), reducing the component features from a test into $K - 1$ features, which is only one dimension in our case. This would reduce the number of features in our data set by a similar amount as our feature reduction using total scores, but perhaps result in a set of more easily classified features because the new features would be dimensions that separate the classes well. Other notable preprocessing techniques that we did not explore in this thesis include outlier handling and correlation analysis.

Additionally, there are ways of using neural networks that were not explored in this thesis. A standard feed-forward neural network could have been used as a model in our baseline and derived statistics approaches and perhaps produce superior results. There are also different ways of constructing LSTM networks that were not explored. Ultimately we only explored LSTM networks with a limited set of hyper-parameters in this thesis. LSTM results could also potentially improve if a larger data set was used.

Finally, we would like to add that we believe the code we developed during this project could quite easily be adapted for an extension of this thesis or a project along similar lines. The code is constructed in a modular way where most preprocessing steps and all models can be added or removed without affecting the whole. In addition, the code can be run in multiple different ways by setting various options in the code, e.g. which preprocessing steps to run or which models to build. By building upon this foundation, the code can be altered or extended to accommodate different solutions.

The code is available on GitHub at https://github.com/Tjersland/Master_2020_Prediction_of_Psychosis_in_Parkinsons_Patients_using_Machine_learning

6.3 Conclusion

This project aimed to predict psychosis in PD patients using classical ML and deep learning models on selected features from the PPMI database. It is important to predict psychosis as early as possible, so that appropriate treatment can be started and negative effects can be reduced. We based ourselves on data that is more easily collected and therefore more available, like clinical tests, questionnaires and demographic data. We did not use less available data, like brain imaging scans or cerebrospinal fluid samples. This should make our models more useful for a wider selection of PD patients.

We developed three prediction approaches: (1) using only baseline data with classical ML models, (2) using derived statistics with classical ML models and (3) using full longitudinal data with a deep learning model. On average, and for all metrics used, the derived statistics approach performs the best. The baseline approach and deep learning approach are more similar in terms of performance, but the baseline approach performs better on most metrics. It is logical that the derived statistics approach outperforms baseline, as the derived statistics approach makes use of longitudinal data, while the baseline approach is limited to a single observation for each subject. The poor results from our deep learning model can likely be explained by a too small data set. It can perhaps also be that we did not find the right neural network architecture for an optimal prediction.

The best performing models for the derived statistics approach are SVM models with various kernels. The best performing model is SVM with a linear kernel. SVM with polynomial and radial kernels follow behind. The LR model, although simple, performs well and better than any tree based models.

The best performing model for the baseline approach is the LR model. SVM with a linear kernel performs similar to the LR model, but slightly worse on most metrics. Random forest model performs excellent on some metrics, but not on sensitivity.

The baseline and deep learning approaches had a large classification error on subjects who experience symptoms in the first year. For the derived statistics approach, such subjects are filtered out before applying the models. This is an unfortunate side-effect of requiring at least two observations for a subject in order to derive meaningful longitudinal statistics for that subject. This might partly explain why the derived statistics approach performs so well, because this approach did not have to classify this group of difficult subjects.

We perform inference on our tree based models. Some results from this inference is that greater autonomic symptoms at baseline, lower olfactory results at baseline, a

steep decline in cognitive function and how long the patient is diagnosed with PD are all associated with an increased risk of psychosis in PD patients. These findings are supported by cited scientific literature.

Our results show that relatively well performing classical ML models for predicting psychosis in PD patients can be developed using easily collected and often available data. Deep learning models most likely require a larger data set in order to get similar or better results. In addition, results from inference indicate that our models make sense from a clinical perspective as they emphasis the importance of features that are considered risk factors for PD psychosis by scientific literature on the subject.

List of Figures

List of Tables

Appendix A

Main Eligibility Criteria

The criteria highlighted in green are the same between the two cohorts. The non-highlighted criteria are different, but judged non-problematic. The red highlighted criteria are different and are deemed problematic.

A.1 De Novo Cohort

Main Eligibility Criteria	**Parkinson Disease (PD) Subjects:** Inclusion: Patients must have at least two of the following: resting tremor, bradykinesia, rigidity (must have either resting tremor or bradykinesia); OR either asymmetric resting tremor or asymmetric bradykinesia. A diagnosis of Parkinson disease for 2 years or less at Screening. Hoehn and Yahr stage I or II at Baseline. Confirmation from imaging core that screening dopamine transporter SPECT scan is consistent with dopamine transporter deficit (or for sites where DaTSCAN™ is not available, that VMAT-2 PET scan is consistent with VMAT deficit). Not expected to require PD medication within at least 6 months from Baseline. Male or female age 30 years or older at time of PD diagnosis. Exclusion: Currently taking levodopa, dopamine agonists, MAO-B inhibitors, amantadine or other PD medication. Has taken levodopa, dopamine agonists, MAO-B inhibitors or amantadine within 60 days of Baseline. Has taken levodopa or dopamine agonists prior to Baseline for more than a total of 60 days. Received any of the following drugs that might interfere with dopamine transporter SPECT imaging: Neuroleptics, metoclopramide, alpha methyldopa, methylphenidate, reserpine, or amphetamine derivative, within 6 months of Screening. Current treatment with anticoagulants (e.g., coumadin, heparin) that might preclude safe completion of the lumbar puncture. Condition that precludes the safe performance of routine lumbar puncture, such as prohibitive lumbar spinal disease, bleeding diathesis, or clinically significant coagulopathy or thrombocytopenia. Use of investigational drugs or devices within 60 days prior to Baseline (dietary supplements taken outside of a clinical trial are not exclusionary, e.g., coenzyme Q10).

A.2 Genetic Cohort

Genetic Cohort: Parkinson Disease (PD) subjects

Inclusion:

Patients must have at least two of the following: resting tremor, bradykinesia, rigidity (must have either resting tremor or bradykinesia); OR either asymmetric resting tremor or asymmetric bradykinesia.

A diagnosis of Parkinson disease for 7 years or less at Screening.

Hoehn and Yahr stage < 4 at Baseline.

Male or female age 18 years or older.

Willingness to undergo genetic testing and to be informed of genetic testing results

Confirmation of mutation in LRRK2, GBA or SNCA

For subjects taking any drugs that might interfere with dopamine transporter SPECT imaging (Neuroleptics, metoclopramide, alpha methyldopa, methylphenidate, reserpine, or amphetamine derivative) must be willing and able from a medical standpoint to hold the medication for at least 5 half-lives prior to screening DatSCAN™ imaging.

Exclusion:

Current treatment with anticoagulants (e.g. coumadin, heparin) that might preclude safe completion of the lumbar puncture.

Condition that precludes the safe performance of routine lumbar puncture, such as prohibitive lumbar spinal disease, bleeding diathesis, or clinically significant coagulopathy or thrombocytopenia.

Appendix B

PPMI Variable Definitions and Score Calculations

PARKINSON'S PROGRESSION MARKERS INITIATIVE

Play a Part in Parkinson's Research

Variable Definitions and Score Calculations

Summary

The following tables describe the calculation of the derived variables in the PPMI study.

Study Groups

Characteristic	Variables	Dataset
Enrolled PD Subject	PATNO, APPRDX, ENROLLDT Merge SCREEN with RANDOM and find each unique PATNO with APPRDX = '1' that is not missing ENROLLDT	RANDOM (Consent and Enrollment), SCREEN
Enrolled Healthy Control	PATNO, APPRDX, ENROLLDT Merge SCREEN with RANDOM and find each unique PATNO with APPRDX = '2' that is not missing ENROLLDT	RANDOM (Consent and Enrollment), SCREEN
Enrolled SWEDD Subject	PATNO, APPRDX, ENROLLDT Merge SCREEN with RANDOM and find each unique PATNO with APPRDX = '3' that is not missing ENROLLDT	RANDOM (Consent and Enrollment), SCREEN
Enrolled Prodromal Subject	PATNO, APPRDX, ENROLLDT Merge SCREEN with RANDOM and find each unique PATNO with APPRDX = '4' that is not missing ENROLLDT	RANDOM (Consent and Enrollment), SCREEN
Enrolled Genetic Cohort Subject	PATNO, APPRDX, ENROLLDT Merge SCREEN with RANDOM and find each unique PATNO with APPRDX = '5' or '6' that is not missing ENROLLDT APPRDX = '5' are PD subjects APPRDX = '6' are Unaffected subjects	RANDOM (Consent and Enrollment), SCREEN
Enrolled Genetic Registry Subject	PATNO, APPRDX, ENROLLDT Merge SCREEN with RANDOM and find each unique PATNO with APPRDX = '7' or '8' that is not	RANDOM (Consent and Enrollment),

	missing ENROLLDT APPRDX = '7' are PD subjects APPRDX = '8' are Unaffected subjects	SCREEN

Demographics and PD Characteristics

Age	ENROLLDT - BIRTHDT	RANDOM (Consent and Enrollment)
Gender	GENDER = 2 is Male GENDER = 0 or 1 is Female	RANDOM (Consent and Enrollment)
Race	RAINDALS, RAASIAN, RABLACK, RAHAWOPI, RAWHITE, RANOS Other = RAINDALS, RAHAWOPI, RANOS, or more than one race specified	SCREEN
Family History of PD	BIOMOMPD, BIODADPD, FULSIBPD, HALFSIBPD, MAGPARPD, PAGPARPD, MATAUPD, PATAUPD, KIDSPD Subject has family history of PD if any one or more of these variables = '1'	FAMHXPD
Disease Duration	PDDXDT, ENROLLDT Duration = number of months between PDDXDT and ENROLLDT	PDFEAT, RANDOM (Consent and Enrollment)
TD / PIGD Classification	First calculate Tremor and PIGD scores: Tremor score = Mean of the following variables: NP2TRMR, NP3PTRMR, NP3PTRML, NP3KTRMR, NP3KTRML, NP3RTARU, NP3RTALU, NP3RTARL, NP3RTALL, NP3RTALJ, NP3RTCON PIGD score = Mean of the following variables: NP2WALK, NP2FREZ, NP3GAIT, NP3FRZGT, NP3PSTBL Then calculate ratio = Tremor score / PIGD score.	NUPDRS2P, NUPDRS3

	If ratio ≥ 1.15, OR if PIGD score = 0 and Tremor score > 0, then subject is TD. If ratio ≤ 0.9 then subject is PIGD. If ratio > 0.9 and < 1.15, OR if Tremor score and PIGD score = 0, then subject is Indeterminate.	

Motor Assessments

MDS-UPDRS Part I	NP1COG, NP1HALL, NP1DPRS, NP1ANXS, NP1APAT,NP1DDS, NP1SLPN, NP1SLPD, NP1PAIN, NP1URIN, NP1CNST, NP1LTHD, NP1FATG Part I Score = sum of these 13 variables	NUPDRS1, NUPDRS1P
MDS-UPDRS Part II	NP2SPCH, NP2SALV, NP2SWAL, NP2EAT, NP2DRES, NP2HYGN, NP2HWRT, NP2HOBB, NP2TURN, NP2TRMR, NP2RISE, NP2WALK, NP2FREZ Part II Score = sum of these 13 variables	NUPDRS2P
MDS-UPDRS Part III	NP3SPCH, NP3FACXP, NP3RIGN, NP3RIGRU, NP3RIGLU, PN3RIGRL, NP3RIGLL, NP3FTAPR, NP3FTAPL, NP3HMOVR, NP3HMOVL, NP3PRSPR, NP3PRSPL, NP3TTAPR, NP3TTAPL, NP3LGAGR, NP3LGAGL, NP3RISNG, NP3GAIT, NP3FRZGT, NP3PSTBL, NP3POSTR, NP3BRADY, NP3PTRMR, NP3PTRML, NP3KTRMR, NP3KTRML, NP3RTARU, NP3RTALU, NP3RTARL, NP3RTALL, NP3RTALJ, NP3RTCON Part III Score = sum of these 33 variables	NUPDRS3
MDS-UPDRS Part IV	NP4WDYSK, NP4DYSKI, NP4OFF, NP4FLCTI, NP4FLCTX, NP4DYSTN Part IV Score = sum of these 6 variables	NUPDRS4
MDS-UPDRS Total Score	Sum of MDS-UPDRS Parts I, II, III	NUPDRS1, NUPDRS1P, NUPDRS2P, NUPDRS3

PARKINSON'S PROGRESSION MARKERS INITIATIVE

Play a Part in Parkinson's Research

Non-Motor Assessments

Benton Judgment of Line Orientation Score	Sum of BJLOT1 – BJLOT30	LINEORNT
Epworth Sleepiness Scale	Sum of ESS1 - ESS8 Subjects with ESS < 10 are "Not Sleepy" Subjects with ESS ≥ 10 are "Sleepy"	EPWORTH
GDS Raw Score	Add 1 point for each response of "No" (0) to any of the following variables: GDSSATIS, GDSGSPIR, GDSHAPPY, GDSALIVE, GDSENRGY Add 1 point for each response of "Yes" (1) to any of the following variables: GDSDROPD, GDSEMPTY, GDSBORED, GDSAFRAD, GDSHLPLS, GDSHOME, GDSMEMRY, GDSWRTLS, GDSHOPLS, GDSBETER Subjects with GDS ≥ 5 are "Depressed" Subject with GDS < 5 are "Not Depressed"	GDSSHORT
HVLT Immediate/Total Recall	Sum of HVLTRT1 - HVLTRT3	HVLT
HVLT Discrimination Recognition	HVLTREC - (HVLTFPRL + HVLTFPUN)	HVLT
HVLT Retention	HVLTRDLY / max(HVLTRT2, HVLTRT3)	HVLT
Letter Number Sequencing (LNS)	Sum of LNS1A – LNS7C	LNSPD
MOCA Total Score	Unadjusted Score = sum of MCAALTTM, MCACUBE, MCACLCKC, MCACLCKN, MCACLCKH, MCALION, MCARHINO, MCACAMEL, MCAFDS, MCABDS, MCAVIGIL, MCASER7, MCASNTNC, MCAVF, MCAABSTR, MCAREC1, MCAREC2, MCAREC3, MCAREC4, MCAREC5, MCADATE, MCAMONTH, MCAYR,	MOCA, SOCIOECO

	MCADAY, MCAPLACE, MCACITY If EDUCYRS ≤ 12 and Unadjusted Score < 30, add 1 more point to score. If EDUCYRS > 12, do not add any more points to score.	
QUIP	For Sections A - D, add 1 point if <u>either</u> question has a response of "Yes" (1): Section A: CNTRLGMB, TMGAMBLE Section B: CNTRLSEX, TMSEX Section C: CNTRLBUY, TMBUY Section D: CNTRLEAT, TMEAT For Section E, add 1 point for <u>each</u> response of "Yes" (1): TMTORACT, TMTMTACT, TMTRWD	QUIPCS
REM Sleep Behavior Disorder (RBD)	Add 1 point for <u>each</u> response of "Yes" (1) to any of the following variables: DRMVIVID, DRMAGRAC, DRMNOCTB, SLPLMBMV, SLPINJUR, DRMVERBL, DRMFIGHT, DRMUMV, DRMOBJFL, MVAWAKEN, DRMREMEM, SLPDSTRB Add 1 point if <u>any</u> of the following variables has a response of "Yes" (1): STROKE, HETRA, PARKISM, RLS, NARCLPSY, DEPRS, EPILEPSY, BRNINFM, CNSOTH If any of the previous variables are missing, then RBD score is missing. Subjects with score ≥ 5 are RBD Positive Subjects with score < 5 are RBD Negative	REMSLEEP
SCOPA-AUT Total Score	SCAU1 - SCAU25 For questions 1-21 (SCAU1 - SCAU21), add 3 points for each response of "9." Otherwise, add the number of points in response. For questions 22-25 (SCAU22 - SCAU25), add 0 points for each response of "9." Otherwise, add the number of points in response.	SCOPA

Semantic Fluency (SFT)	Sum of VLTANIM, VLTVEG, VLTFRUIT	SFT
State Trait Anxiety Total Score (STAI)	STAIAD1 - STAIAD40 Add values for the following questions: 3, 4, 6, 7, 9, 12, 13, 14, 17, 18, 22, 24, 25, 28, 29, 31, 32, 35, 37, 38, 40 Use reverse scoring for the remaining questions and add to the first score (e.g., if value = 1, add 4 points to score; if value = 2, add 3 points to score, etc).	STAI
STAI - State Subscore	STAIAD1 - STAIAD20 Add values for the following questions: 3, 4, 6, 7, 9, 12, 13, 14, 17, 18 Use reverse scoring for the values of the remaining questions through question 20 and add to the first value.	STAI
STAI - Trait Subscore	STAIAD21 - STAIAD40 Add values for the following questions: 22, 24, 25, 28, 29, 31, 32, 35, 37, 38, 40 Use reverse scoring for the values of the remaining questions and add to the first value.	STAI
UPSIT Raw Score	Sum of UPSITBK1 - UPSITBK4	UPSIT

PARKINSON'S PROGRESSION MARKERS INITIATIVE

Play a Part in Parkinson's Research

Cognitive

Mild Cognitive Impairment (MCI)	COGDECLN, FNCDTCOG, DVT_TOTAL_RECALL, DVT_RECOG_DISC_INDEX, DVS_JLO_MSSAE, DVS_LNS, DVT_SFTANIM, DVT_SDM Subject has MCI if the following 3 criteria are met: 1) Cognitive decline marked as 'Yes' (COGDECLN = '1') 2) Any 2 or more of the following cognitive tests are >1.5 SD below the standardized mean: • HVLT Total Recall (DVT_TOTAL_RECALL ≤ 35) • HVLT Recognition Discrimination (DVT_RECOG_DISC_INDEX ≤ 35) • Benton Judgment of Line Orientation (DVS_JLO_MSSAE ≤ 6) • Letter Number Sequencing (DVS_LNS ≤ 6) • Semantic Fluency Test (DVT_SFTANIM ≤ 35) • Symbol Digit Modalities (DVT_SDM ≤ 35) 3) Functional impairment marked as 'No' (FNCDTCOG = '0').	COGCATG, HVLT, LINEORNT, LNSPD, SFT, SDM

DaTSCAN

Contralateral	For Healthy Controls, no contralateral side is identified. Instead, the average of the left and right values is calculated. For PD and SWEDD subjects: Use Left value if DOMSIDE = '2' (Right) Use Right value if DOMSIDE = '1' (Left) Use the average of the left and right values if DOMSIDE = '3' (Symmetric)	SBR, PDFEAT
Ipsilateral	For Healthy Controls, no ipsilateral side is identified. Instead, the average of the left and right values is calculated. For PD and SWEDD subjects: Use Right value if DOMSIDE = '2' (Right) Use Left value if DOMSIDE = '1' (Left) Use the average of the left and right values if DOMSIDE = '3' (Symmetric)	SBR, PDFEAT

Appendix C

Variable Explanations

	A	B	C
1		VARIABLE	DESCRIPTION
2	Montreal Cognitive Assessment MoCA		Test measures different abilities: visuoconstructional, naming, attention, repetition, verbal fluency, delayed recall, orientation.
3		MCAALTTM	Alternating Trail Making
4		MCACUBE	Visuoconstructional Skills (Cube)
5		MCACLCKC	Visuoconstructional Skills (Clock Cont)
6		MCACLCKN	Visuoconstructional Skills (Clock Num)
7		MCACLCKH	Visuoconstructional Skills (Clock Hands)
8		MCALION	Naming - Lion
9		MCARHINO	Naming - Rhino
10		MCACAMEL	Naming - Camel
11		MCAFDS	Attention - Forward Digit Span
12		MCABDS	Attention - Backward Digit Span
13		MCAVIGIL	Attention - Vigilance
14		MCASER7	Attention - Serial 7s
15		MCASNTNC	Sentence Repetition
16		MCAVFNUM	Verbal Fluency - Number of words
17		MCAVF	Verbal Fluency
18		MCAABSTR	Abstraction
19		MCAREC1	Delayed Recall - Face
20		MCAREC2	Delayed Recall - Velvet
21		MCAREC3	Delayed Recall - Church
22		MCAREC4	Delayed Recall - Daisy
23		MCAREC5	Delayed Recall - Red
24		MCADATE	Orientation - Date
25		MCAMONTH	Orientation - Month
26		MCAYR	Orientation - Year
27		MCADAY	Orientation - Day
28		MCAPLACE	Orientation - Place
29		MCACITY	Orientation - City
30	Hopkins Verbal Learning Test HVLT		HVLT consists of a 12-item word list drawn from three semantic categories, presented in three learning trials that are followed by a yes/no recognition trial, containing the 12 target words and 12 distracter
31		HVLTRT1	Immediate Recall Trial 1
32		HVLTRT2	Immediate Recall Trial 2
33		HVLTRT3	Immediate Recall Trial 3
34		HVLTRDLY	Delayed Recall
35		HVLTREC	Recognition
36		HVLTFPRL	Recognition - false positives, related
37		HVLTFPUN	Recognition - false positives, unrelated
38	Benton Judgment of Line Orientation BJLO		Subjects are evaluated on 15 items per test period.
39		JLO_TOTRAW	Sum of 15 items

	A	B	C
40	Letter - Number Sequencing LNS		Subject scores 1/0 for correct/incorrect answer in each trial
41		LNS1A	Trial 1a
42		LNS1B	Trial 1b
43		LNS1C	Trial 1c
44		LNS2A	Trial 2a
45		LNS2B	Trial 2b
46		LNS2C	Trial 2c
47		LNS3A	Trial 3a
48		LNS3B	Trial 3b
49		LNS3C	Trial 3c
50		LNS4A	Trial 4a
51		LNS4B	Trial 4b
52		LNS4C	Trial 4c
53		LNS5A	Trial 5a
54		LNS5B	Trial 5b
55		LNS5C	Trial 5c
56		LNS6A	Trial 6a
57		LNS6B	Trial 6b
58		LNS6C	Trial 6c
59		LNS7A	Trial 7a
60		LNS7B	Trial 7b
61		LNS7C	Trial 7c
62	Semantic Verbal Fluency		Subject scores 1/0 for correct/incorrect answer in each category.
63		VLTANIM	Total Number of animals
64		VLTVEG	Total Number of vegetable
65		VLTFRUIT	Total Number of fruits
66	REM Sleep Disorder Questionnaire RBD		All questions on the test are YES=1/NO=0.
67		DRMVIVID	Vivid Dreams
68		DRMAGRAC	Aggressive or Action-packed dreams
69		DRMNOCTB	nocturnal behaviour
70		SLPLMBMV	move arms/legs during sleep
71		SLPINJUR	hurt bed partner
72		DRMVERBL	speaking in sleep
73		DRMFIGHT	sudden limb movements
74		DRMUMV	complex movements
75		DRMOBJFL	things fell down
76		MVAWAKEN	my movements awake me
77		DRMREMEM	remember dreams
78		SLPDSTRB	sleep is disturbed
79		STROKE	stroke
80		HETRA	head trauma

	A	B	C
81		PARKISM	parkinsonism
82		RLS	RLS
83		NARCLPSY	narcolepsy
84		DEPRS	depression
85		EPILEPSY	epilepsy
86		BRNINFM	inflammatory disease of the brain
87		CNSOTH	other
88	Geriatric Depression Scale (Short) GDS		All questions on the test are YES/NO. Each question gives 1 or 0 points on the test, a higher score means a more severe depression. Which of YES/NO gives a 1 or no depends on the question asked.
89		GDSSATIS	Basically satisfied with your life?
90		GDSDROPD	Dropped many activities and interests?
91		GDSEMPTY	Feel that your life is empty?
92		GDSBORED	Often get bored?
93		GDSGSPIR	In good spirits most of the time?
94		GDSAFRAD	Afraid something bad will happen to you?
95		GDSHAPPY	Feel happy most of the time
96		GDSHLPLS	Often feel helpless?
97		GDSHOME	Prefer to stay at home?
98		GDSMEMRY	More problems with memory than most?
99		GDSALIVE	Wonderful to be alive now?
100		GDSWRTLS	Feel pretty worthless the way you are?
101		GDSENRGY	Feel full of energy?
102		GDSHOPLS	Feel that your situation is hopeless?
103		GDSBETER	Most people are better off than you are?
104	University of Pennsylvania Smell Identification Test		Subject scores 1/0 for recognized/unrecognized scent for each of 10 scents in each booklet.
105		UPSITBK1	Score from Booklet #1
106		UPSITBK2	Score from Booklet #2
107		UPSITBK3	Score from Booklet #3
108		UPSITBK4	Score from Booklet #4
109	Epworth Sleepiness Scale ESS		All questions from this test (ESS1 to ESS8) are graded from 0 to 3, with 3 being the most severe. These questions relate to the self-rated chance of falling asleep or dozing off while performing the activities
110		ESS1	Sitting and reading
111		ESS2	Watching TV
112		ESS3	Sitting, inactive in a public place
113		ESS4	As a passenger in a car for an hour
114		ESS5	Lying down to rest in the afternoon
115		ESS6	Sitting and talking to someone
116		ESS7	Sitting quietly after lunch
117		ESS8	In a car, while stopped in traffic
118	SCOPA-AUT		SCOPA-AUT test problems with various bodily functions in the past month. A higher score means a more often problem.
119		SCAU1	SCOPA Item 1

	A	B	C
120		SCAU2	SCOPA Item 2
121		SCAU3	SCOPA Item 3
122		SCAU4	SCOPA Item 4
123		SCAU5	SCOPA Item 5
124		SCAU6	SCOPA Item 6
125		SCAU7	SCOPA Item 7
126		SCAU8	SCOPA Item 8
127		SCAU9	SCOPA Item 9
128		SCAU10	SCOPA Item 10
129		SCAU11	SCOPA Item 11
130		SCAU12	SCOPA Item 12
131		SCAU13	SCOPA Item 13
132		SCAU14	SCOPA Item 14
133		SCAU15	SCOPA Item 15
134		SCAU16	SCOPA Item 16
135		SCAU17	SCOPA Item 17
136		SCAU18	SCOPA Item 18
137		SCAU19	SCOPA Item 19
138		SCAU20	SCOPA Item 20
139		SCAU21	SCOPA Item 21
140		SCAU22	SCOPA Item 22
141		SCAU23	SCOPA Item 23
142		SCAU24	SCOPA Item 24
143		SCAU25	SCOPA Item 25
144	Symbol Digit Modalities		
145		SDMTOTAL	Symbol Digit Modalities Total Correct
146	MDS - UPDRS 1		A higher score means more severe symptoms.
147		NP1COG	COGNITIVE IMPAIRMENT, value in range 0, 1, 2, 3, 4
148		NP1HALL	HALLUCINATIONS AND PSYCHOSIS, value in range 0, 1, 2, 3, 4
149	MDS - UPDRS 2		A higher score means more severe symptoms.
150		NP2TRMR	TREMOR, value in range 0, 1, 2, 3, 4
151		NP2WALK	WALKING AND BALANCE, value in range 0, 1, 2, 3, 4
152		NP2FREZ	FREEZING, value in range 0, 1, 2, 3, 4
153	MDS - UPDRS 3		Motor examination, values in range: 0, 1, 2, 3, 4. A higher score means
154		NP3SPCH	3.1 Speech
155		NP3FACXP	3.2 Facial expression
156		NP3RIGN	3.3a Rigidity - Neck
157		NP3RIGRU	3.3b Rigidity - RUE
158		NP3RIGLU	3.3c Rigidity - LUE
159		PN3RIGRL	3.3d Rigidity - RLE
160		NP3RIGLL	3.3e Rigidity - LLE
161		NP3FTAPR	3.4a Finger Tapping Right Hand
162		NP3FTAPL	3.4b Finger Tapping Left Hand
163		NP3HMOVR	3.5a Hand movements - Right Hand
164		NP3HMOVL	3.5b Hand movements - Left Hand

	A	B	C
165		NP3PRSPR	3.6a Pronation-Supination - Right Hand
166		NP3PRSPL	3.6b Pronation-Supination - Left Hand
167		NP3TTAPR	3.7a Toe tapping - Right foot
168		NP3TTAPL	3.7b Toe tapping - Left foot
169		NP3LGAGR	3.8a Leg agility - Right leg
170		NP3LGAGL	3.8b Leg agility - Left leg
171		NP3RISNG	3.9 Arising from chair
172		NP3GAIT	3.10 Gait
173		NP3FRZGT	3.11 Freezing of gait
174		NP3PSTBL	3.12 Postural stability
175		NP3POSTR	3.13 Posture
176		NP3BRADY	3.14 Global spontaneity of movement
177		NP3PTRMR	3.15a Postural tremor - Right Hand
178		NP3PTRML	3.15b Postural tremor - Left hand
179		NP3KTRMR	3.16a Kinetic tremor - Right hand
180		NP3KTRML	3.16b Kinetic tremor - Left hand
181		NP3RTARU	3.17a Rest tremor amplitude - RUE
182		NP3RTALU	3.17b Rest tremor amplitude - LUE
183		NP3RTARL	3.17c Rest tremor amplitude - RLE
184		NP3RTALL	3.17d Rest tremor amplitude - LLE
185		NP3RTALJ	3.17e Rest tremor amplitude - Lip/jaw
186		NP3RTCON	3.18 Constancy of rest
187		NHY	3.21 Hoehn and Yahr Stage, values in range: 0, 1, 2, 3, 4, 5
188		PD_MED_USE	values in range: 0, 1, 2, 3, 4, 5, 6 ,7
189	Family History (PD)		Subject has family history of PD if any one or more of these variables =
190		BIOMOMPD	Biological Mother with PD
191		BIODADPD	Biological Father with PD
192		FULSIBPD	Full Siblings with PD
193		HAFSIBPD	Half Siblings with PD
194		MAGPARPD	Maternal Grandparents with PD
195		PAGPARPD	Paternal Grandparents with PD
196		MATAUPD	Maternal Aunts and Uncles with PD
197		PATAUPD	Paternal Aunts and Uncles with PD
198		KIDSPD	How many children with PD
199	PD Features		
200		PDDXDT	Date of Parkinson's disease diagnosis
201	Randomization table		
202		PATNO	Patient Number
203		EVENT_ID	Event Name
204		ENROLLDT	Enrollment Date
205		BIRTHDT	Birth Date
206		GENDER	Gender
207	Socio-Economics		
208		EDUCYRS	Number of years of education

Appendix D

Schedule of Activities

Schedule of Activities- Parkinson's Disease (PD) Subjects [Please see modified Schedule of Activities on page 28 for visits post January 1, 2019 or post V15 (whichever is later)]

Visit Number		SC	BL	V01	V02	V03	V04b	V05b	V06b	V07b	V08b	V09b	V10b	V11b	V12	V13	V14	V15	PW	ST	Unsch Visit
Visit Description	**Months (±30 days)**	**-45 day**	**0**	**3**	**6**	**9**	**12**	**18**	**24**	**30**	**36**	**42**	**48**	**54**	**60**	**72**	**84**	**96**	---	---	---
Written Informed Consent		X																			
Consent/Withdrawal of Consent for Future Procedures#		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
Advance Directive/Review Continuing Ability to Consent #		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X		X
Consent to Share Contact Information (FOUND in PPMI companion protocol)#		X					X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Consent to Share Contact Information with PPMI Brain and Tissue Bank/Advance Directive for PPMI Brain and Tissue Bank#†							X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Review Inclusion/Exclusion Criteria		X	X																		
Medical & Family History/Demographics		X																			
Physical Examination		X																			
Neurological Examination		X					X		X		X		X		X	X	X	X	X	Xk	Xg
Vital Signs		X	X[c]	X	X	X	X[c]	X	X[c]	X	X[c]	X	X[c]	X	X[c]	X[c]	X[c]	X[c]	X[c]	X	X
Blood Sample for DNA		X																			
Clinical Laboratory Assessments		X					X		X		X		X		X	X	X	X	X	Xk	Xg
Biomic blood sample			X[f]	X	X[f]	X	X[f]	X	X[f]	X	X[f]	X	X[f]	X	X[f]	X[f]	X[f]	X[f]	X[f]	X[f]		
PBMC blood sample			X		X			X		X		X		X							
MDS-UPDRS (including Hoehn & Yahr)h		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
Modified Schwab & England ADL		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
Clinical Diagnosis Assessment(s)		X					X		X		X		X		X	X	X	X	X	X	
MDS-UPDRS Repeat Part III/Hoehn & Yahr j							X		X		X		X		X	X	X	X	X		
Physical Activity Scale for the Elderly (PASE)							X		X		X		X		X	X	X	X	X		
Hopkins Verbal Learning Test – Revised			X				X		X		X		X		X	X	X	X	X	X[k]	
Benton Judgment of Line Orientation			X				X		X		X		X		X	X	X	X	X	X[k]	
Semantic Fluency			X				X		X		X		X		X	X	X	X	X	X[k]	
Letter Number Sequencing			X				X		X		X		X		X	X	X	X	X	X[k]	
Symbol Digit Modalities Test			X				X		X		X		X		X	X	X	X	X	X[k]	
Montreal Cognitive Assessment (MoCA)		X					X		X		X		X		X	X	X	X	X	X[k]	
Epworth Sleepiness Scale			X		X		X		X		X		X		X	X	X	X	X	X	
REM Sleep Behavior Questionnaire			X		X		X		X		X		X		X	X	X	X	X	X	
Geriatric Depression Scale (GDS-15)			X		X		X		X		X		X		X	X	X	X	X	X	
State-Trait Anxiety Inventory for Adults			X		X		X		X		X		X		X	X	X	X	X	X	
QUIP			X		X		X		X		X		X		X	X	X	X	X	X	
SCOPA-AUT			X		X		X		X		X		X		X	X	X	X	X	X	
Cognitive Categorization			X		X		X		X		X		X		X	X	X	X	X	X[k]	

Olfactory Testing (UPSIT)		X																	
MRI (structural)		X																	
MRI (DTI)e		X				X		X				X						^X		
DAT imaging[m,o]	X					X		X				X						^X	X[k]	
[18F] florbetaben PET imaging (see companion protocol)s								X		(X)		(X)		(X)	(X)	(X)	(X)	(X) +		
VMAT-2 imaging[m,o, r] (see companion protocol)	X[p]					X		X				X						^X	X[k]	
Lumbar puncture (CSF collection)		X		X		X		X		X		X		X	X	X	X	*X	X[k]	
Adverse Eventsa	X	X		X		X		X		X		X		X	X	X	X	X	X	
Current Medical Conditions Review		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Concomitant Medication Review	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Skin Biopsy t (see companion protocol)		X (+ follow up call within 7-10 days of biopsy)																		X
Blood draw[t]																				X
Sensor-PPMI activities (see companion protocol)	X																			

[a] Adverse events assessed at the visit and by phone 7 to 10 days following LP and/or DaTSCAN injection and following skin biopsy and/or [^{18}F] florbetaben PET imaging (for participating sites).
[b] Telephone consent will occur at Months 15, 21, 27, 33, 39, 45, 51 and 57.
[c] Height and weight also collected.
[e] Diffusion tensor MRI scan and resting state sequences conducted at selected sites.
[f] Biomic urine sample also collected.
[g] Conduct as clinically indicated – see protocol Sect. 5.3.18.
[h] Part IV once subject has started PD medication.
[j] Repeat assessment 1hr post treatment for subjects on levodopa or dopamine agonist.
[k] Not conducted depending on when ST visit completed – see protocol Sect. 5.3.20.
[m] DAT completed at all sites except Australia; VMAT completed in Australia and selected U.S. sites.

[†] Intent to Donate documentation completed with PPMI Brain and Tissue Bank study team or local PPMI site based on accessibility, local regulations and availability of infrastructure. For subjects who have not yet been approached concerning the Brain and Tissue Bank, please review subject interest in providing permission to forward their contact information to the PPMI Brain and Tissue Bank study team at their next visit (including interim visits). For subjects who may have already been approached concerning the Brain and Tissue Bank, but were undecided concerning participation or who have not yet signed an Intent to Donate form- please review Brain and Tissue Bank with subject at next annual visit (sites may also review this at interim visits if site deems appropriate).

[o] Urine (or serum) pregnancy test prior to injection for women of childbearing potential.
[p] Serum pregnancy test prior to injection day for women of childbearing potential; ECG for all subjects.
[r] Subjects enrolled in U.S. may agree to initial VMAT imaging scan at YR01 rather than Screening.[s] [^{18}F]
[s]florbetaben PET imaging at selected sites (see companion protocol) at either Month 24, 36, 48, 60, 72, 84 or 96[t] Skin biopsy for participating sites completed at any routine or unscheduled visit post amendment approval; for sites collecting blood samples as part of companion study, blood draw completed at unscheduled visit post amendment approval.
[t] Skin biopsy for participating sites completed at any routine or unscheduled visit post amendment approval; for sites collecting blood samples as part of companion study, blood draw completed at unscheduled visit or interim visit post amendment approval. (sites completing collection of blood sample at interim visit must forego collection of PBMCs and biomic blood samples at interim visit)

ST = Symptomatic Therapy
PW = Premature Withdrawal (*if not done in last 3 mths; ^if not done in last 12 mths; [+]if subject has not completed scan and is willing to complete at PW)
[#] For active subjects, begin at next visit post amendment approval and consent

Appendix E

Confusion Matrices and ROC Curves

This appendix shows the confusion matrices and ROC curves for all models presented in Chapter 5 - *Results*.

E.1 Baseline Approach

E.1.1 Logistic Regression

	True	False
Negative	57	32
Positive	22	9

Table E.1: Baseline Approach: Logistic Regression Confusion Matrix

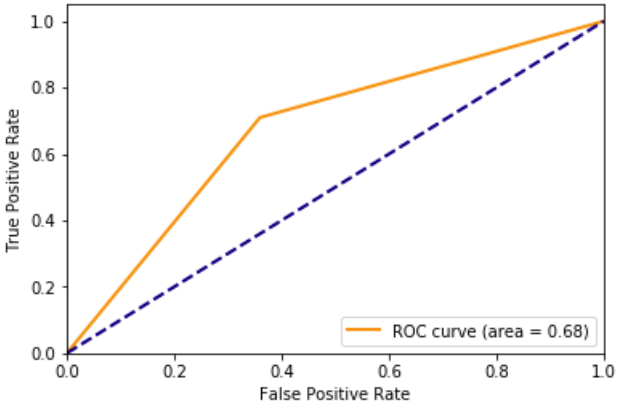


Figure E.1: Baseline Approach: Logistic Regression ROC Curve

E.1.2 SVM-linear

	True	False
Negative	63	26
Positive	19	12

Table E.2: Baseline Approach: SVM-Linear Confusion Matrix

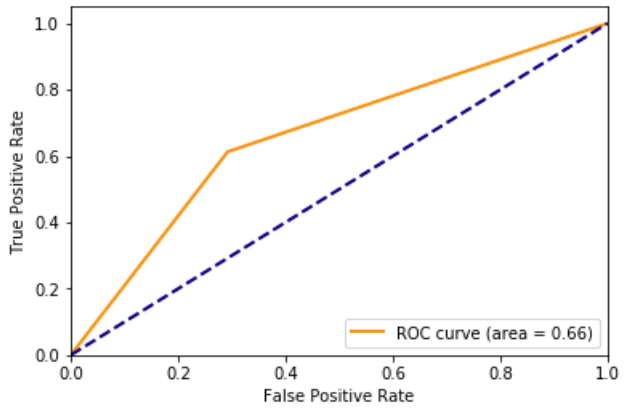


Figure E.2: Baseline Approach: SVM-Linear ROC Curve

E.1.3 SVM-polynomial

	True	False
Negative	64	25
Positive	18	13

Table E.3: Baseline Approach: SVM-Polynomial Confusion Matrix

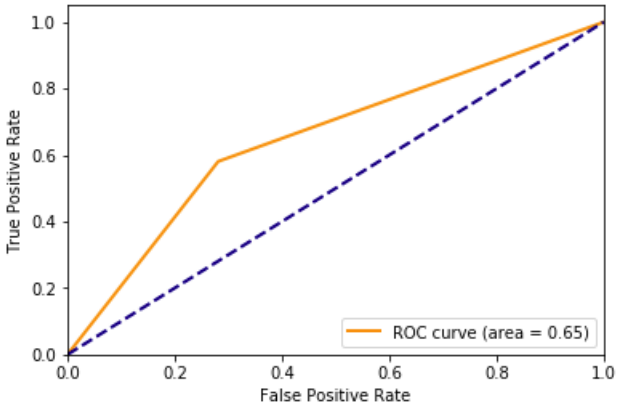


Figure E.3: Baseline Approach: SVM-Polynomial ROC Curve

E.1.4 SVM-radial

	True	False
Negative	64	25
Positive	16	15

Table E.4: Baseline Approach: SVM-Radial Confusion Matrix

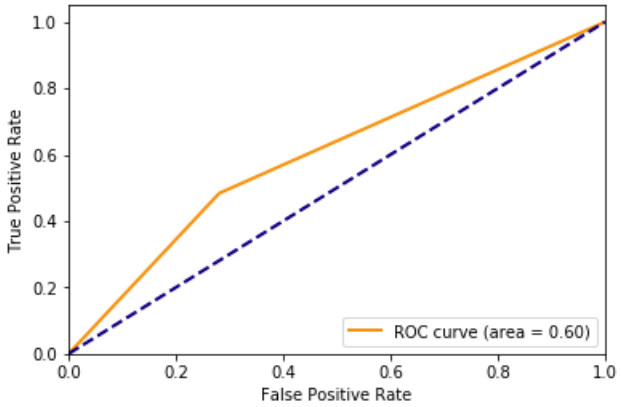


Figure E.4: Baseline Approach: SVM-Radial ROC Curve

E.1.5 Decision Tree

	True	False
Negative	51	38
Positive	21	10

Table E.5: Baseline Approach: Decision Tree Confusion Matrix

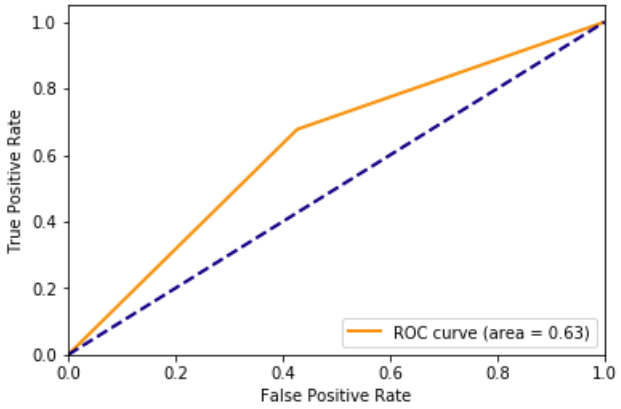


Figure E.5: Baseline Approach: Decision Tree ROC Curve

E.1.6 Random Forest

	True	False
Negative	71	18
Positive	16	15

Table E.6: Baseline Approach: Random Forest Confusion Matrix

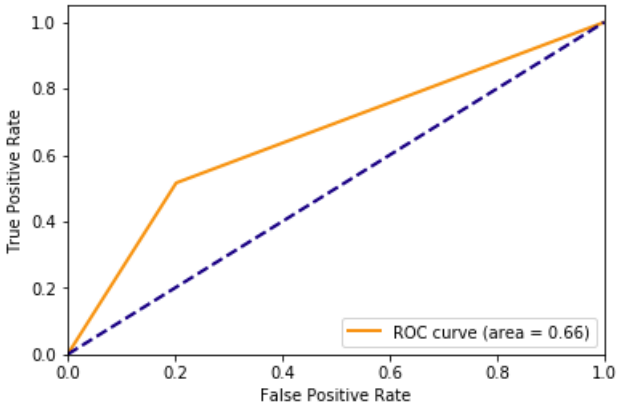


Figure E.6: Baseline Approach: Random Forest ROC Curve

E.1.7 Boosted Trees

	True	False
Negative	63	26
Positive	17	14

Table E.7: Baseline Approach: Boosted Trees Confusion Matrix

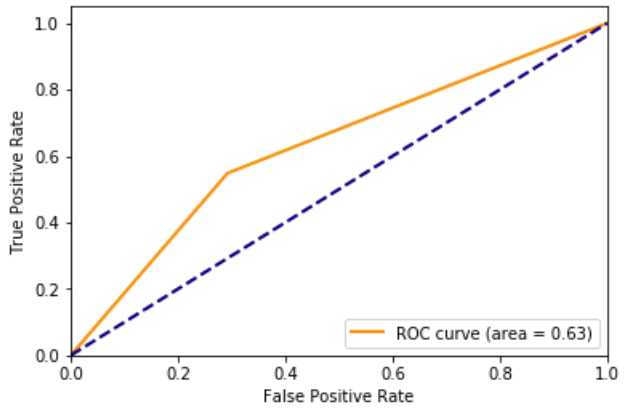


Figure E.7: Baseline Approach: Boosted Trees ROC Curve

E.2 Derived Statistics Approach

E.2.1 Logistic Regression

	True	False
Negative	58	23
Positive	17	8

Table E.8: Derived Statistics Approach: Logistic Regression Confusion Matrix

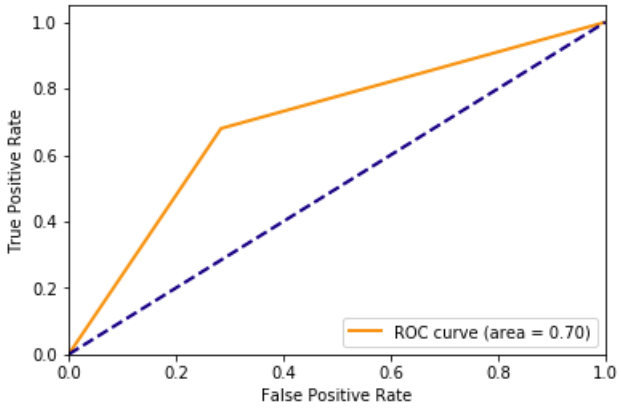


Figure E.8: Derived Statistics Approach: Logistic Regression ROC Curve

E.2.2 SVM-linear

	True	False
Negative	70	11
Positive	18	7

Table E.9: Derived Statistics Approach: SVM-Linear Confusion Matrix

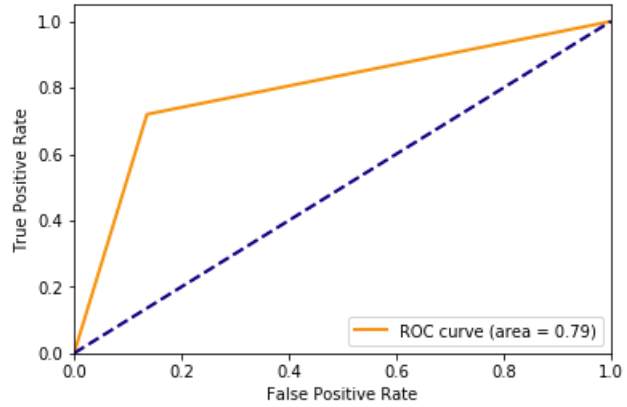


Figure E.9: Derived Statistics Approach: SVM-Linear ROC Curve

E.2.3 SVM-polynomial

	True	False
Negative	60	21
Positive	19	6

Table E.10: Derived Statistics Approach: SVM-Polynomial Confusion Matrix

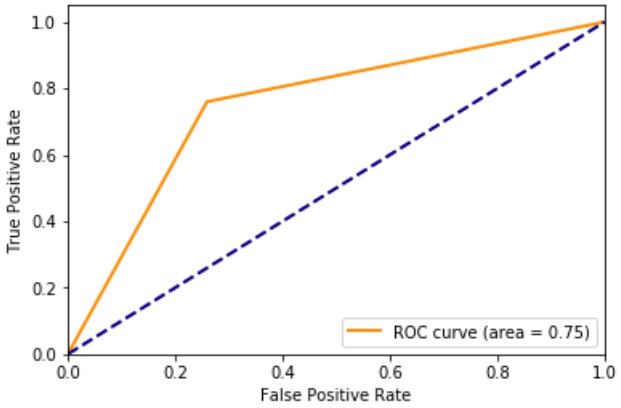


Figure E.10: Derived Statistics Approach: SVM-Polynomial ROC Curve

E.2.4 SVM-radial

	True	False
Negative	59	22
Positive	19	6

Table E.11: Derived Statistics Approach: SVM-Radial Confusion Matrix

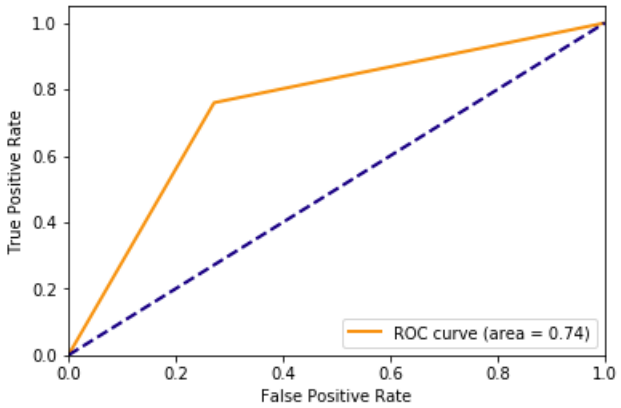


Figure E.11: Derived Statistics Approach: SVM-Radial ROC Curve

E.2.5 Decision Tree

	True	False
Negative	58	23
Positive	13	12

Table E.12: Derived Statistics Approach: Decision Tree Confusion Matrix

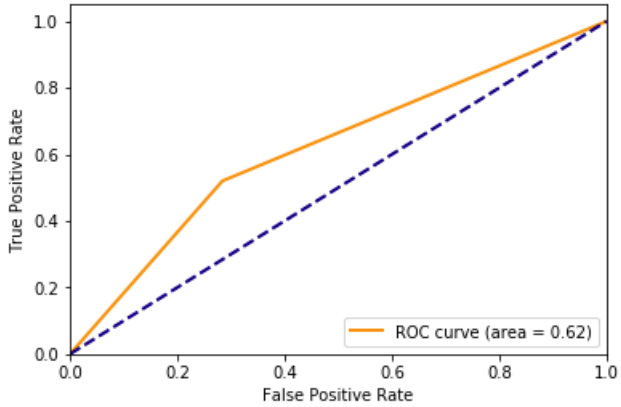


Figure E.12: Derived Statistics Approach: Decision Tree ROC Curve

E.2.6 Random Forest

	True	False
Negative	58	23
Positive	16	9

Table E.13: Derived Statistics Approach: Random Forest Confusion Matrix

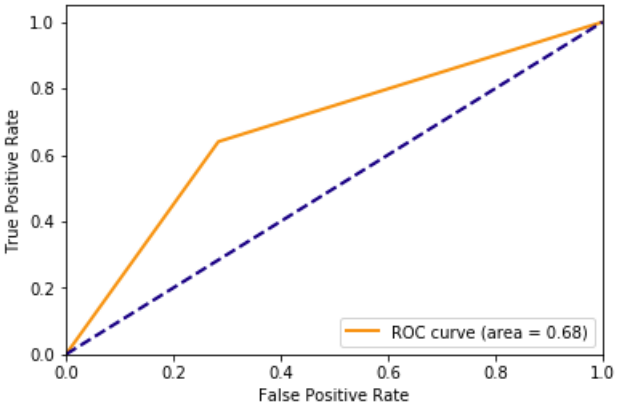


Figure E.13: Derived Statistics Approach: Random Forest ROC Curve

E.2.7 Boosted Trees

	True	False
Negative	56	25
Positive	15	10

Table E.14: Derived Statistics Approach: Boosted Trees Confusion Matrix

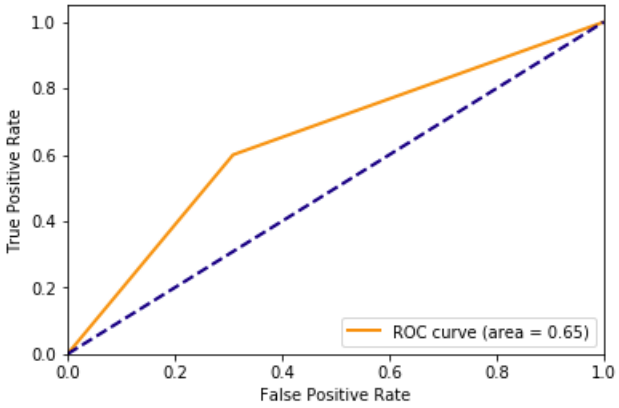


Figure E.14: Derived Statistics Approach: Boosted Trees ROC Curve

E.3 Deep Learning Approach

E.3.1 LSTM

We can not show a confusion matrix and ROC curve that exactly match the test results presented in 5.1.3 as these test results are the average results from multiple LSTM

networks. This section therefore shows the result of a single represenative LSTM model with these test results:

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Test	0.662	0.385	0.476	0.729	0.426	0.603	0.193

Table E.15: Single LSTM network: Test Results

	True	False
Negative	43	16
Positive	10	11

Table E.16: Deep Learning Approach: LSTM Confusion Matrix

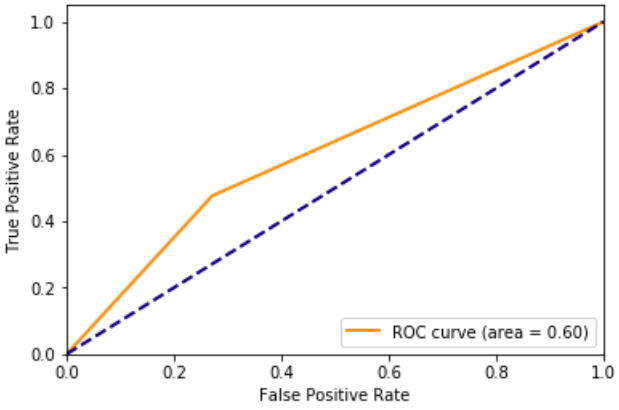


Figure E.15: Deep Learning Approach: LSTM ROC Curve

Appendix F

Prediction Performance vs. Year

This appendix shows the sensitivity of our models on predicting psychosis for subjects that experience their first psychotic symptoms after n years.

F.1 Baseline Approach

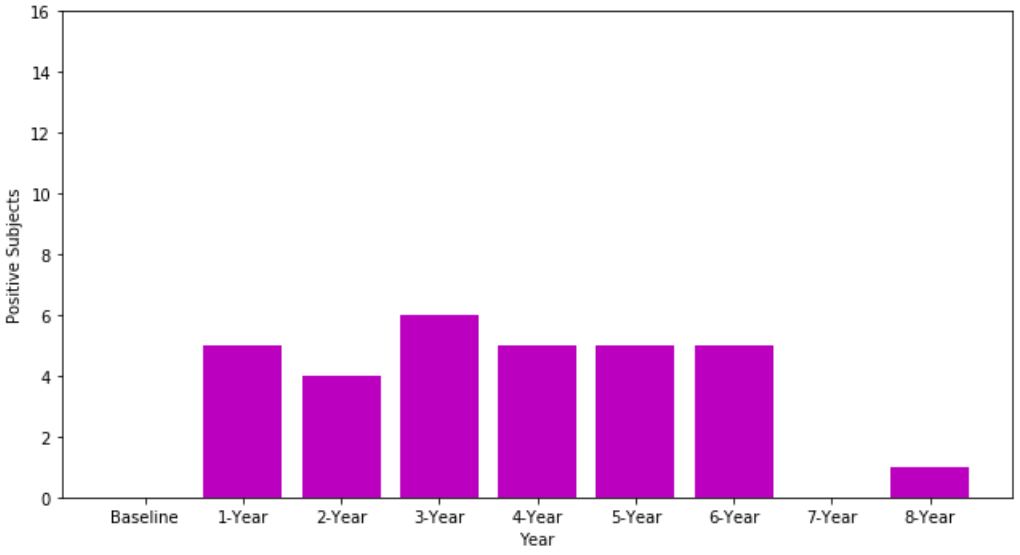


Figure F.1: Baseline Approach: Number of subjects who show first symptoms of psychosis per year in the test set

F.1.1 Logistic Regression

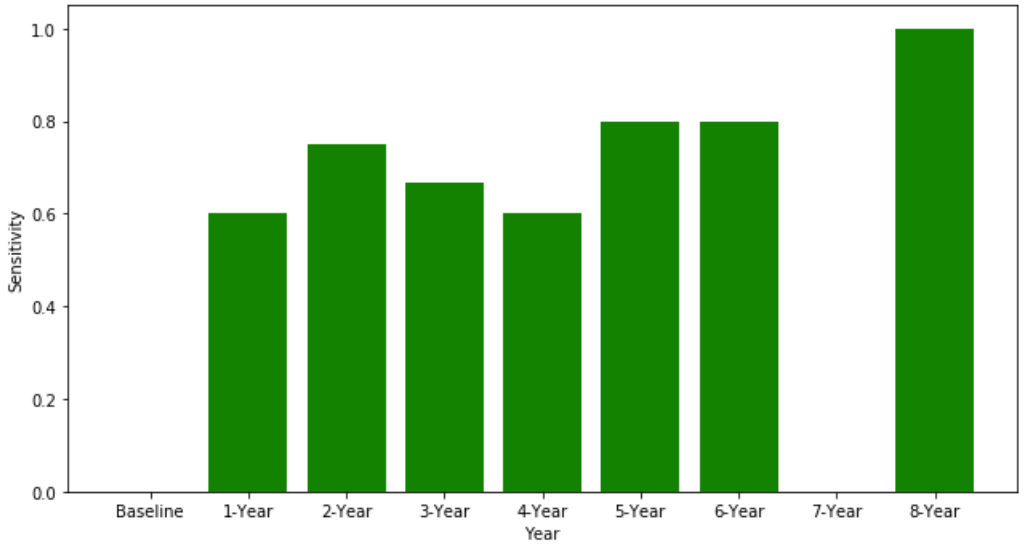


Figure F.2: Baseline Approach: Sensitivity per year - Logistic Regression

F.1.2 SVM-linear

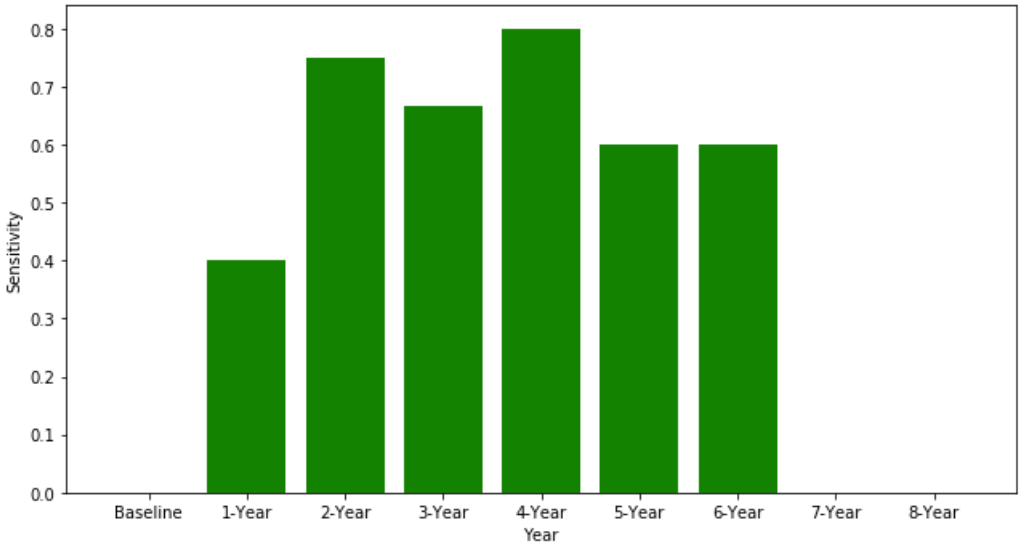


Figure F.3: Baseline Approach: Sensitivity per year - SVM-linear

F.1.3 SVM-polynomial

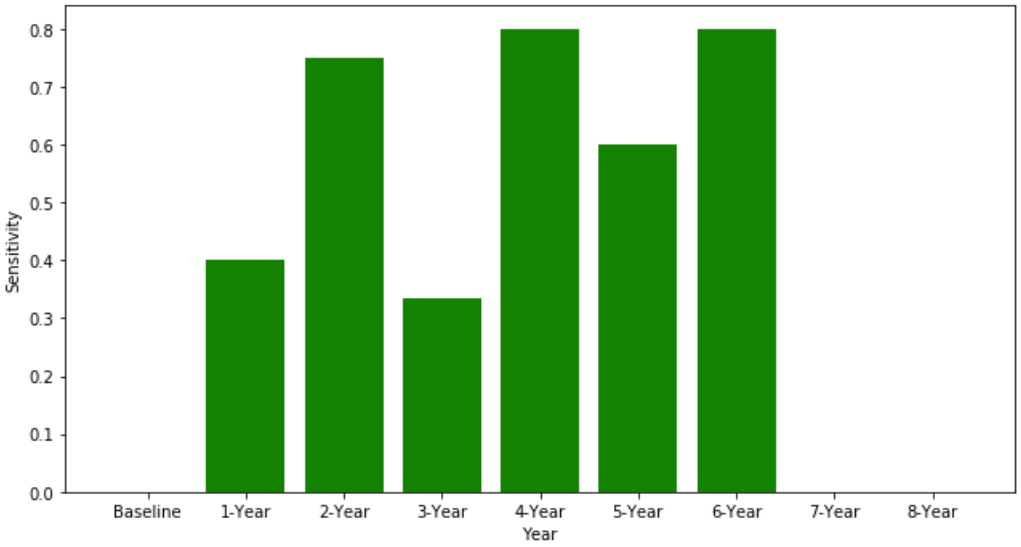


Figure F.4: Baseline Approach: Sensitivity per year - SVM-poly

F.1.4 SVM-radial

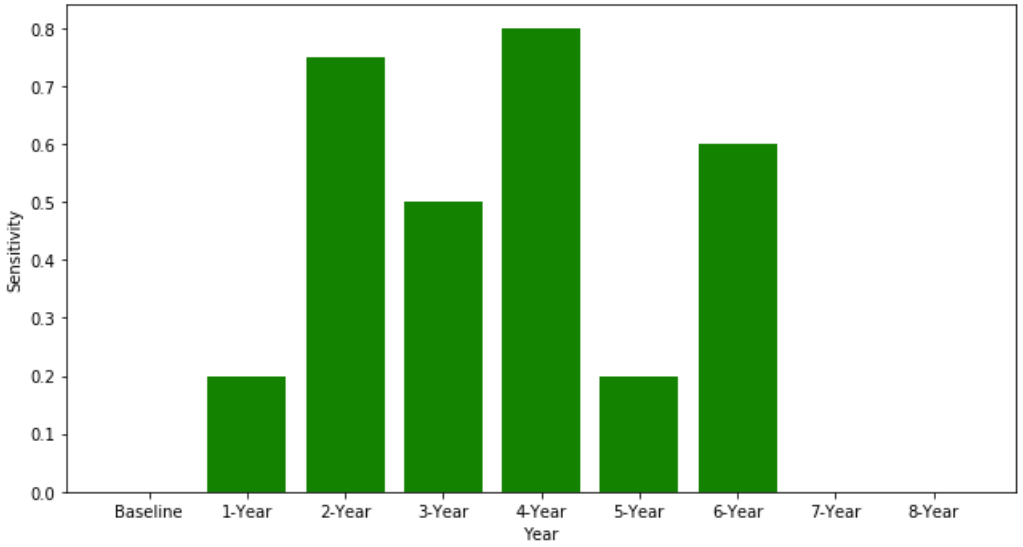


Figure F.5: Baseline Approach: Sensitivity per year - SVM-radial

F.1.5 Decision Tree

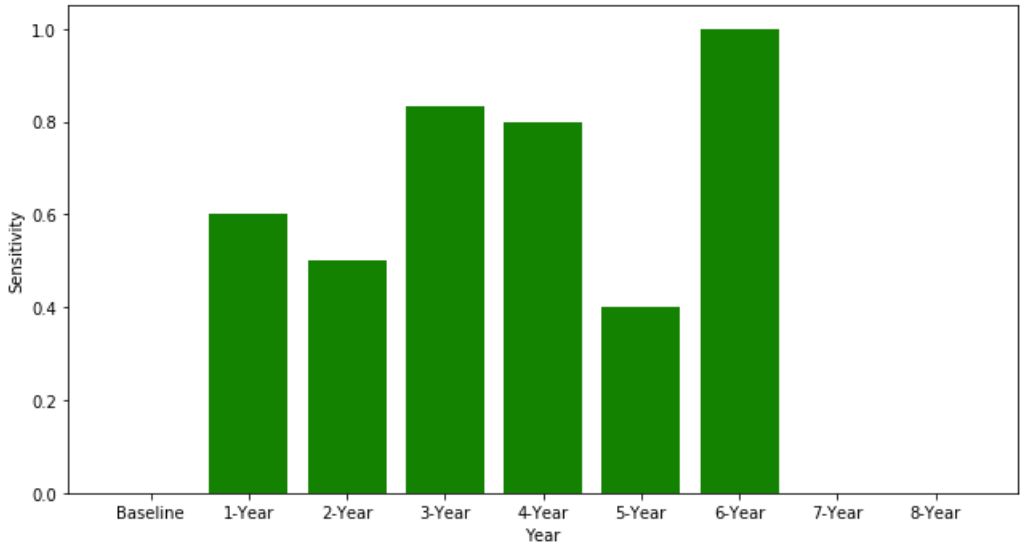


Figure F.6: Baseline Approach: Sensitivity per year - Decision Tree

F.1.6 Random Forest

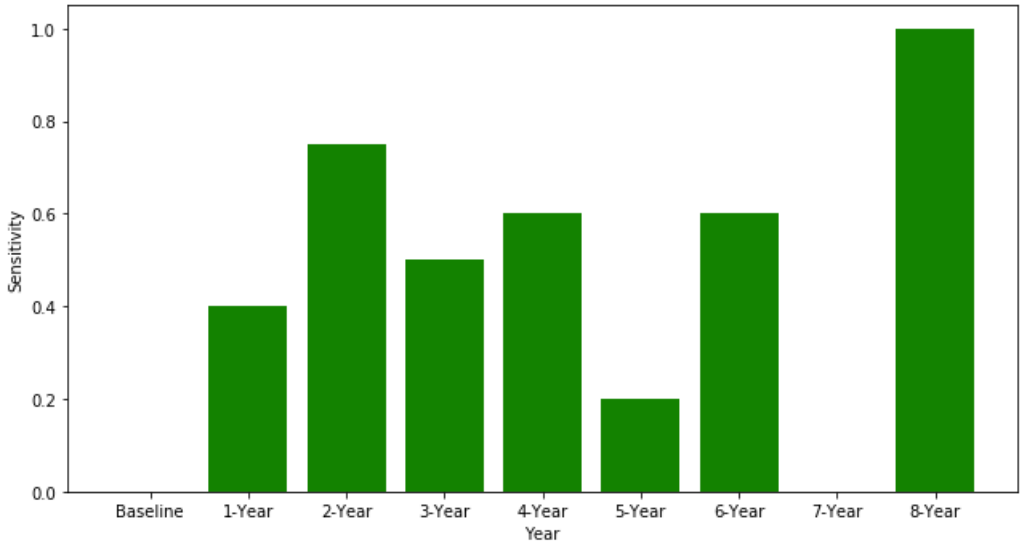


Figure F.7: Baseline Approach: Sensitivity per year - Random Forest

F.1.7 Boosted Trees

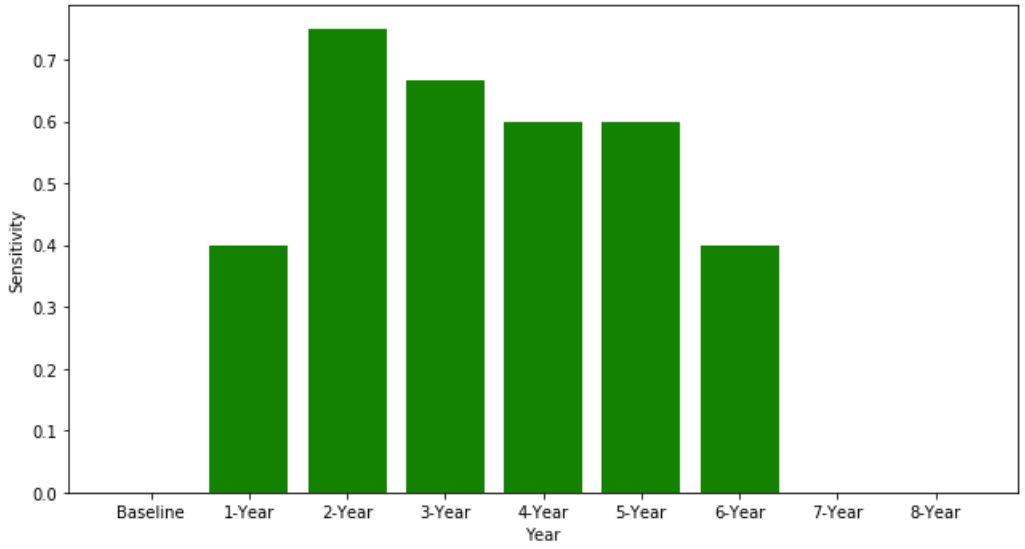


Figure F.8: Baseline Approach: Sensitivity per year - Boosted Trees

F.2 Derived Statistics Approach

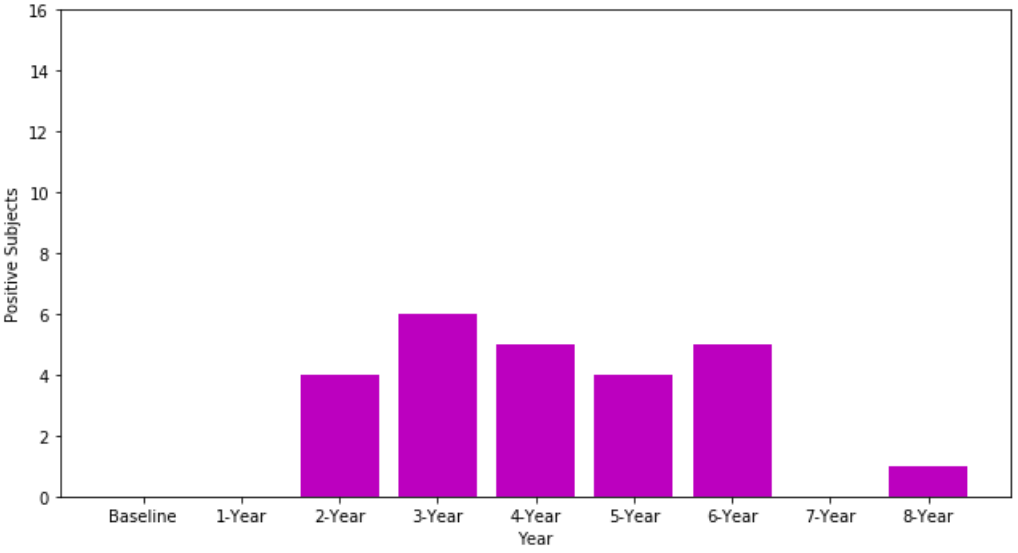


Figure F.9: Derived Statistics Approach: Number of subjects who show first symptoms of psychosis per year in the test set

F.2.1 Logistic Regression

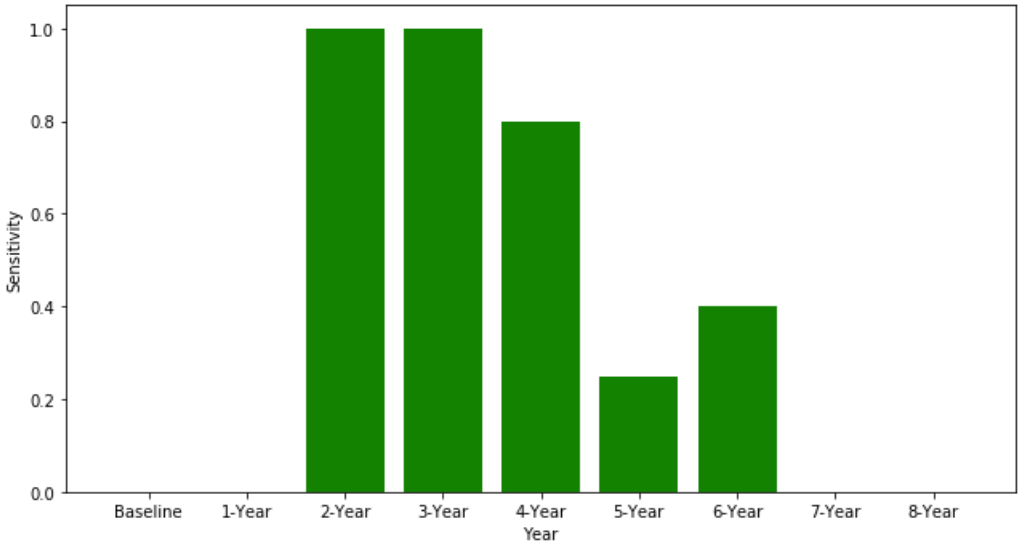


Figure F.10: Derived Statistics Approach: Sensitivity per year - Logistic Regression

F.2.2 SVM-linear

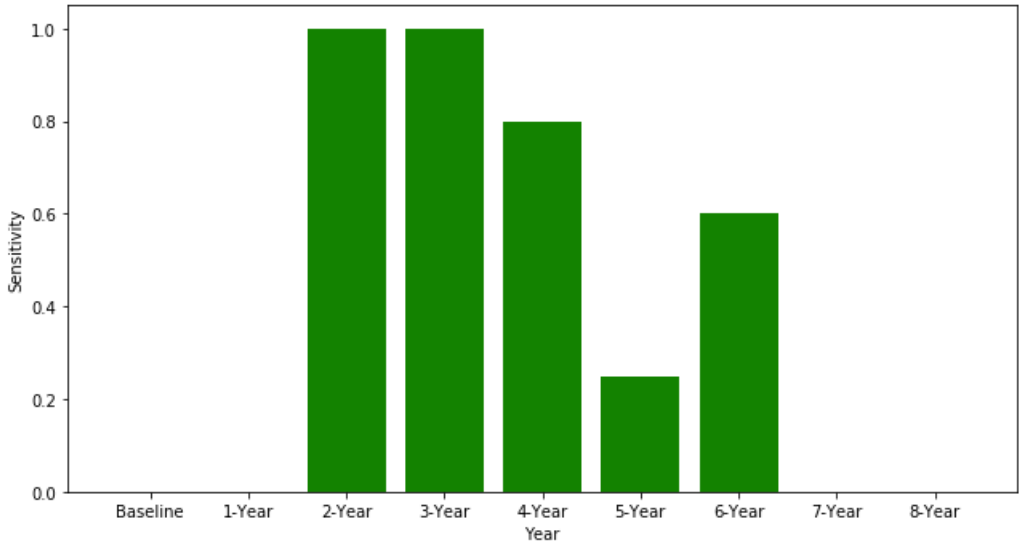


Figure F.11: Derived Statistics Approach: Sensitivity per year - SVM-linear

F.2.3 SVM-polynomial

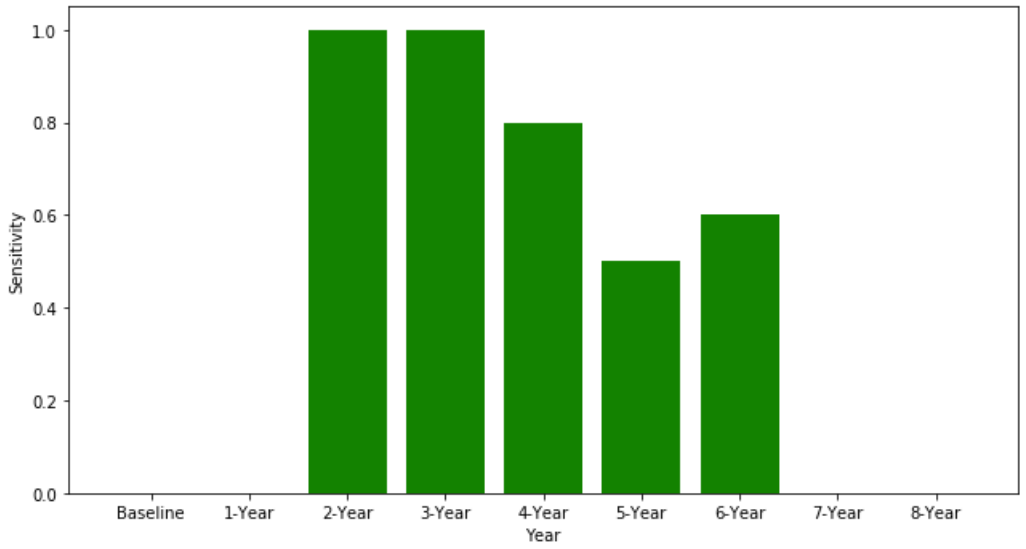


Figure F.12: Derived Statistics Approach: Sensitivity per year - SVM-poly

F.2.4 SVM-radial

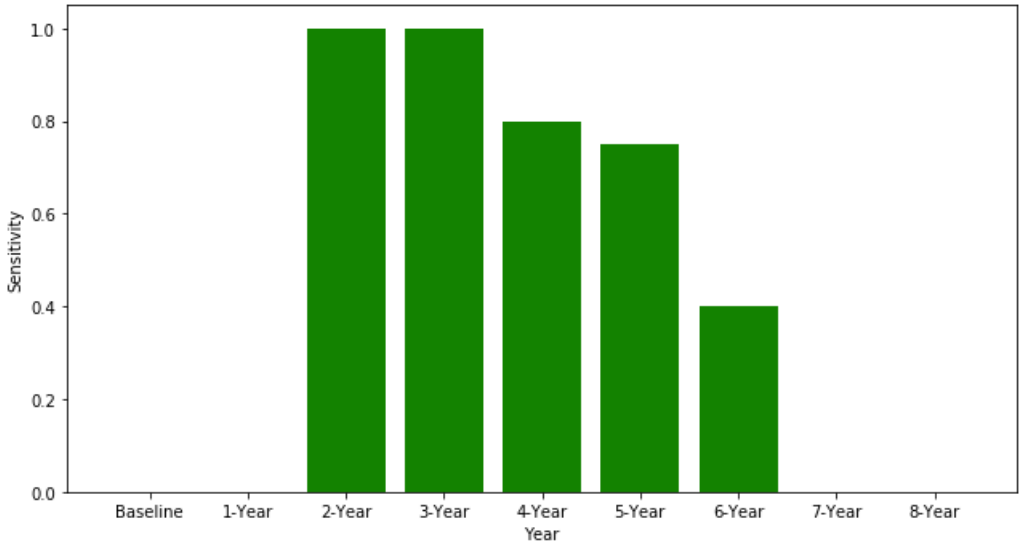


Figure F.13: Derived Statistics Approach: Sensitivity per year - SVM-radial

F.2.5 Decision Tree

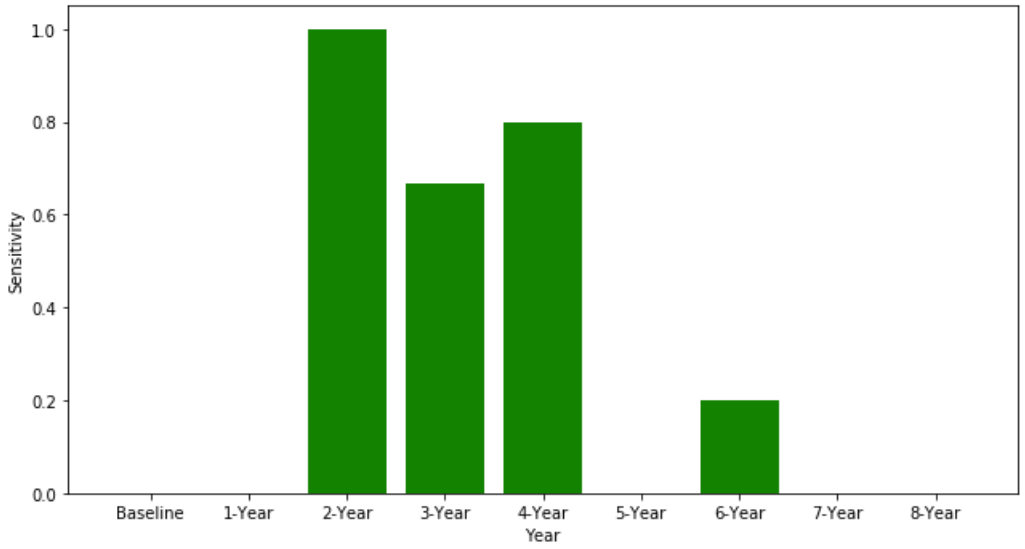


Figure F.14: Derived Statistics Approach: Sensitivity per year - Decision Tree

F.2.6 Random Forest

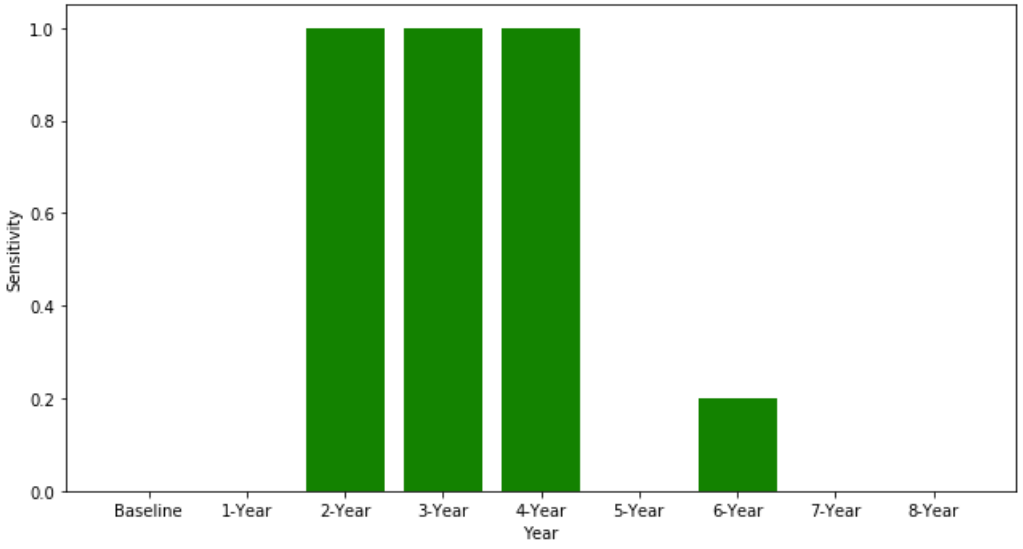


Figure F.15: Derived Statistics Approach: Sensitivity per year - Random Forest

F.2.7 Boosted Trees

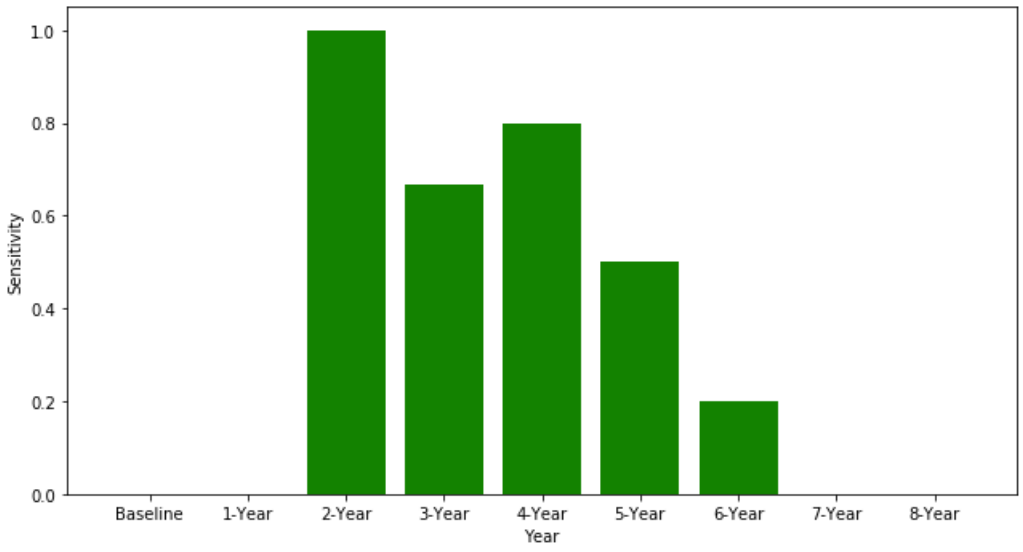


Figure F.16: Derived Statistics Approach: Sensitivity per year - Boosted Trees

F.3 Deep Learning Approach

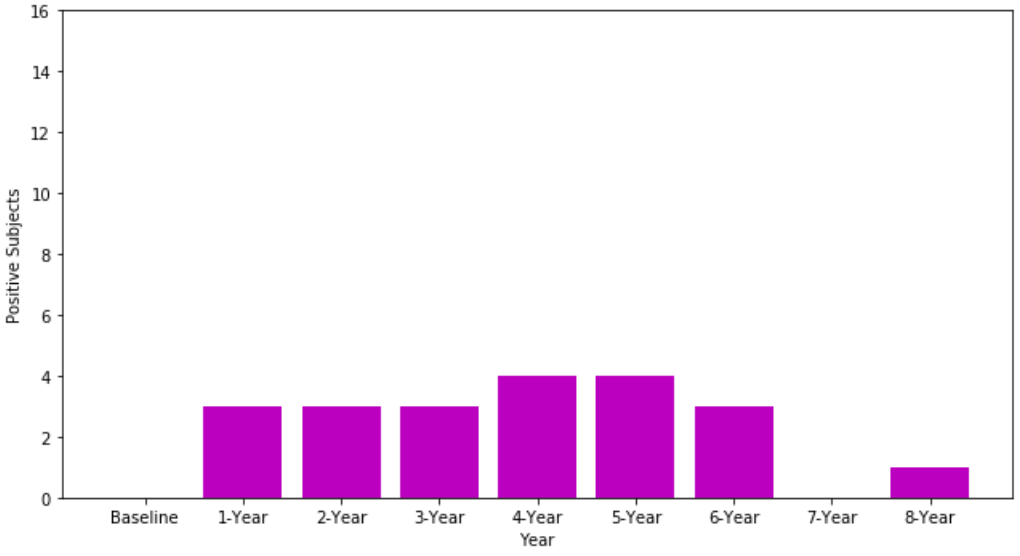


Figure F.17: Deep Learning Approach: Number of subjects who show first symptoms of psychosis per year in the test set

F.3.1 LSTM

We can not show a sensitivity vs year plot that exactly match the test results presented in 5.1.3 as these test results are the average results from multiple LSTM networks. This

section therefore shows the result of a single represenative LSTM model with these test results:

Model	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Test	0.662	0.385	0.476	0.729	0.426	0.603	0.193

Table F.1: Single LSTM network: Test Results

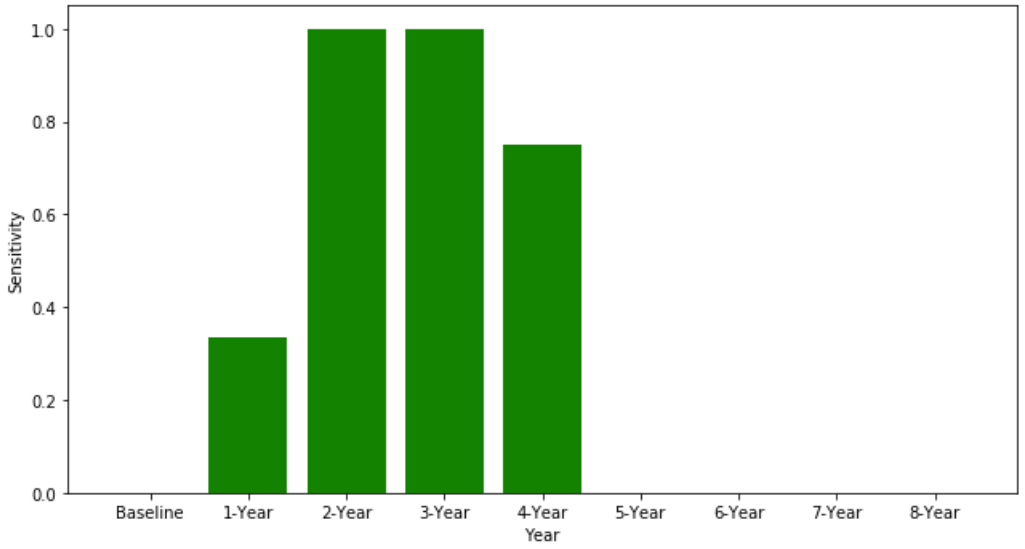


Figure F.18: Deep Learning Approach: Sensitivity per year - LSTM

Appendix G

Improving Results: HEOM

G.1 Results

In this Appendix we will try to improve results from our best performing models by using HEOM missing value imputation instead of mode/mean in the preprocessing step. We will test this by using a pre-processed data set with HEOM missing value imputation to build our two best performing models from the baseline approach and our two best performing models from the derived approach. From the baseline approach we will do logistic regression and SVM-linear. From the derived approach we will do SVM-linear and SVM-polynomial. We will compare results from these models to results from the same models in chapter 5 that use mode/mean missing value imputation.

G.1.1 Baseline Approach

The following preproccesing steps are applied on the data set for this approach:

- HEOM missing value imputation
- Feature reduction through total scores
- Train/test set ratio of 70/30
- Prediction based on baseline data

Logistic Regression

The tuning rounds used to find the final tuned model is summarized in Table G.1:

Tuning Round	Random Search 1	Random Search 2	Grid search 1	Grid search 2
Parameters	C,L	C,L	C,L	C,L
Range/Grid C	[0,10]	[0,10]	0.33,0.01	0.0001,0.001, 0.01,0.1
Range/Grid L	L1	L2	L1,L2	L2
Chosen C	0.33	0.01	0.01	0.01
Chosen L	L1	L2	L2	L2

Table G.1: Baseline Approach - HEOM: Logistic Regression Tuning. SS = Step Size

The chosen hyper-parameters from grid search is L2-norm for regularization penalty and 0.01 for regularization strength. The results from nested cross validation, training and test set is summarized in Table G.2.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.637	0.372	0.552	0.667	0.440	0.609	0.200
Nested CV - std	0.125	0.162	0.220	0.111	0.177	0.150	0.276
Training results	0.692	0.426	0.606	0.721	0.500	0.663	0.296
Test results	0.658	0.407	0.710	0.640	0.518	0.675	0.308

Table G.2: Baseline Approach - HEOM: Logistic Regression Results

SVM - Linear Kernel

The tuning rounds used to find the final tuned model for a linear kernel is summarized in Table G.3:

Tuning Round	Random Search 1	Grid search 1
Parameters	C	C
Range/Grid C	[0,10]	[1,12], SS = 1
Chosen C	6.24	4

Table G.3: Baseline Approach - HEOM: SVM Linear Kernel Tuning. SS = Step Size

The chosen hyper-parameters from grid search is a regularization strength of 4 for a linear kernel. The results from nested cross validation, training and test set is summarized in Table G.4

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.627	0.354	0.562	0.649	0.429	0.606	0.190
Nested CV - std	0.064	0.061	0.164	0.100	0.086	0.070	0.123
Training results	0.724	0.470	0.662	0.745	0.550	0.704	0.370
Test results	0.683	0.422	0.613	0.708	0.500	0.660	0.290

Table G.4: Baseline Approach - HEOM: SVM Linear Kernel Results

G.1.2 Derived Statistics Approach

The following preproccesing steps are applied on the data set for this approach:

- HEOM missing value imputation
- Feature reduction through total scores
- Train/test set ratio of 70:30
- Prediction based on derived statistics

SVM - Linear Kernel

The tuning rounds used to find the final tuned model for a linear kernel is summarized in Table G.5:

Tuning Round	Random Search 1	Grid search 1
Parameters	C	C
Range/Grid C	[0,10]	[0.0001,0.001,0.01,0.1]
Chosen C	0.01	0.01

Table G.5: Derived Approach - HEOM: SVM Linear Kernel Tuning

The chosen hyper-parameter from grid search is a regularization strength of 0.01 for a linear kernel. The results from nested cross validation, training and test set is summarized in Table G.6

SVM - Polynomial Kernel

The tuning rounds used to find the final tuned model for a polynomial kernel is summarized in Table G.7.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.710	0.291	0.400	0.800	0.331	0.600	0.178
Nested CV - std	0.069	0.208	0.309	0.115	0.235	0.121	0.227
Training results	0.786	0.532	0.724	0.805	0.613	0.765	0.481
Test results	0.708	0.432	0.760	0.691	0.551	0.726	0.389

Table G.6: Derived Approach - HEOM: SVM Linear Kernel Results

Tuning Round	Random Search 1	Grid search 1
Parameters	C,γ,d,r	Cγ,d,r
Range/Grid C	[0,10]	[0.10,0.80], SS = 0.05
Range/Grid γ	(0,10]	[0.01,0.15], SS = 0.01
Range/Grid d	[1,5]	[1,2], SS = 0.1
Range/grid r	[0,10]	[0,8], SS = 1
Chosen C	0.46	0.30
Chosen γ	0.12	0.05
Chosen d	1	1
Chosen r	4	0

Table G.7: Derived Approach - HEOM: SVM Polynomial Kernel Tuning. SS = Step Size

The chosen hyper-parameters are;

- Regularization strength: 0.30
- Gamma scaling: 0.05
- Polynomial degree: 1
- Independent term: 0

The results from nested cross validation, training and test set for this set of hyper-parameters is summarized in Table G.8

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.706	0.383	0.537	0.758	0.440	0.647	0.261
Nested CV - std	0.086	0.182	0.279	0.067	0.210	0.146	0.271
Training results	0.758	0.487	0.638	0.795	0.552	0.716	0.397
Test results	0.736	0.462	0.720	0.741	0.562	0.730	0.406

Table G.8: Derived Approach - HEOM: SVM Polynomial Kernel Results

G.2 Comparison

We can compare the test results from chapter 5 to the results presented in this appendix. The comparisons can be viewed in Tables G.9, G.10, G.11 and G.12.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
MM	0.658	0.407	0.710	0.640	0.518	0.675	0.308
HEOM	0.658	0.407	0.710	0.640	0.518	0.675	0.308

Table G.9: MM VS HEOM: Baseline Approach Logistic Regression Test Results

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
MM BL SVM-Linear	0.683	0.422	0.613	0.708	0.500	0.660	0.290
HEOM BL SVM-Linear	0.683	0.422	0.613	0.708	0.500	0.660	0.290

Table G.10: MM VS HEOM: Baseline Approach SVM-Linear Test Results

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
MM DE SVM-Linear	0.830	0.621	0.720	0.864	0.667	0.792	0.556
HEOM DE SVM-Linear	0.708	0.432	0.760	0.691	0.551	0.726	0.389

Table G.11: MM VS HEOM: Derived Approach SVM-Linear Test Results

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
MM DE SVM-Poly	0.745	0.475	0.760	0.741	0.585	0.750	0.439
HEOM DE SVM-Poly	0.736	0.462	0.720	0.741	0.562	0.730	0.406

Table G.12: MM VS HEOM: Derived Approach SVM-Polynomial Test Results

The baseline approach test results are identical between Mode/Mean and HEOM missing value imputation. The derived statistics approach test results are different, with mode/mean performing slightly better on most metrics. We can conclude from these results that the method of missing value imputation should have minimal impact on model performance. Mode/mean missing value imputation might perform slightly better than HEOM for this data set.

Appendix H

Improving Results: PCA

H.1 Results

In this Appendix we will try to improve results from our best performing models by using PCA instead of total scores for feature reduction in the preprocessing step. We will test this by using a preprocessed data set with PCA missing feature reduction to build our two best performing models from the baseline approach and our two best performing models from the derived statistics approach. From the baseline approach we will do logistic regression and SVM-linear. From the derived statistics approach we will do SVM-linear and SVM-polynomial. We will compare results from these models to results from the same models in chapter 5 that use total scores as feature reduction.

We keep enough PCA dimensions to capture around 65% of the variance of the original data set. This results in 31 PCA dimensions. This is the same number of features we get by reducing with total scores. That should make results comparable between this appendix and chapter 5.

H.1.1 Baseline Approach

The following preproccesing steps are applied on the data set for this approach:

- mean/mode missing value imputation
- Feature reduction through PCA
- Train/test set ratio of 70/30
- Prediction based on baseline data

Logistic Regression

The tuning rounds used to find the final tuned model is summarized in Table H.1:

Tuning Round	Random Search 1	Random Search 2	Grid search 1	Grid search 2
Parameters	C,L	C,L	C,L	C,L
Range/Grid C	[0,10]	[0,10]	0.16,0.01	0.0001,0.001, 0.01,0.1
Range/Grid L	L1	L2	L1,L2	L2
Chosen C	0.16	0.01	0.01	0.01
Chosen L	L1	L2	L2	L2

Table H.1: Baseline Approach - PCA: Logistic Regression Tuning

The chosen hyper-parameters from grid search is L2-norm for regularization penalty and 0.01 for regularization strength. The results from nested cross validation, training and test set is summarized in Table H.2.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.633	0.374	0.596	0.648	0.453	0.622	0.220
Nested CV - std	0.127	0.139	0.234	0.129	0.165	0.148	0.265
Training results	0.692	0.431	0.662	0.702	0.522	0.682	0.325
Test results	0.667	0.400	0.581	0.697	0.474	0.639	0.251

Table H.2: Baseline Approach - PCA: Logistic Regression Results

SVM - Linear Kernel

The tuning rounds used to find the final tuned model for a linear kernel is summarized in Table H.3:

Tuning Round	Random Search 1	Grid search 1
Parameters	C	C
Range/Grid C	[0,10]	[0.01,0.30], SS = 0.01
Chosen C	0.12	0.02

Table H.3: Baseline Approach - PCA: SVM Linear Kernel Tuning. SS = Step Size

The chosen hyper-parameters from grid search is a regularization strength of 4 for a linear kernel. The results from nested cross validation, training and test set is summarized in Table H.4

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.638	0.370	0.564	0.663	0.442	0.614	0.206
Nested CV - std	0.097	0.106	0.182	0.102	0.125	0.113	0.201
Training results	0.713	0.458	0.690	0.721	0.551	0.706	0.368
Test results	0.675	0.409	0.581	0.708	0.480	0.644	0.262

Table H.4: Baseline Approach - PCA: SVM Linear Kernel Results

H.1.2 Derived Statistics Approach

The following preproccesing steps are applied on the data set for this approach:

- mode/mean missing value imputation
- Feature reduction through PCA
- Train/test set ratio of 70/30
- Prediction based on derived statistics

SVM - Linear Kernel

The tuning rounds used to find the final tuned model for a linear kernel are summarized in Table H.5:

Tuning Round	Random Search 1	Grid search 1
Parameters	C	C
Range/Grid C	[0,10]	[0.0001,0.001,0.01,0.1]
Chosen C	0.01	0.01

Table H.5: Derived Approach - PCA: SVM Linear Kernel Tuning

The chosen hyper-parameters from grid search is a regularization strength of 0.01 for a linear kernel. The results from nested cross validation, training and test set are summarized in Table H.6

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.726	0.352	0.483	0.795	0.400	0.639	0.257
Nested CV - std	0.088	0.235	0.302	0.132	0.247	0.126	0.240
Training results	0.786	0.526	0.862	0.763	0.654	0.813	0.544
Test results	0.660	0.366	0.600	0.679	0.455	0.640	0.243

Table H.6: Derived Approach - PCA: SVM Linear Kernel Results

SVM - Polynomial Kernel

The tuning rounds used to find the final tuned model for a polynomial kernel are summarized in Table H.7.

Tuning Round	Random Search 1	Grid search 1
Parameters	C,γ,d,r	Cγ,d,r
Range/Grid C	[0,10]	[0.5,8.5], SS = 0.5
Range/Grid γ	(0,10]	[0.0001,0.001,0.01,0.1]
Range/Grid d	[1,5]	[1,2], SS = 0.1
Range/grid r	[0,10]	[0.0,2.0], SS = 0.5
Chosen C	4.10	1.0
Chosen γ	0.001	0.01
Chosen d	1	1
Chosen r	1	0

Table H.7: Derived Approach - PCA: SVM Polynomial Kernel Tuning. SS = Step Size

The chosen hyper-parameters are;

- Regularization strength: 1
- Gamma scaling: 0.01
- Polynomial degree: 1
- Independent term: 0

The results from nested cross validation, training and test set for this set of hyper-parameters are summarized in Table H.8

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Nested CV - mean	0.602	0.372	0.683	0.584	0.458	0.634	0.240
Nested CV - std	0.209	0.141	0.241	0.298	0.145	0.122	0.220
Training results	0.786	0.526	0.862	0.763	0.654	0.813	0.544
Test results	0.660	0.366	0.600	0.679	0.455	0.640	0.243

Table H.8: Derived Approach - PCA: SVM Polynomial Kernel Results

H.2 Comparison

We can compare the test results from chapter 5 to the results presented in this appendix. The comparisons can be viewed in Tables H.9, H.10, H.11 and H.12.

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Total	0.658	0.407	0.710	0.640	0.518	0.675	0.308
PCA	0.667	0.400	0.581	0.697	0.474	0.639	0.251

Table H.9: Total Scores VS PCA: Baseline Approach Logistic Regression Test Results

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Total BL SVM-Linear	0.683	0.422	0.613	0.708	0.500	0.660	0.290
PCA BL SVM-Linear	0.675	0.409	0.581	0.708	0.480	0.644	0.262

Table H.10: Total Scores VS PCA:: Baseline Approach SVM-Linear Test Results

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Total DE SVM-Linear	0.830	0.621	0.720	0.864	0.667	0.792	0.556
PCA DE SVM-Linear	0.660	0.366	0.600	0.679	0.455	0.640	0.243

Table H.11: Total Scores VS PCA:: Derived Approach SVM-Linear Test Results

Result Type	accuracy	precision	sensitivity	specificity	f1	roc_auc	MCC
Total DE SVM-Poly	0.745	0.475	0.760	0.741	0.585	0.750	0.439
PCA DE SVM-Poly	0.660	0.366	0.600	0.679	0.455	0.640	0.243

Table H.12: Total Scores VS PCA:: Derived Approach SVM-Polynomial Test Results

The results from the data set that uses PCA as feature reduction are worse for all models tested. Every metric is worse, except for accuracy and specificity in the baseline approach logistic regression. We can conclude that with a similar amount of feature reduction, PCA performs worse as a feature reduction method than total scores for this dataset.

Bibliography

[1] American Parkinson Disease Association. A guide for understanding Parkinson's disease psychosis, hallucinations and delusions. URL https://www.apdaparkinson.org/what-is-parkinsons/symptoms/psychosis/. [Accessed: Apr 8, 2020].

[2] Laura B. Zahodne and Hubert H. Fernandez. Parkinson's psychosis. *Current treatment options in neurology*, 12(3):200–211, May 2010. ISSN 1534-3138. doi: 10.1007/s11940-010-0072-y. URL https://pubmed.ncbi.nlm.nih.gov/20842582. 20842582[pmid].

[3] Matthew J. Barrett, Jamie C. Blair, Scott A. Sperling, Mark E. Smolkin, and T. Jason Druzgal. Baseline symptoms and basal forebrain volume predict future psychosis in early parkinson disease. *Neurology*, 2018. doi: https://doi.org/10.1212/WNL.0000000000005421.

[4] Charles H. Adler, Thomas G. Beach, Joseph G. Hentz, Holly A. Shill, John N. Caviness, Erika Driver-Dunckley, Marwan N. Sabbagh, Lucia I. Sue, Sandra A. Jacobson, Christine M. Belden, and Brittany N. Dugger. Low clinical diagnostic accuracy of early vs advanced parkinson disease. *Neurology*, 83(5):406–412, 2014. ISSN 0028-3878. doi: 10.1212/WNL.0000000000000641. URL https://n.neurology.org/content/83/5/406.

[5] Dominic H ffytche, Joana B Pereira, Clive Ballard, K Ray Chaudhuri, Daniel Weintraub, and Dag Aarsland. Risk factors for early psychosis in pd: insights from the parkinson's progression markers initiative. *Journal of Neurology, Neurosurgery & Psychiatry*, 88(4):325–331, 2017. ISSN 0022-3050. doi: 10.1136/jnnp-2016-314832. URL https://jnnp.bmj.com/content/88/4/325.

[6] Sykepleien. Dette bør du vite om Parkinsons sykdom. URL https://sykepleien.no/2017/10/dette-bor-du-vite-om-parkinsons-sykdom. [Accessed: Feb 14, 2020].

[7] American National Institute on Aging. Parkinson's Disease. URL https://www.nia.nih.gov/health/parkinsons-disease. [Accessed: Apr 10, 2020].

[8] World Health Organization. Parkinson's disease. URL https://www.who.int/mental_health/neurology/neurological_disorders_report_web.pdf. [Accessed: Feb 14, 2020].

[9] PPMI. Parkinson's Progression Markers Initiative's website - main page, . URL https://www.ppmi-info.org. [Accessed: Feb 14, 2020].

[10] Merriam Webster. Cohort. URL https://www.merriam-webster.com/dictionary/cohort. [Accessed: Feb 14, 2020].

[11] PPMI. Parkinson's progression markers initiative's website - study cohorts, . URL https://www.ppmi-info.org/study-design/study-cohorts. [Accessed: Feb 14, 2020].

[12] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 0-387-31073-8.

[13] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning*. MIT Press, 2016. ISBN 9780262035613.

[14] Leo Breiman. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statist. Sci.*, 16(3):199–231, 08 2001. doi: 10.1214/ss/1009213726. URL https://doi.org/10.1214/ss/1009213726.

[15] Jørn Wetterslev Janus Christian Jakobsen, Christian Gluud and Per Winkel. When and how should multiple imputation be used for handling missing data in randomised clinical trials – a practical guide with flowcharts. *BMC Medical Research Methodology*, December 2017. doi: https://doi.org/10.1186/s12874-017-0442-1. URL https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/s12874-017-0442-1. [Accessed: Apr 18, 2020].

[16] Kate Tilling Paul Madley Dowd, Rachael Hughes and Jon Heron. The proportion of missing data should not be used to guide decisions on multiple imputation. *Journal of Clinical Epidemiology*, June 2019. URL https://www.sciencedirect.com/science/article/pii/S0895435618308710. [Accessed: Apr 18, 2020].

[17] R.J.A. Little and D.B. Rubin. *Statistical Analysis with Missing Data*. Wiley Series in Probability and Statistics. Wiley, 2019. ISBN 9780470526798. URL https://books.google.no/books?id=BemMDwAAQBAJ.

[18] Chaoqun Li and Hongwei Li. Correlation weighted heterogeneous euclidean-overlap metric. *International Journal of Computers and Applications*, 33(4):341–346, 2011. doi: 10.2316/Journal.202.2011.4.202-3179. URL https://www.tandfonline.com/doi/abs/10.2316/Journal.202.2011.4.202-3179.

[19] Peter E. Hart Richard O. Duda and David G. Stork. *Pattern Classification*. Wiley, 2001. ISBN 0-471-05669-3.

[20] H. Motoda and H. Liu. Feature selection, extraction and construction. In *Proceedings of Foundation of Data Mining, PAKDD02*, pages 67–72, 2002. URL http://www.ar.sanken.osaka-u.ac.jp/~motoda/papers/fdws02.pdf.

[21] Lindsay I Smith. A tutorial on principal components analysis, 2002. URL http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf. [Accessed: Mar 29, 2020].

[22] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, New York, NY, 2017.

[23] Susan Jamieson. Likert scales: how to (ab)use them. *Medical Education*, November 25 2004. doi: 10.1111/j.1365-2929.2004.02012.x. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.474.608&rep=rep1&type=pdf.

[24] Statistics Solutions. Can an ordinal likert scale be a continuous variable? URL https://www.statisticssolutions.com/can-an-ordinal-likert-scale-be-a-continuous-variable/. [Accessed: Mar 29, 2020].

[25] Krzysztof Jajuga and Marek Walesiak. Standardization of data set under different measurement scales. URL http://keii.ae.jgora.pl/pracownicy/mw/2000_Jajuga_Walesiak_Springer.PDF. [Accessed: Mar 29, 2020].

[26] James Carifio and Rocco J. Perla. Ten common misunderstandings, misconceptions, persistent myths and urban legends about likert scales and likert response formats and their antidotes. *Journal of Social Sciences*, 2007. URL https://thescipub.com/PDF/jssp.2007.106.116.pdf. [Accessed: Mar 29, 2020].

[27] Patrick Heagerty. Lectures: Longitudinal Data Analysis, 2006. URL https://faculty.washington.edu/heagerty/Courses/VA-longitudinal/private/LDAchapter.pdf. [Accessed: Mar 21, 2020].

[28] Andreas C. Müller and Sarah Guido. *Introduction to Machine Learning with Python*. O'Reilly Media, Sebastopol, CA, 2016.

[29] Tianqi Chen. Introduction to Boosted Trees. URL https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf. [Accessed: Apr 27, 2020].

[30] Mark Hall Ian Witten, Eibe Frank and Christopher Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016. ISBN 9780128043578.

[31] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1101. URL https://www.aclweb.org/anthology/P16-1101.

[32] Long short-term memory. URL https://en.wikipedia.org/wiki/Long_short-term_memory. [Accessed: May 6, 2020].

[33] Christopher M Florkowski. Sensitivity, specificity, receiver-operating characteristic (roc) curves and likelihood ratios: Communicating the performance of diagnostic tests. *The Clinical biochemist. Reviews*, 29 (Suppl 1):S83–S87, Aug 2008. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2556590/. [Accessed: Mar. 15, 2020].

[34] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21, Jan 2 2020. doi: 10.1186/s12864-019-6413-7.

[35] Hyperparameter optimization. URL https://en.wikipedia.org/wiki/Hyperparameter_optimization. [Accessed: May 17, 2020].

[36] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL http://jmlr.org/papers/v13/bergstra12a.html.

[37] Hyperparameter tuning. URL https://www.oreilly.com/library/view/evaluating-machine-learning/9781492048756/ch04.html. [Accessed: Apr 25, 2020].

[38] Gavin C. Cawley and Nicola L. C. Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11:2079–2107, 2010. URL http://jmlr.csail.mit.edu/papers/volume11/cawley10a/cawley10a.pdf.

[39] T.C.N. Lo B.R. Thanvi and D.P. Harsh. Psychosis in parkinson's disease. *Postgraduate Medical Journal*, pages 644–646, October 6 2005. doi: 10.1136/pgmj.2004.032029. [Accessed: Mar. 11, 2020].

[40] MoCA - official website. URL https://www.mocatest.org/.

[41] Stacy Belkonen. *Hopkins Verbal Learning Test*, pages 1264–1265. Springer New York, New York, NY, 2011. ISBN 978-0-387-79948-3. doi: 10.1007/978-0-387-79948-3_1127. URL https://doi.org/10.1007/978-0-387-79948-3_1127.

[42] PPMI. Parkinson's Progression Markers Initiative's website - study protocol, . URL http://www.ppmi-info.org/wp-content/uploads/2018/02/PPMI-AM-13-Protocol.pdf. [Accessed: Feb 14, 2020].

[43] Semantic Verbal Fluency test in dementia: Preliminary retrospective analysis. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5619418/. [Accessed: Feb 14, 2020].

[44] REM Sleep Behavior Disorder Screening Questionnaire. URL https://en.wikipedia.org/wiki/REM_Sleep_Behavior_Disorder_Screening_Questionnaire. [Accessed: Feb 14, 2020].

[45] Geriatric Depression Scale. URL https://en.wikipedia.org/wiki/Geriatric_Depression_Scale. [Accessed: Feb 14, 2020].

[46] University of Pennsylvania Smell Identification Test. URL https://en.wikipedia.org/wiki/University_of_Pennsylvania_Smell_Identification_Test. [Accessed: Feb 14, 2020].

[47] About the ESS. URL https://epworthsleepinessscale.com/about-the-ess/. [Accessed: Feb 14, 2020].

[48] Autonomic Nervous System - Definition. URL https://www.lexico.com/definition/autonomic_nervous_system. [Accessed: Feb 14, 2020].

[49] SCales for Outcomes in PArkinson's disease - Autonomic Dysfunction. URL https://www.movementdisorders.org/MDS/MDS-Rating-Scales/Scales-for-Outcomes-in-Parkinsons-disease---Autonomic-Dysfunction.htm. [Accessed: Feb 14, 2020].

[50] International Parkinson and Movement Disorder Society. The MDS-sponsored Revision of the Unified Parkinson's Disease Rating Scale. URL https://www.movementdisorders.org/MDS-Files1/PDFs/Rating-Scales/MDS-UPDRS_English_FINAL_Updated_August2019.pdf. [Accessed: May 5, 2020].

[51] Sally J. Vogel, Sarah J. Banks, Jeffrey L. Cummings, and Justin B. Miller. Concordance of the montreal cognitive assessment with standard neuropsychological measures. *Alzheimer's Dementia: Diagnosis, Assessment Disease Monitoring*, 1(3):289 – 294, 2015. ISSN 2352-8729. doi: https://doi.org/10.1016/j.dadm.2015.05.002. URL http://www.sciencedirect.com/science/article/pii/S2352872915000512.

[52] B Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5:221–232, 2016. doi: https://doi.org/10.

1007/s13748-016-0094-0. URL https://link.springer.com/article/10.1007/s13748-016-0094-0.

[53] K. Jajuga and M. Walesiak. Standardisation of data set under different measurement scales. In Reinhold Decker and Wolfgang Gaul, editors, *Classification and Information Processing at the Turn of the Millennium*, pages 105–112, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-642-57280-7.

[54] sklearn.utils.class_weight.compute_class_weight. URL https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html?highlight=class_weight#. [Accessed: Jun 08, 2020].

[55] 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*, 12:2825–2830, 2011. URL http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf.

[56] sklearn.linear_model.linearregression, . URL https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html. [Accessed: May 27, 2020].

[57] sklearn.svm.svc, . URL https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed: May 27, 2020].

[58] sci-kit learn developers. 1.4. Support Vector Machines, . URL https://scikit-learn.org/stable/modules/svm.html#svm-classification. [Accessed: May 8, 2020].

[59] Decision trees, . URL https://scikit-learn.org/stable/modules/tree.html. [Accessed: May 27, 2020].

[60] 3.2.4.3.1.sklearn.ensemble.randomforestclassifier, . URL https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Accessed: May 27, 2020].

[61] Xgboost python package. URL https://xgboost.readthedocs.io/en/latest/python/index.html. [Accessed: May 28, 2020].

[62] XGBoost developers. Notes on Parameter Tuning, . URL https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html. [Accessed: May 15, 2020].

[63] XGBoost developers. XGBoost Parameters, . URL https://xgboost.readthedocs.io/en/latest/parameter.html. [Accessed: May 15, 2020].

[64] Keras developers. Keras - main website, . URL https://keras.io/. [Accessed: May 19, 2020].

[65] J. Brownlee. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018. URL https://books.google.no/books?id=T1-nDwAAQBAJ.

[66] A.P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley, 2007. ISBN 9780470512500. URL https://books.google.no/books?id=IZosIcgJMjUC.

[67] G.B. Orr and K.R. Müller. *Neural Networks: Tricks of the Trade*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003. ISBN 9783540494300. URL https://books.google.no/books?id=VCKqCAAAQBAJ.

[68] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[69] 3.5.2. function for prediction-error metrics, . URL https://scikit-learn.org/0.15/modules/model_evaluation.html#function-for-prediction-error-metrics. [Accessed: May 28, 2020].

[70] 3. model selection and evaluation, . URL https://scikit-learn.org/stable/model_selection.html. [Accessed: May 28, 2020].

[71] sci-kit learn developers. RBF SVM parameters, . URL https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Accessed: May 9, 2020].

[72] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 01 2002. doi: 10.1613/jair.953.

[73] John B Carlin Michael Spratt Patrick Royston Michael G Kenward Angela M Wood Jonathan A C Sterne, Ian R White and James R Carpenter. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ British Medical Journal*, June 2009. doi: https://doi.org/10.1136/bmj.b2393. URL https://www.bmj.com/content/338/bmj.b2393. [Accessed: Apr 18, 2020].

[74] Science Direct. Amyloid, . URL https://www.sciencedirect.com/topics/medicine-and-dentistry/amyloid. [Accessed: May 30, 2020].

[75] Science Direct. Multiple Discriminant Analysis, . URL https://www.sciencedirect.com/topics/computer-science/multiple-discriminant-analysis. [Accessed: May 30, 2020].