www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Maastricht University

Faculty of Sciences
School for Information Technology
Master of Statistics

Master's thesis

Towards personalized medicine: Predicting disability progression of MS patients from evoked potentials

NOAH KASUNUMBA
Thesis presented in fulfillment of the requirements for the degree of Master of Statistics, specialization Bioinformatics

SUPERVISOR :
dr. Thijs BECKER

SUPERVISOR :
Dr. Liesbet PEETERS

2019
2020

Faculty of Sciences

School for Information Technology

Master of Statistics

Master's thesis

Towards personalized medicine: Predicting disability progression of MS patients from evoked potentials

NOAH KASUNUMBA

Thesis presented in fulfillment of the requirements for the degree of Master of Statistics, specialization Bioinformatics

SUPERVISOR :

dr. Thijs BECKER

SUPERVISOR :

Dr. Liesbet PEETERS

1 Abstract

Background: Multiple sclerosis (MS) is an incurable chronic disease of the central nervous system (CNS) that is affecting millions of people world wide. Even though magnetic resonance imaging (MRI) and Expanded disability status scale (EDSS) are widely used in assessing MS disability progression, evoked potentials (EPs) are used too. Predicting MS disability progression two years earlier could trigger treatment change in MS patients leading to better treatment outcomes. Until now, predicting MS disability progression has been a challenge. Previous works have used logistic regression (LR), a linear classifier. In this research we have applied several machine learning approaches to motor evoked potential (MEP), and for the first time to visual evoked potential (VEP) and somatosensory evoked potential (SSEP). We have further used a longitudinal approach (regression trees with random effects model (REEMtree)) for the first time to the three EPs using data for patients under treatment from Rehabilitation & MS center in Overpelt, Belgium.
Methods: We use the lasso approach to all EP latencies and EDSS to select variables that minimise cross validation error. We apply the LR, random forest (RF), neural network (NN), boosting and REEMtree models to predict MS disability progression after two years. Each of these models predicted disability progression 10000 times, and each time we studied the receiver operating character (ROC) area under the curve (AUC). We applied Statistical significance tests to establish the best ways to predict MS disability progression.
Results: Using REEMtree, a longitudinal model that exploits the clustered nature of the data, yields incredibly good results (AUC 0.967 ± 0.013 for MEP, 0.963 ± 0.013 for VEP and 0.967 ± 0.012 for SSEP) outperforming the second best model (RF) and the baseline model (LR) ((0.732 ± 0.05, 0.687 ± 0.06), (0.72 ± 0.06, 0.661 ± 0.07) and (0.702 ± 0.06, 0.623 ± 0.08) for MEP, VEP and SSEP respectively). This is a very good classifier, given the fact that we only had EDSS and EPs as biomarkers. The MEP and SSEP have similar MS prediction performance (p_value 0.07) which is superior to VEP (p_value < 0.000). Finally ensembling limb or eye specific analysis with the longitudinal model does not improve MS disability progression prediction for all EPs (p_values ≥ 0.92 for all EPs).
Conclusions: There is hope for MS patients that machine learning approaches can be used to predict MS disability progression with high accuracy. The non linear classifiers out perform the linear classifiers, and the longitudinal model out performs all the other models. This model can inform clinicians about the future treatment choices for better MS treatment outcomes.

Contents

2 Acknowledgements

As a student of Hasselt University, I am firstly grateful to the Master of statistics, long distance programme which has enabled me to acquire high class knowledge, while giving me a chance to work and support my family. I am very grateful to all the professors that have taught me, they indeed gave me more than what I needed to succeed at this project.

I would like to express my special thanks of gratitude to my supervisors: Dr. Thijs Becker and Dr Liesbet Peeters who have given me an excellent opportunity to do this wonderful project. I have been exposed to real world data from a hospital setting, giving me an opportunity to make a difference. This project has been successful only because of your continuous guidance, reviews with very important feedback that you relentlessly gave. I am gratefully indebted to you for not getting tired of me, your doors were always open whenever I ran into a trouble spot or had a question about my research.

I must express my very profound gratitude to my wife, Awinjo Juliet, for the love, support and continuous encouragement throughout my years of study. Special thanks go to my children, Abraham Kasunumba, Jemimah Kwagala, Keren Hope and Liesbet Kirabo for always being my motivation. This accomplishment would not have been possible without you.

Above all, I am grateful to the Almighty God, who already knew me from the beginning of the age. He has always put the right people in my path, protected, provided and encouraged me in all my life's endeavours.

Thank you

Noah Kasunumba.

List of Tables

List of Figures

3 List of abbreviations

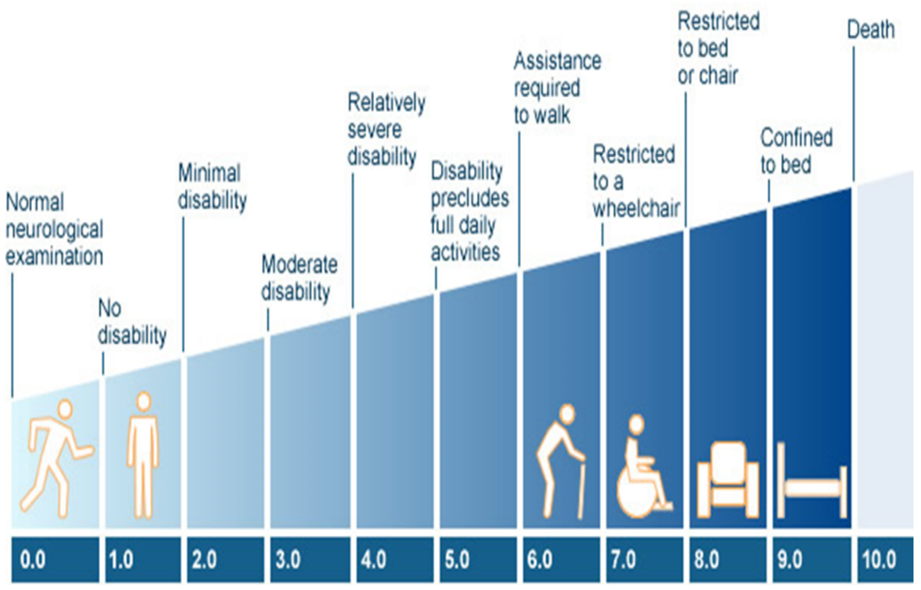
AH	Adductor Hallucis
APB	Abductor Pollicis Brevis
AUC	Area Under Curve
BH	Benjamini-Hochberg
BY	Benjamini–Yekutieli
CNS	Central Nervous System
EDSS	Expanded Disability Status Scale
EP	Evoked Potentials
FDR	False Discovery Rate
LASSO	Least Absolute Shrinkage and Selection Operator
LR	Logistic Regression
MEP	Motor EP
MRI	Magnetic Resonance Imaging
MS	Multiple Sclerosis
NN	Neural Network
OOB	Out Of Bag
PCA	Principle Component Analysis
REEMtree	Regression Trees with Random Effects Model
RF	Random Forest
ROC	Receiver Operating Characteristic
SD	Standard Deviation
SEP	somatosensory EP
VEP	Visual EP

4 Introduction

Multiple sclerosis (MS) is an inflammatory disease that affects the CNS, i.e. the brain and spinal cord, and usually starts between 20 and 40 years of age [37] affecting more women by at least twice the men. It is the commonest demyelinating disease of the CNS in northern Europe and America and even though its etiology is still unknown, a T cell-mediated autoimmune pathogenesis is likely to be responsible for the demyelination [26]. This demyelination causes patients to acquire symptoms which include sensation deficits and motor, autonomic and neurocoginitive dysfunction which depend on the site of the lesions [44]. It is estimated that in Europe alone, there are over 700,000 patients living with the disease with over 2.5 million people world wide [7]. Like any other disease, its impact on the social status and the well being on both the individuals and the economy at large can not be ignored [22, 38]. Currently there is no cure for the disease, but only therapeutic strategy is offered to patients to reduce the risk of relapses and disability progression [28]. With this disease, if no treatment is offered, at least 50% of MS patients will progress into another worse state after two decades, independent of acute attacks [10, 42].

Predicting disability progression, accurately, in MS is still a major challenge [39]. If the non-response to the current treatment is detected early, a treatment switch would help inform treatment choices to improve the outcome of the treatment [44]. There are so many biomarkers available for MS, and even though the magnetic resonance imaging (MRI), which visualizes lesions in CNS, is used, there are other clinical parameters which include expanded disability status scale (EDSS) [23] which can also be used to assess disability progression [13, 40, 27, 32, 24].
In the figure 1, the higher the EDSS score, the worse the disease stage and it would help patients if progression to another stage does not occur.

In addition to the EDSS and MRI, evoked potentials (EP) can be used to monitor MS disability, but also used in assessing MS progression [29, 25, 21, 16, 33]. According to [44], EP provide a quantitative information on the functional integrity of well-defined pathways of the CNS, and reveal early infra-clinical lesions. These are able to detect the reduction in electrical conduction caused by damage along these pathways even when the change is too subtle to be noticed by a person or to translate into clinical symptoms. There are many EPs that can be extracted from MS patients [41], but for this particular study, we have three EPs which include the Visual evoked potental (VEP), motor evoked potential (MEP) and somatosensory evoked potential (SSEP).



Normal neurological examination
No disability
Minimal disability
Moderate disability
Relatively severe disability
Disability precludes full daily activities
Assistance required to walk
Restricted to a wheelchair
Restricted to bed or chair
Confined to bed
Death
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0

Figure 1: The EDSS score at each MS disability stage.

The time taken for the signal to arrive (latency) has always been used to study MS disability progression in literature, using linear correlation with EDSS and also using logistic regression [44]. This study is perhaps the second of its kind to use machine learning technology to predict MS disability progression from EPs. Most of the studies have used cross sectional analysis techniques. The only work that has been done using machine learning is by [44] which focused on MEP but did not explore other EPs. In addition, the analysis was done by analysing the arms and legs separately, and then aggregating the results from both limbs. While our research does not use time series data, it has used different EPs and also used, in addition to the methods regularly used, a model that takes into account the longitudinal nature of the data set (REEMtree). Even though [44] analysed both limbs separately, this study will test if this approach yields significantly better results than simply studying all the latencies at once.

4.1 Research questions

We have four hypotheses for this study outlined below

- Not all EP latencies are required when used with baseline EDSS to predict MS disability progression.
- Machine learning approaches perform differently when predicting MS disability progression.
- Some EPs give superior results compared to others when used to predict MS disability progression after two years.
- Predicting MS disability progression by using latencies from each limb (MEP and SSEP) or eye (VEP) separately has superior predictive power as compared to studying all latencies at once for all EPs.

We have hence developed four research questions that have helped us come up with relevant conclusions, and these are outlined below.

- Which latencies are important when used with baseline EDSS for predicting MS disability progression with MEP, VEP and SSEP?
- Which machine learning model best predicts MS disability progression after 2 year?
- Which EP provides best predictions for MS disability progression after two years?
- Is it true that predicting MS disability progression using the left and right latencies separately gives superior results compared to when all latencies are combined at once for all EPs?

5 Description of the data set

5.1 Provided data

To explore the subject matter, we use data from Pelt Rehabilitation & MS center, a specialized center for people with MS and other disorders of the nervous system which is located in Pelt, Belgium. The provided data on evoked potentials included the MEP (both upper and lower with 1971 visits), VEP (2146), SSEP (upper (1984) and lower (331)) and EDSS (8396) visits. All the data sets, except the EDSS data set had information about the patient id, visit dates and latencies. The EDSS data set had information about the patient id, patient visit dates and the EDSS score during the visit.

These are longitudinal data collected between 2006 and 2017. The EDSS visits are not necessarily the visits for all the other EP visits, which implies for example that not every MEP visit will have an EDSS measurement, and so forth. Like any other longitudinal data set, the data at hand had missing values, which are considered the first obstacle in predictive modelling. The proportion of missing data was more with the SSEP lower and SSEP upper data sets with 29% and 17% records with missing data respectively. The EDSS data set did not have any missing data as seen in table 1. In machine learning the idea is to maximally utilize the available data. We have used the Hmisc package in R [17] to impute the missing values in the simplest manner possible. With this package, we have an option to impute using a user defined statistic for example the median, mean, random value etc. We have imputed with random value, which is simply any observation within the variable picked at random. We have opted out of the other choices for reason of not having one value dominate the missing latency values.

Because not every EP visit had EDSS measurement, the baseline EDSS is the one for the individual's visit closest to the EP visit with in one year. EP visits without EDSS measurements that fall within one year have been discarded from further analyses.

We finally find the EDSS measurement after 2 years from the baseline EDSS. Again here, we have given an allowance of measurements taken between 1.5 and 3 years, whichever is close to 2 years as the final EDSS measurement taken after two years. Records that did not have a followup visit after two years have been omitted from further analyses. After this step, we had the following visits ready for analysis, 1724 (MEP), 1364 (VEP), 1286 (SSEP Upper) and 80 (SSEP lower).

We use the standard definition of disability progression [20], where the patient has progressed if $EDSS_{T1} - EDSS_{To} \geq 1.0$ for $EDSS_{To} \leq 5.5$, or if $EDSS_{T1} - EDSS_{To} \geq 0.5$ for $EDSS_{To} > 5.5$. The summary of this step is shown in table 1
The SSEP lower final data set is too small to train machine learning models. We have hence forth eliminated it from further study and used the MEP, VEP and SSEP upper data sets to answer our research questions.

	EDSS	**MEP**	**VEP**	**SSEP_upper**	**SSEP_lower**
Number Visits	8396	1971	2146	1984	331
Variables	3	6	14	10	18
Unique Patients	585	405	558	551	257
% rows with missing data	0(0%)	138(7%)	69(3.2%)	339(17.1%)	96(29%)
% With baseline EDSS	-	1889(95.8%)	2031(94.6%)	1878(94.7%)	323(97.6%)
% With follow up after 2 years	-	1724(87%)	1364(63.6%)	1286(64.8%)	80(24.2%)
% progression after 2 years	-	172(9.97%)	146(10.7%)	127(9.88%)	3(3.75%)

Table 1: Summary of the data provided at different stages for all the EPs

5.2 EP experiments

The data provided resulted from experiments explained below. For VEP, three electrodes were placed on the head, a checkerboard pattern was flashed at a distance of 1m20. Left and right eye were each measured twice (the other eye covered). The final measurement was averaged over 200 individual measurements, latencies were annotated automatically and around 10% needed to be adjusted manually. For the SSEP upper, three electrodes were placed on the head, a muscle in the thumb was stimulated, Left and right hand were each measured twice, the final measurement was averaged over 150 individual measurements, latencies were annotated automatically and around 10% needed to be adjusted manually. Regarding MEP, three electrodes were placed on the hands or feet, a magnetic coil stimulated the motor cortex, started at excitation strength 45% (hands) or 50% (feet) and increased by 5% until amplitude stopped increasing or reached 1, each time-series was an individual measurement, latencies were annotated automatically and around 90% needed to be adjusted manually. The figures 2 3 and 4 depict the three different EPs used for modeling in this work with annotated latencies.

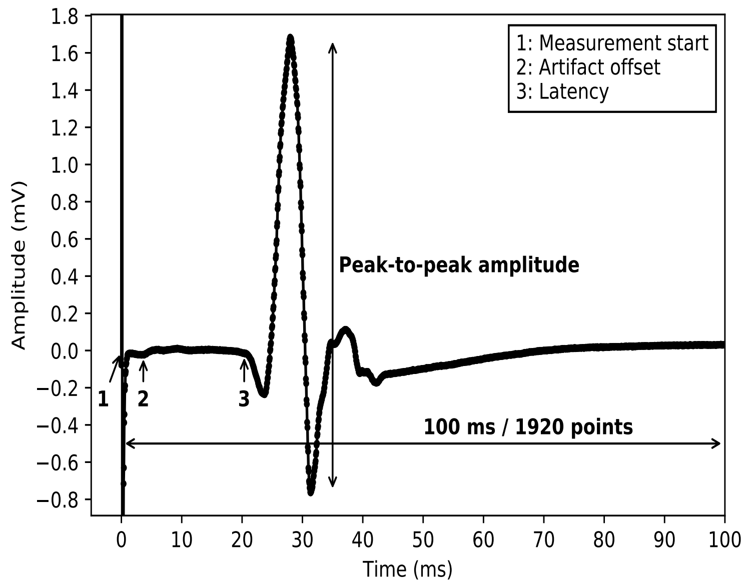


Figure 2: MEP with annotated latencies. Image source: [44]

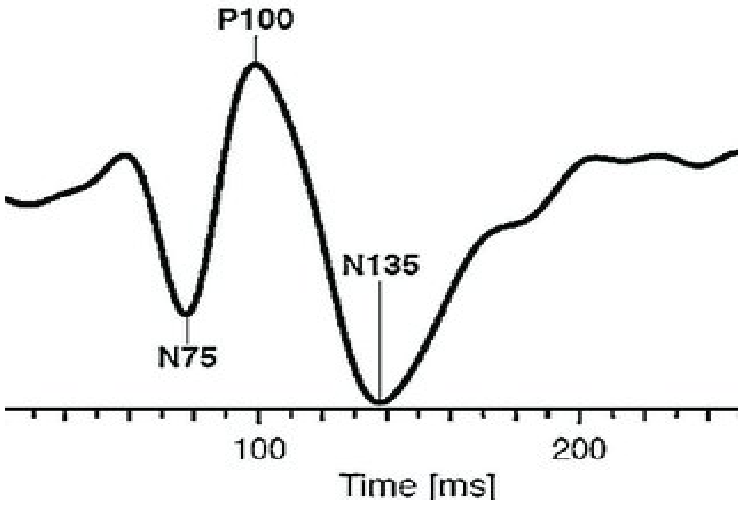


Figure 3: VEP with annotated latencies. Image source: [30]

In our data set, we used latency N145 instead of N135 shown in figure 3. We have two measurements for each latency (N145, P100 and N75) on both the left and right eyes.

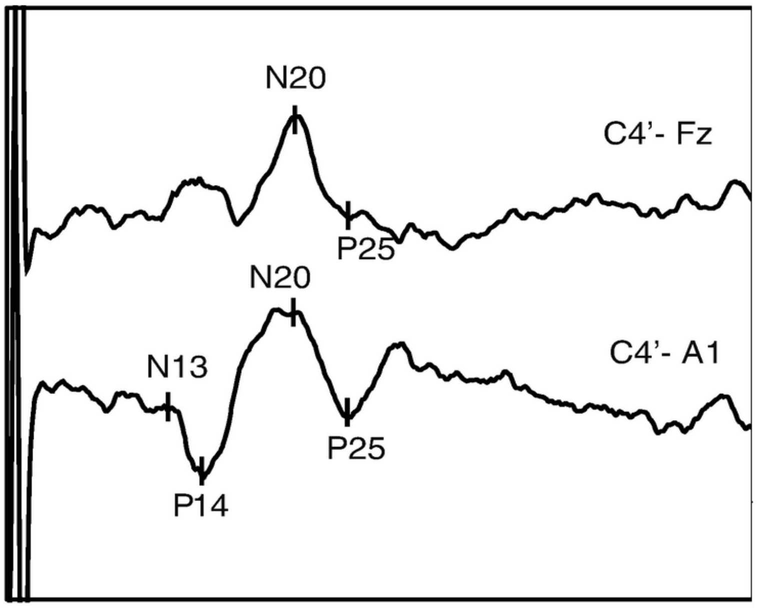


Figure 4: SSEP Upper with annotated latencies. Image source: [18]

In this work, we used latencies N20 and P25. we have not used latencies N13 and P14 shown in figure 4. Like with the VEP, we have two measurements for N20 latency on both the left and right limbs. We have three measurements for latency P25 on the right limb and one measurement on the left limb.

5.3 MEP Exploration

We have used the box plots to explore all the latencies in the MEP data set between patients that progressed and those that did not progress in the figures 5 and 6. we notice many latency values out of range in both groups, but we have opted to use all the data for all EPs since there are no better methods that could be used to identify the latency measurement error except the one by one manual checking method which is very time consuming. The general conclusion drawn from the MEP plots is that the average latencies for patients that progressed are always higher than that of patients that did not progress.

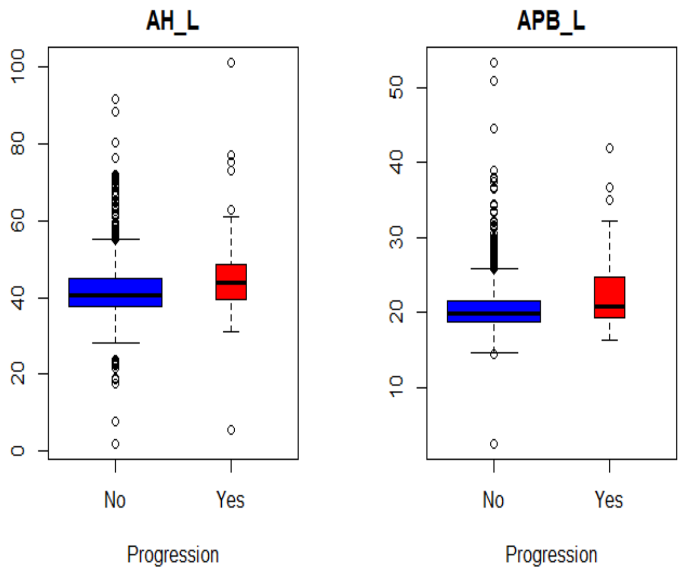


Figure 5: The relationship between the AH_L, APB_L latencies and MS disability progression

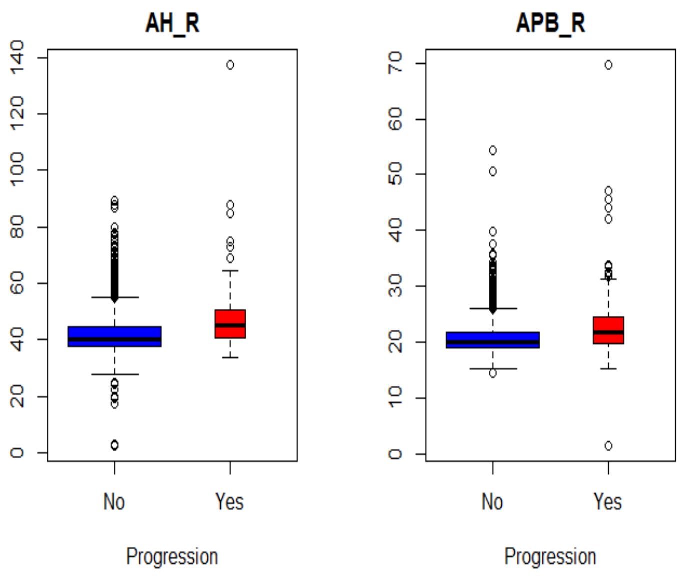


Figure 6: The relationship between the AH_R, APB_R latencies and MS disability progression

From the box plots, both the left and the right limb's latencies seem to have an effect on MS progression.

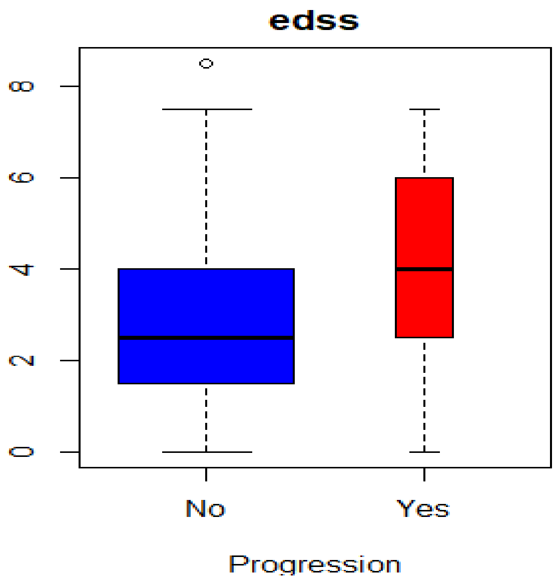


Figure 7: The relationship between the EDSS and MS disability progression for MEP data set

In the figure 7 above, we have plotted the EDSS for the two categories in different colours. Its clear that the baseline EDSS for the two categories is different from each other. The average EDSS for patients that progressed was 3.96 which is higher than that in patients that did not progress of 2.92. This suggests that baseline EDSS could be vital in predicting disability progression.

5.4 VEP Exploration

The VEP data set graphs presented in figures 8, 9, 10, 11, 12 and 13 also present the same trend that the average latencies in people that progressed are quit higher than that for patients that never progressed after two years. All the latencies here for both eyes show the same trend suggesting that both the left and the right eye may have a part to play in predicting MS disability progression.

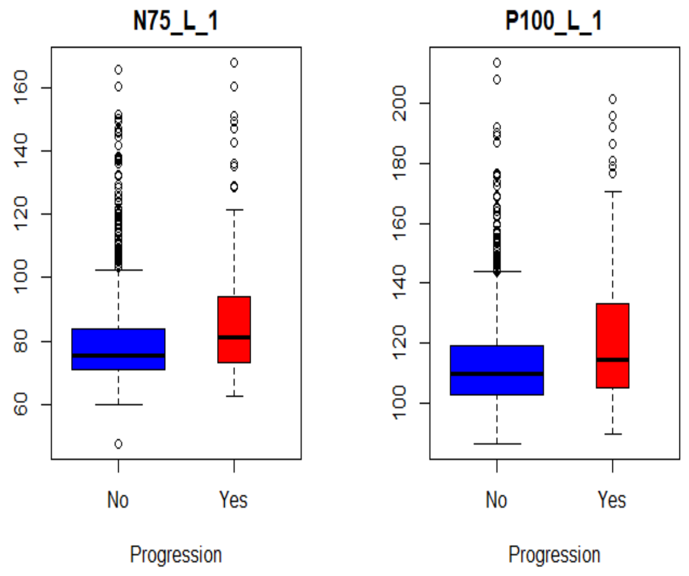


Figure 8: The relationship between the VEP latencies N75_L_1, P100_L_1 and MS disability progression

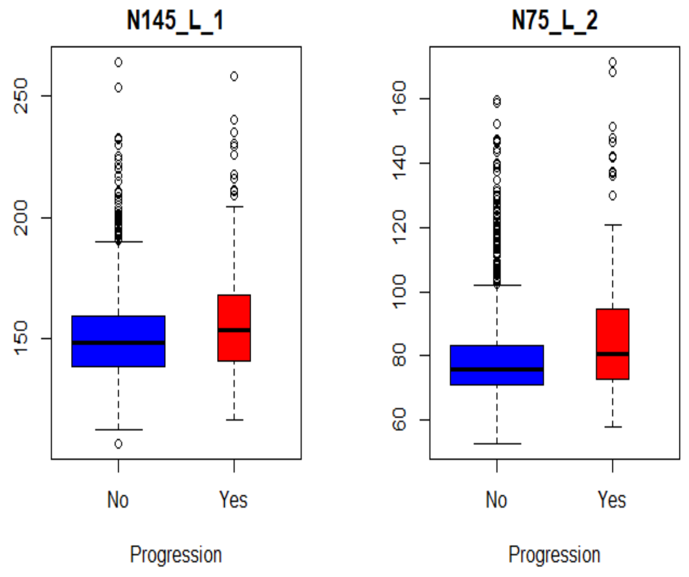


Figure 9: The relationship between the VEP latencies N145_L_1, N75_L_2 and MS disability progression

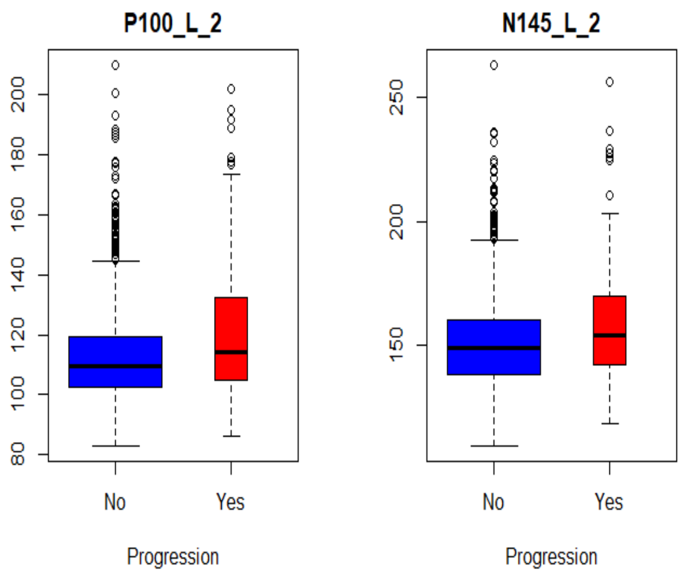


Figure 10: The relationship between the VEP latencies P100_L_2, N145_L_2 and MS disability progression

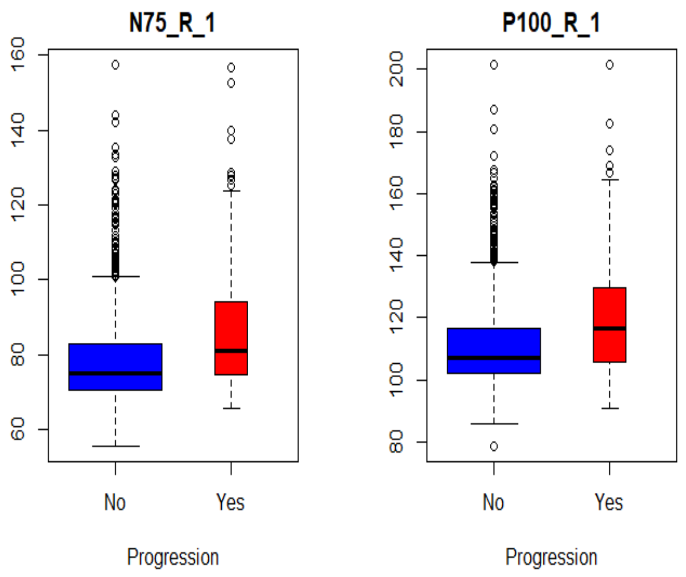


Figure 11: The relationship between the VEP latencies N75_R_1, P100_R_1 and MS disability progression

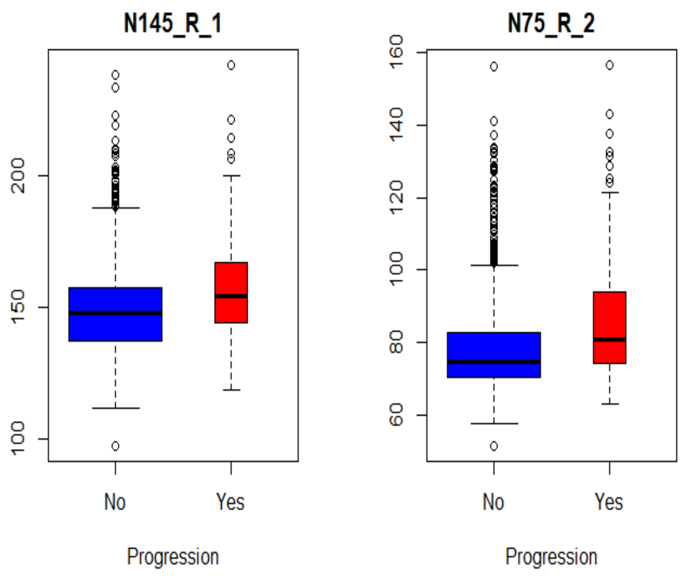


Figure 12: The relationship between the VEP latencies N145_R_1, N75_R_2 and MS disability progression

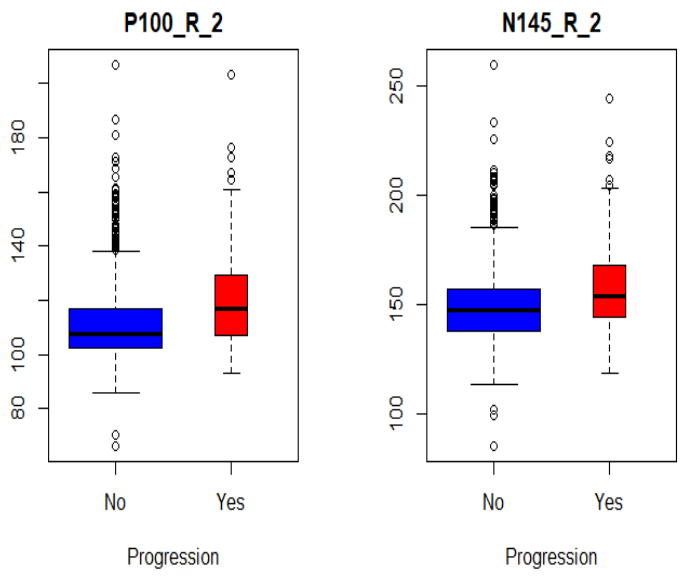


Figure 13: The relationship between the VEP latencies P100_R_2, N145_R_2 and MS disability progression

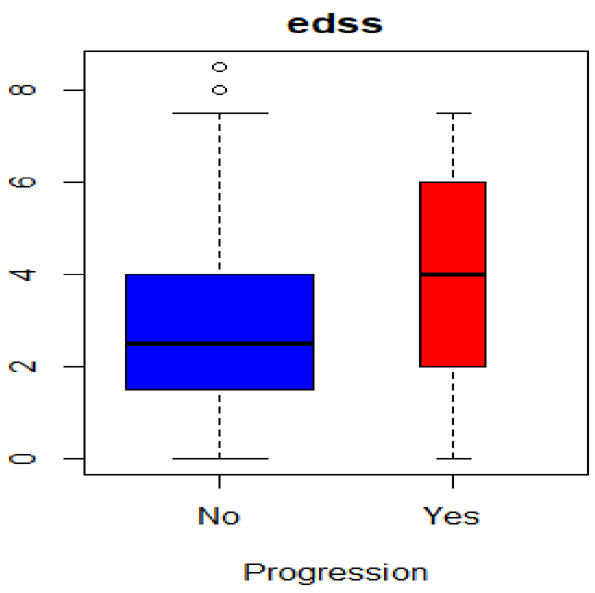


Figure 14: The relationship between the EDSS and MS disability progression for VEP data set

Like in the MEP data set, a plot shown in figure 14 shows a distinction in EDSS between the two group. The average EDSS among the patients that progressed is 4.12 and that among the patients that never progressed is 2.95 suggesting again that the EDSS could play a prediction role.

5.5 SSEP Exploration

The SSEP latencies are not any different from those of other EPs. The figures 15, 16, 17, 18 presented below show that the average latencies for patients that progressed is a bit higher than that of patients that never progressed with latency P25_L_1 and P25_R_2.1 seemingly outstanding among all the other latencies seen to influence disability progression.

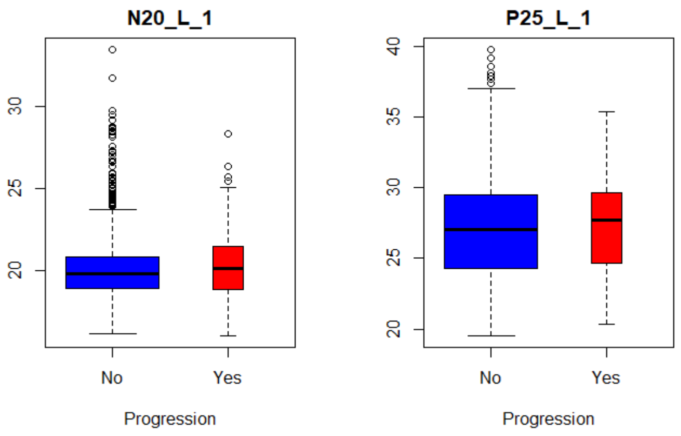


Figure 15: The relationship between the SSEP latencies N20_L_1, P25_L_1 and MS disability progression

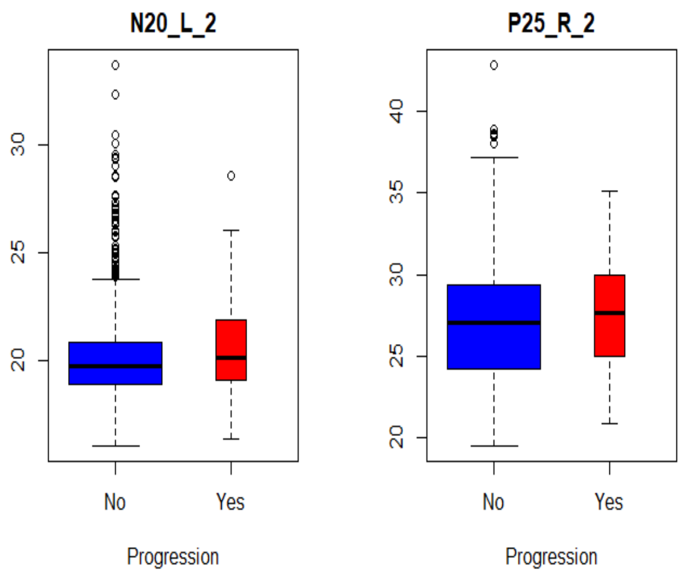


Figure 16: The relationship between the SSEP latencies N20_L_2, P25_R_2 and MS disability progression

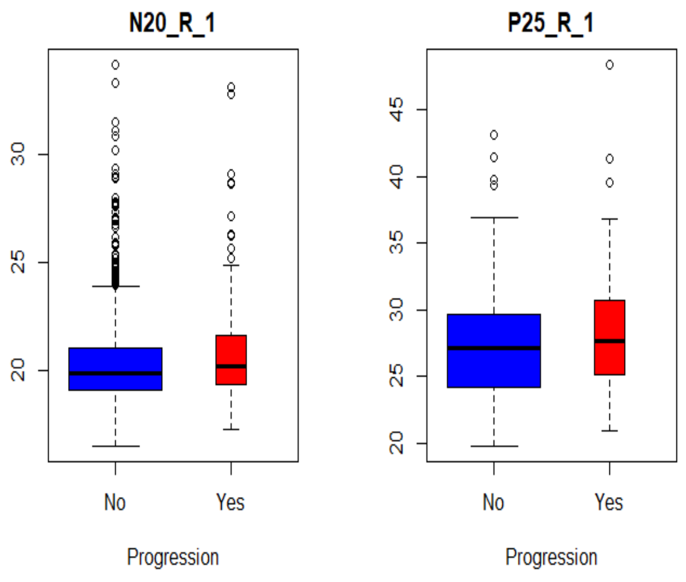


Figure 17: The relationship between the SSEP latencies N20_R_1, P25_R_1 and MS disability progression

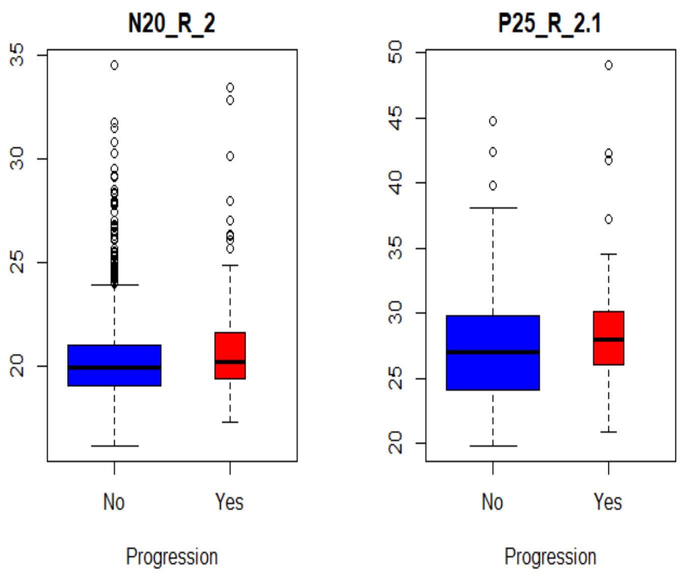


Figure 18: The relationship between the SSEP latencies N20_R_2, P25_R_2.1 and MS disability progression

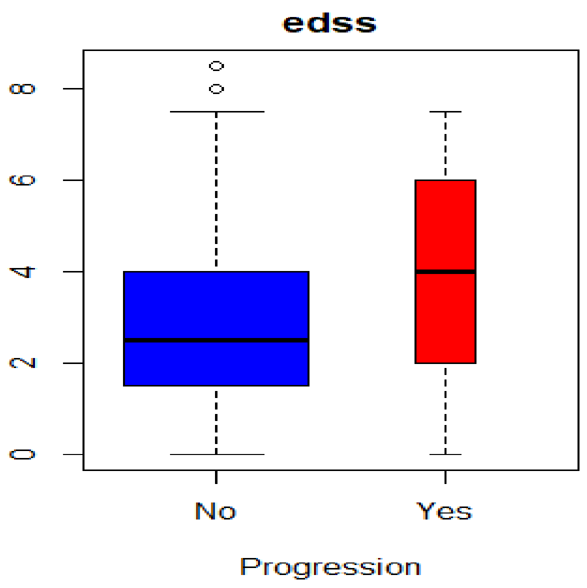


Figure 19: The relationship between the SSEP EDSS and MS disability progression

The plot 19 shows a clear distinction between the baseline EDSS among the two groups of patients. The average EDSS for the patients that progressed is 3.93 and that among patients that did not progress is 2.93, again, like other EPs suggesting how important the baseline EDSS might be in the MS disease prediction role.

6 Description of the methods

In this section, we describe five machine learning methods that have been used in studying and answering the research questions. As put by [9], a machine learning system is trained rather than explicitly programmed. If we present many examples relating to the task at hand, the machine learning system learns statistical structures in the examples, or the train data set and these allow the system to develop rules that automate the task. First off, we have started with exploring the latencies for the different EPs, using two unsupervised methods. We then embark on variable selection, or Latency selection procedure that enables us to model with only the latencies that are key to the MS disability progression.

6.1 Unsupervised learning

The task at hand is surely of supervised learning, it is however important to understand the nature of the independent variables we are dealing with before bringing in the response variable. The goal of unsupervised learning is to discover patterns that might exist with in the covariates [3, 8], or the observations, so that we can then discover subgroups, if they exist and also visualize the data in a meaningful sense.

Unsupervised PCA

PCA allows us to summarize the data with a smaller number of variables that represent many variables if they are correlated. These smaller number of variables can collectively explain the variability in the original data set. Reducing the number of features can help to visualize the data at once, PCA hence finds a low-dimensional representation of the data set which contain as much information as possible. Assuming that all the individuals live in m(number of features)-dimensional space and assuming that the data is correlated, not all these dimensions will be useful. The first principal component of features $X_1, X_2, ...X_p$ is the linear combination of features

$$PC_1 = a_{11}X_1 + a_{21}X_2 + ... + a_{p1}X_p$$

with

$$\sum_{j=1}^{p} a_{j1}^2 = 1$$

that has the largest variance. The loadings on the first principle component $(a_{11}, a_{21}...a_{p1})$ define the direction in the feature space along which the data vary the most. When we have the first principal component, we then get the second principle component as a linear combination of $X_1, X_2, ...X_p$ with the greatest variance among all liner combinations of $X_1, X_2, ...X_p$ that are uncorrelated with PC_1, and so forth.

Clustering

Clustering is a tool for unsupervised learning used to partition observations into distinct subgroups so that the observations within the same cluster are similar to each other and different from those of other clusters [43, 2]. While the PCA seen in the previous section seeks to find a low dimensional representation of the data

that best explains a large percentage of the variations in the data set, clustering seeks to find subgroups that exist in the data set. There are two widely used methods of clustering: the k-means method and the hierarchical clustering. The k-means method is used when we need to specify the number of clusters in advance from the data set and the task is to separate the observations within these clusters. Hierarchical clustering on the other hand subsets the data with as many subgroups as the observations and usually it's up to the analyst to define the reasonable number of subgroups in the data set. In this study we shall hence focus on hierarchical clustering.
In this type of clustering, we choose a dissimilarity measure between the pairs of observations, and by starting with each observation as its own cluster, two clusters that are similar to each other, using a dissimilarity measure are fused together so that there are n-1 clusters. The next two clusters that are similar to each other are then fused so that there are n-2 cluster. This trend continues until all observations belong to a cluster. For this study, we have used the Euclidean distance, which often works well with continuous data as a dissimilarity measure.
While conducting hierarchical clustering, we mainly have four types of linkages. These include; the complete linkage which computes all dissimilarities between all pairs of two clusters and takes the largest of the dissimilarity, the single linkage which after computing all pairwise dissimilarities picks the smallest, the average linkage which picks the average of the dissimilarity and the centroid linkage which picks the dissimilarity between the centroids of the two subgroups. We have used complete linkage mainly because the dendrograms by this type of linkage are balanced.

6.2 Latency selection

As seen in the data description section, the data provided was symmetrical in nature with the latencies for left and right limbs (MEP and SSEP) and left or right eye (VEP) for each of the EPs.

There are many methods that can be used for variable selection. These include subset selection methods, shrinkage methods and dimension reduction methods. For this particular task, we prefer the shrinkage methods and in particular the LASSO. This is so because the LASSO is the only shrinkage methods that reduces coefficients to zero and selects latencies that only minimize the variance of the estimated coefficients, hence avoiding over fitting, and which in turn leads to better accuracy, but also keep the model as simple as possible for easy interpretation.

The LASSO selection method

Generally, assuming we have a continuous response variable, the approach of variable selection used in lasso is to minimize the quantity

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

where λ is the penalty parameter, or the tuning parameter.
The lasso uses l1 penalty which has an effect on forcing some of the coefficient estimates to zero. From the equation above, when the value of λ is zero, the lasso

procedure will select all the variables in the data set and when λ is extremely large, none of the variables will be selected. The value of λ, the tuning parameter is therefore selected wisely, using cross validation. A grid of λ values are taken and at each of these values, across validation error is calculated. The value of λ that provides the least value of cross validation error is then chosen with its selected variables. The same approach is used even when we have a dichotomous variable, like for our problem where the response is disability progression after 2 years.

6.3 The logistic regression model

Predicting categories from the available data reduces to a classification problem and in this section we tackle the basic logistic classifier as our baseline model. We shall apply this model to predict the $P_r(progression = yes|latencies)$
The logit function [19, 11] is given by the expression

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

where X=$(X_1, ..., X_p)$ are p predictors. Our problem is to predict the probability of disability progression. Re arranging the above equation gives us the following prediction function

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}{1+e^{\beta_0+\beta_1 X_1+...+\beta_p X_p}}$$

We can then use the maximum likelihood method, the more general approach used in estimating coefficient to estimate non linear coefficients $\beta_0, \beta_1, ..., \beta_p$. With maximum likelihood [6, 19], we seek to estimate $\beta_0, \beta_1, ..., \beta_p$ such that the predicted probability of disability progression for every individual in the train set is close to the individual's actual progression status. It does this by finding the values of $\beta_0, \beta_1, ..., \beta_p$ that give a probability that is close to 1 if the individual's disease progressed after two years, or a probability close to 0 if the individual's disease did not progress after two years. This is done by choosing the values that maximize the likelihood function:

$$l(\beta_0, \beta_1, ..., \beta_p) = \prod_{i:y_i=1} p(x_i) \prod_{i':y'_i=0} (1-p(x_i))$$

6.4 Tree based methods

Tree based methods have been widely studied and applied in statistics literature, like in [15, 4, 1] and many other references as good predictive methods. Tree based classification methods involve stratification of the predictor space into a number of regions [19]. Classification trees are built by first dividing the predictor space into R distinct and non-overlapping regions and then assigning the same prediction to every observation belonging to each of the regions. If we have a continuous response, the regions are build by dividing the predictor space into high dimensional rectangles and the aim is to find regions that minimise RSS. Due to computational difficulties, we instead use a top down approach called the recursive binary splitting. This

approach begins at the top of the tree and then successively splits the predictor space downwards.
While performing this approach, we select the predictors X_j and the cut-point l such that splitting the predictor space into the regions $X|X_j < l$ and $X|X_j \geq l$ leads to the greatest possible reduction in RSS.
While classification trees have very high interpretability property, they have a very poor prediction accuracy because they are not robust i.e. a small change in the data can lead to a huge change in the predicted tree.
Because of this short fall, we have used advanced tree methods here, which instead of using a single tree, they use multiple trees and then the final outcome is an aggregate of the outcome of all the multiple trees.

The Random forest model

To overcome the challenge of high variance resulting from fitting a single tree, the random forest model uses bootstrap algorithm to create different train sets from a single train set. This is achieved by sampling from the train set multiple times, and each time a tree is grown. The final outcome is the average of all the outcomes from each grown tree. This is particularly true, from the statistical point of view, we know that if we have independent observations $J_1, ..., J_n$, each with variance σ^2, then the variance of the mean is given by σ^2/n, in short, the variance will reduce if we have different samples, each with its own variance.

The problem at hand is however a classification problem, and we don't have the mean, or the variance. The same idea also applies even with classification problems, except that now, for every test observation we can record the predicted class for all the trees grown and take the majority class selected by all the samples. It turns out that we can even check for model performance with random forest models even without having to use validation approaches. With random forest, each of the bagged trees makes use of only approximately two-thirds of the data and the remaining one-third, also referred to as out-of-bag (OOB) is used as test data. At the end of the day, each observation shall have B/3 predictions, where B is the number of bootstrap samples used. We can hence tell the model performance by aggregating these predictions, if the response variable is continuous, or by taking the majority class if the problem at hand is classification, like in our case which is a valid estimate of the classification error.

Random forest is specifically different from the bagged models in the way it decorrelates the trees. Generally with bagged model, at each split in the tree, the algorithm will use all the available predictors. Unfortunately in case there are few strong predictors, with the bagged model, all the trees will have the same prediction dominated by the strong predictors and aggregating such similar trees does not have an impact on the already high variance. To overcome this challenge, Random forest model takes a sample of m predictors as candidates to consider at each split in the tree. In so doing, we don't always have the same dominating predictors at each split for all the bootstrapped samples and eventually this will have a greater impact on lowering the variance of the estimates. For the sake of the data set at hand, since we have already chosen the latencies that are key to the response and also the fact that we have few predictors for each of the models selected by the lasso model, we have used all the predictors at each split of tree growing.

The boosting model

The one other method that is worthy looking at is the boosting which also makes use of several trees, like the random forest. Specifically the boosted models do not use the bootstrap algorithm, instead trees are grown sequentially i.e. the next tree uses information given by the previous tree. The boosting approach is said to learn slowly, that is, given the current model, a decision tree is fit on the residuals and then add the new decision tree into the already fitted function to update the residuals. This approach will continuously improve the prediction slowly, since the construction of a tree is based on the previous tree as opposed to random forest models. This aspect is the key difference between random forest and boosted models and in general, slow learning statistical models tend to perform better than other models [19].
The hyperparameters for the boosted model include the B, which is estimated from cross validation to avoid overfitting, the shrinkage parameter λ which controls the learning rate and the number of splits in each tree which controls the model complexity.

6.5 The longitudinal model

The tree based methods discussed above have good performance power, however the data at hand can be seen to be of clustered nature since we have several measurements for one individual or longitudinal since one individual can have several measurements over time. We have plotted spaghetti graphs for some patients up to eight visits in figure 20 and 21 for two latencies (APB_L and the AH_R). The trend shown in these two plots is also exhibited by all the other latencies for VEP and SSEP.

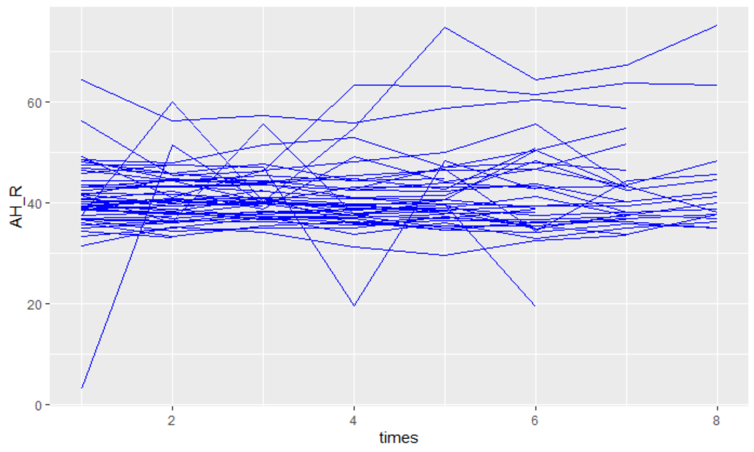


Figure 20: AH_R latency for at most eight visits for a few patients

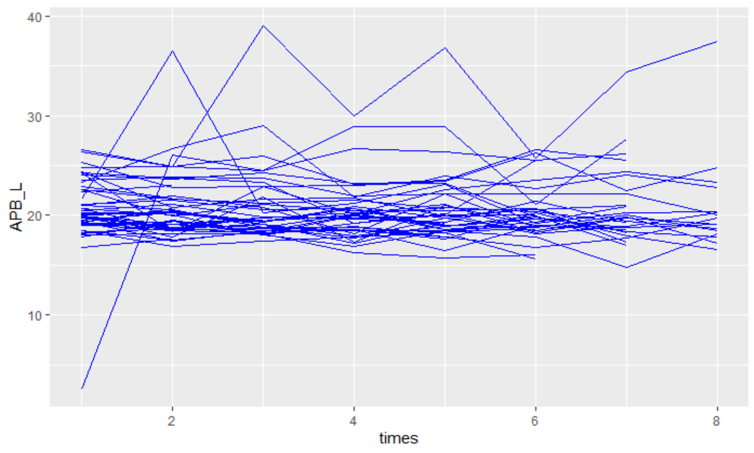


Figure 21: APB_L latency for at most eight visits for a few patients

The two spaghetti graphs display a lot of variability between the patients, we also notice variability within the patients for both latencies. This data is unique as different measurements have been taken for each patient at different time points, and each patient has their own number of measurements e.g. some patients have measurements for only two visit, while others have 8 measurements etc. Analysing this data set without factoring in it's longitudinal and clustered nature will give us poor predictions.

It is believed that [34, 12] were the first to develop methods that apply regression trees to longitudinal data, however current methods have proved better.
To get best predictions from such data, [36] suggests regression trees that take into account random effects. While its true that our data set has a categorical response and our major aim is to predict MS disability progression after two years, which is a dichotomous outcome, the model works just as well if we have numerical values of 0 and 1 representing the two categories. If we simply apply the classification trees to the data set, we simply ignore the correlation structure that exists in the data and this weakens both the modelling and the predictive ability of the classification trees seen above. Usually, mixed models [35, 14] can be used to overcome this challenge, but we have already discussed that trees can perform better than single models and [36] has generalized the mixed model to trees based methods (REEMtree), providing even better output.

6.6 The neural network model

The central task with neural networks is to successfully transform the data from it's input format to a desired output. With neural networks, the layers are the core building blocks, and the emphasis is put on learning with successive layers of continuous improvement in the data representation. These layers, stack on top of each other are the so called neural networks. All that is required for these networks are the data inputs, examples of the expected outputs, the loss function and way to measure the performance of the algorithm. The task at hand is to classify observations to either progressed or non progressed in MS disability. To achieve this, the final layer of the network is set to classify the observation to the different groups. We have provided these networks with train data sets, with the expected response values with the hope that the network will use these to find the structures that exist in the data, the so-called black box, before allowing it to predict on the test data set. For this task, we have used only three layers as model depth.

6.7 The data analysis pipeline

For all of the EPs, we have used all the above five mentioned models, but also used two approaches for each model: The first approach was to model each limb or eye and then ensemble the output from each model and EP (aggregate model). The second approach was to develop a model that consists of all the limbs, or eyes (full model). As a first step, we have scaled all latencies and baseline EDSS to have mean 0 and standard deviation (SD) 1 and applied the lasso procedure to covariates for each of the EPs and each of the approaches to select the covariates that are believed to be influential to disability progression. These are the covariates used for modelling and prediction of the disability progression by each of the models and approach.

The MEP data set for example has five variables (EDSS, AH_L, AH_R, APB_L, APB_R). For the aggregate model, we trained two models with model 1 having the variable inputs EDSS, AH_L, APB_L and model 2 having variable inputs EDSS, AH_R, APB_R. Note that these variable inputs had already been subjected to the lasso selection criteria (see table 2) which received as an input EDSS, AH_L, APB_L for the left latencies and EDSS, AH_R, APB_R for the right side latencies. The output from these two models were probabilities p1 and p2 which were averaged ((p1+p2)/2) to get one single probability for each of the test observations whose ROC AUC was studied. With the all limbs or eyes model, we subjected all the variables (EDSS, AH_L, AH_R, APB_L, APB_R) to a lasso procedure at once and trained one model on the selected variables (EDSS, AH_R, APB_R) (see table 2). The output from this model was one probability for each test set observations whose ROC AUC was studied.

For all the models and approach, we have trained 100 models by taking a sample of 70% of the data as train set and the remaining 30% as a test set. There is always a trade off between what percentage of the data to be used by the test and the train set, our percentage has been drawn from a similar study that has worked on the same data set [44] that showed that train set consisting between 70% and 80% of observations yielded the best result. For every trained model, we again use bootstrap algorithm to sample from the test data 100 times so that we have 100 predictions, and every time we study the Receiver Operating Characteristic area under the curve [31, 5]. For the longitudinal model particularly, we sampled distinct individuals to belong to either the train, or the test set with all their visits. We did not conducted parameter tuning while training the models, except with the lasso procedure. The end result from this whole step is 10000 predictions with 10000 ROC AUC, which have been used to study model performance. This pipeline has been implemented in R version 3.6.0.

7 Results

In this section, we present and interpret the results from our analysis. The section is divided into sections of different analysis steps starting with the unsupervised methods.

7.1 Unsupervised learning

We have used PCA and clustering for unsupervised analysis step, and below we present findings from the two methods for all EPs.

PCA, MEP data set

The four latencies and the EDSS score variable have been studied and the first two PCs explain 79% of the total variations in the original data set. The first principal component places most of its weights on latencies AH_L (0.469), APB_L (0.458), AH_R (0.472) and APB_R (0.456) and less weight on the baseline EDSS (0.374). The second principal component however places its weight on the EDSS feature (-0.927). in a nutshell, we can say that the first PC basically represents the latencies within the data set. From the biplot in figure 22, it is clear that all these latencies are correlated to each other.

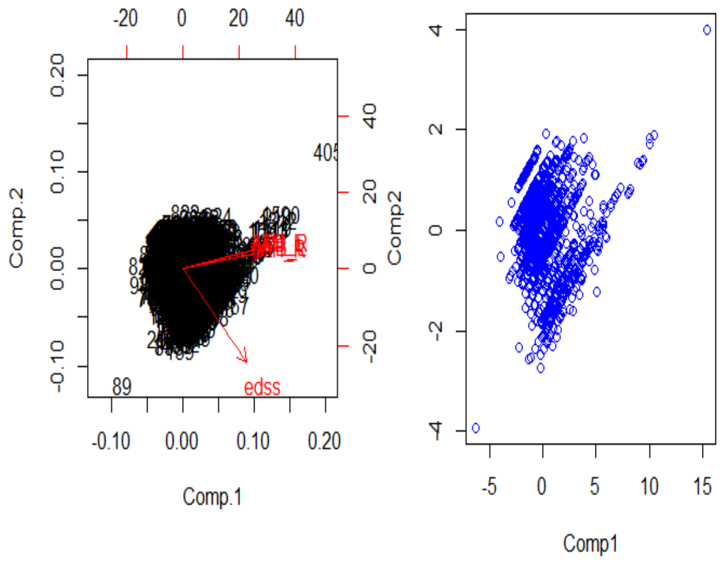


Figure 22: The association of all the MEP covariates

Important to note, from figure 22, on the biplot, the fact that baseline EDSS is very key in this data set and is significantly uncorrelated with the latencies. The biplot also gives us some information about the two categories we are dealing with. While several patients are at the centroid of the biplot, there are a few that are a bit far from the centroid. This implies that we could have groups in our data sets that are defined by the covariates hence concluding that the latencies and the baseline EDSS indeed can be used to predict disability progression. Once we reduce the data (figure 22, right panel), we notice a relationship between the two PCs implying that these two PCs can work together during supervised analysis.

Clustering, MEP data set

We have already said that clustering is particularly necessary when we want to discover subgroups in the data sets. we have in this case studied the covariates, instead of the observations. Figure 23 shows the resulting dendrogram for the MEP latencies. The baseline EDSS is in its own cluster, as expected and the latencies have been clustered together and further, the top and bottom Limbs have also been clustered separately.

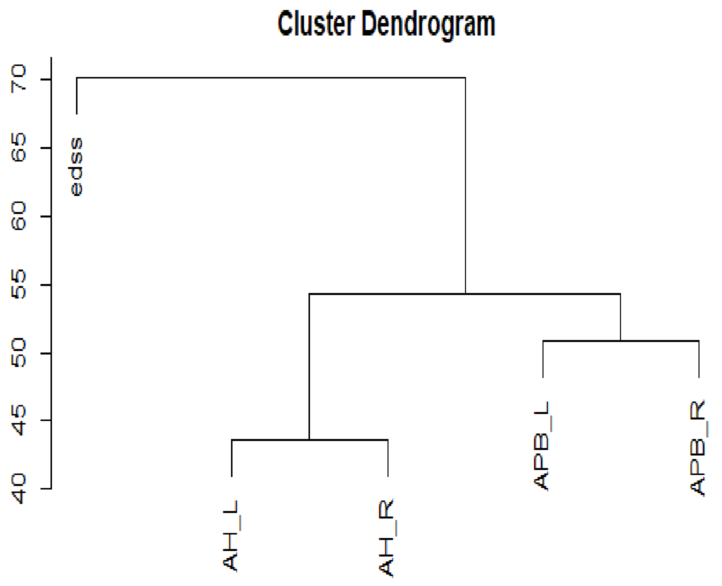


Figure 23: Dendrogram for the MEP data set covariates

In the figure 23, one can easily say that it is reasonable to study the upper and lower limbs separately and then aggregate the results for the MEP data set as these limbs are clustered together. This is as opposed to studying the left and the right limbs which we considered.

The VEP data set

The VEP dataset had 12 latencies and the baseline EDSS. After applying the PCA, the first two principal components explained 81% of the total variation in the original data set. Like in the MEP data set, the first PC has been dominated by latencies and their loadings are about the same implying possible correlation among the latencies. The second PC is also dominated by the same latencies and the EDSS only dominates the fourth principal component.

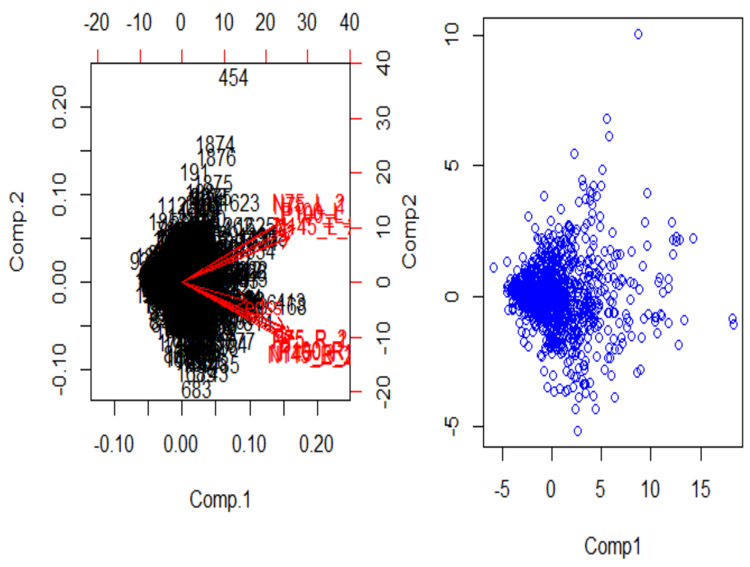


Figure 24: The association of all the VEP covariates

In figure 24 above on the left panel we continue to confirm that the two PCs are dominated by the latencies. Unlike in the MEP data set, there seems to be correlation among latencies regardless of the eye. The biplot in the left panel shows that there could be groups in the data. Individuals far from the center could belong to a certain group different from that of individuals at the center of the biplot. The idea of using these latencies and baseline EDSS to predict disability progression could therefore be valid.

Clustering, VEP data set

In the figure 25, its interesting to see that the latencies under the categories N145 P100 and N75 can be grouped together regardless of the eye.

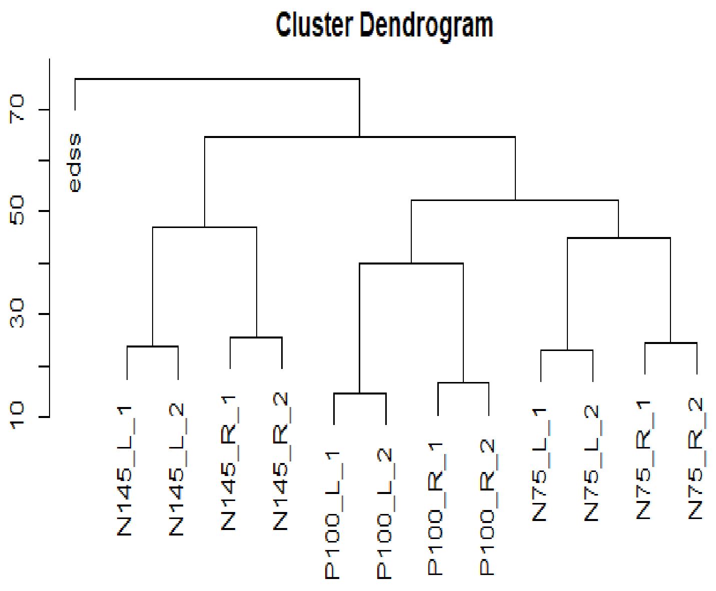


Figure 25: Vep data set Dendrogram

In predicting MS disability progression therefore, it is not reasonable to study latencies from the two eyes separately since for example we note that latencies N145_L_1 and N145_R_1 can belong to one cluster even though they belong to different eyes, as evident in figure 25. Instead, it will be reasonable to look at the N145, P100 and N75 group of latencies separately regardless of the eye, and then ensemble the results.

The SSEP data set

As in the VEP data set, the SSEP also tends to have the same characteristics that are similar. For example both the first and the second PCs are dominated by latencies and the baseline EDSS dominates the fourth PC. Unlike the other data sets, the first two PCs only explain 70% of the total variations in the data.

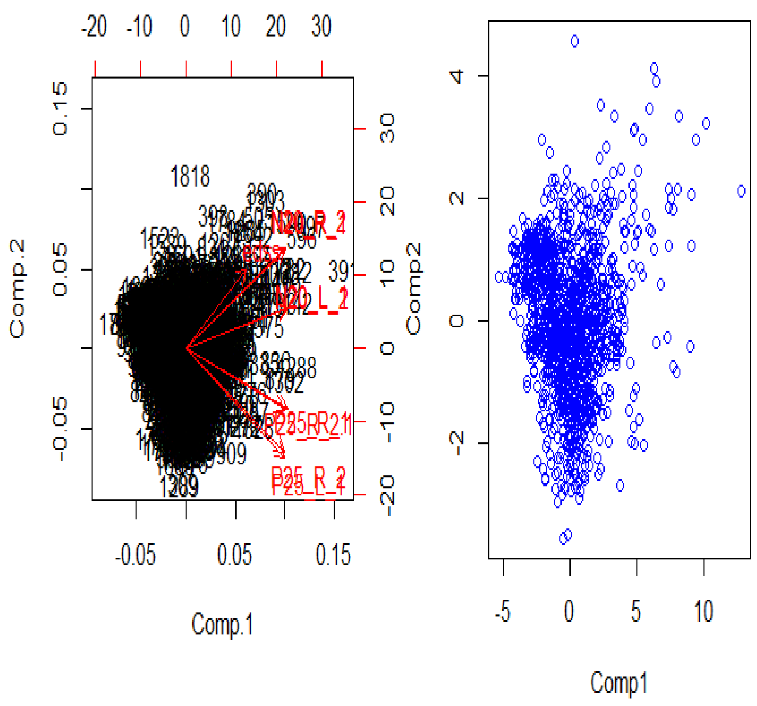


Figure 26: The association of all the SSEP covariates

The data presented in figure 26 is very unique. Some latencies on the left limb are correlated with some latencies on the right limb and vise versa, studying separate limbs may not bring the best result in such a case. In the same figure, like the other EPs, a conclusion can be reached that there appears to be groups defined by the latencies and the baseline EDSS. Individuals that are a bit far from the center might belong to a certain group while others that are at the center of the biplot, belong to another group, hence the idea of using these covariates in predicting disability progression is valid.

Clustering, SSEP data set

Like the other EPs, we have two main clusters, the baseline EDSS and the latencies. This is also evident in figure 27. If we're to have three clusters in this data set, we would have to cluster the N20, P25 group of latencies together regardless of the limb.

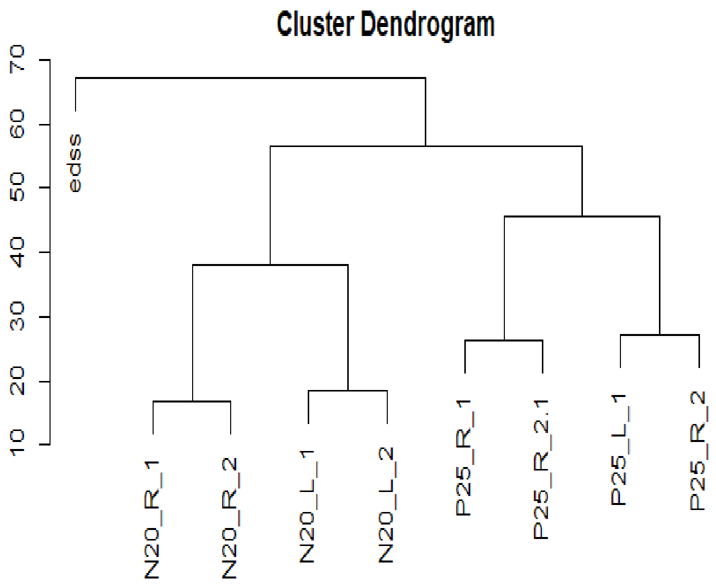


Figure 27: SSEP data set Dendrogram

Figure 27 shows that to predict disability progression, it may be necessary to study the N20 and P25 group of latencies separately regardless of the limb, since these cluster together.

7.2 Latency selection

When we apply the Lasso procedure, we have the latencies selected as shown in table 2. We have already explored the data sets using the unsupervised methods, indeed for the MEP data set, all the latencies face in the same direction, and that EDSS was a key feature. When we feed all the latencies for all limbs to the lasso procedure at once, only variables from the right limb and the baseline EDSS are selected.

EP	Left side Latencies	Right side Latencies	All Latencies
MEP	AH_L, APB_L, EDSS	AH_R, APB_R, EDSS	AH_R, APB_R, EDSS
VEP	N75_L_1, P100_L_1, EDSS	N75_R_1, P100_R_2, EDSS	N75_L_1, N75_R_1, P100_R_2, EDSS
SSEP	EDSS	P25_R_2.1, EDSS	P25_R_2.1, EDSS

Table 2: The selected variables from the lasso step

With the VEP data set, the variables selected are also shown in the table 2. Subjecting all latencies to the lasso procedure for the VEP data set gave us variables from both eyes, and EDSS.

During the SSEP data set exploratory analysis, the EDSS feature dominated the 4^{th} component, which contributed very less to the total variation in the data, yet it's key for disability progression prediction. The results here show that only one latency from the right limb is necessary for prediction together with baseline EDSS.

Our results suggest that both the AH and APB latencies are important while used with baseline EDSS to predict disability progression with MEP. Among all the three latencies for the VEP, neurologists expect the P100 latency to be more important. Indeed as shown in table 2, the P100 latency has not only been very important for both the left and the right eyes, but also important when all variables are considered. Our results suggest that in addition to the P100 latency, the N75 latency is equally very important in MS disability progression prediction. Between the two SSEP latencies (N20 and P25), the N20 is expected to be more important. Our results however have shown that this latency is not very useful in MS disability progression prediction. In the presence of baseline EDSS, the other latency that can be used for prediction is P25.

Important to note is the fact that the lasso was applied to the entire data set, forcing it to choose features that are important to both the train and the test set. This could lead to overfitting, and its possible that other features could have been selected if we had used a separate train set.

7.3 Which machine learning model best predicts MS disability progression after 2 year?

For every EP, we have ten models and the task at hand is to identify the best model among all the ten. Even though we can already tell, without statistically testing, from table 3 that the longitudinal models have the highest average ROC area under the curve and hence out performs all the others (effects size with the baseline logistic and the second best RF models of (0.29, 0.235), (0.302, 0.243) and (0.342, 0.265) for MEP, VEP and SSEP respectively), we go ahead to test statistically if these models are statistically different from all the others.

Model	MEP	VEP	SSEP
Logistic, aggregate	0.679(0.06)	0.661(0.07)	0.623(0.08)
Logistic, all limbs or eyes	0.687(0.06)	0.665(0.07)	0.625(0.07)
RF, aggregate	0.732(0.05)	0.720(0.06)	0.702(0.06)
RF, all limbs or eyes	0.703(0.05)	0.705(0.06)	0.694(0.06)
Boosting, aggregate	0.721(0.05)	0.708(0.06)	0.693(0.06)
Boosting, all limbs or eyes	0.717(0.05)	0.702(0.06)	0.685(0.06)
Neural network, aggregate	0.693(0.06)	0.682(0.07)	0.679(0.06)
Neural network, all limbs or eyes	0.693(0.06)	0.682(0.07)	0.679(0.06)
Longitudinal model, aggregate	0.967(0.013)	0.963(0.013)	0.967(0.012)
Longitudinal model, all limbs or eyes	0.967(0.013)	0.963(0.013)	0.967(0.012)

Table 3: Summary results (mean (SD), ROC AUC) for all models and EPs studied.

We have hence made the longitudinal model that uses all the latencies (longitudinal model, all limbs or eyes) as the baseline model and compared it with all the others. However, because there are 9 tests performed, we need to take into consideration the multiplicity correction by applying both the family wise error rates (bonferroni and holm) procedures and the false discovery rate (FDR).

For all the applied models, only one model can be equalled to the baseline model, and this is the longitudinal model, aggregate which averages predictions from longitudinal model built with the left latencies and that built with the right latencies. Results suggest that the null hypothesis that the baseline model is the same as the other models has been rejected 8 times and accepted once for the longitudinal model, aggregate.
With the bonferroni procedure, the α level of 0.05 was divided with 9 and individual p_values were compared with $\frac{\alpha}{9}$. Models with p_values less than $\frac{\alpha}{9}$ were then considered statistically significant. The holm procedure is conducted in such a way that the 9 p_values are arranged in ascending order, the very first p_value is then compared with $\frac{\alpha}{9}$. If the p_value is less than $\frac{\alpha}{9}$, reject the null hypothesis there is no difference between the two models, else stop and conclude that none of the models are different. This is done for every p_value, every time the denominator (number of p_values) reduces by one. This approach is considered more powerful than the bonferroni procedure. However with the data at hand, the two methods identified that all the 8 models were different from the baseline model.

The two approached looked at are considered very conservative. The third method that is better than the two as far as power is concerned is the false discovery rate (FDR) method. Assuming we have m p_values, the FDR procedure is that arrange all the m p_values in ascending order, and, starting with the largest p_value, compare the ith p_value with the item (i*α)/m. if the ith p_value is greater than the calculated item, stop and make conclusions.

All the approaches in this analysis, including the test where we do not use the multiplicity test (Raw p-value) fortunately agreed that the 8 p_values were statistically different from the baseline model out of the 9 models. We therefore rejected the 8 hypotheses that the baseline model is equal to all models except the longitudinal model, aggregate. The biobase and multitest R packages have been used for the multiplicity analysis. We have done this for all the EPs and the baseline model out performs all the others in all EPs and in all cases its only equal to the aggregate model of the longitudinal type.

7.4 Comparison of aggregate and full models

In the figures 28 29 and 30, we have plotted the ROC AUC for the aggregate and full models, we also show the means of the two models respectively on top of each graph for all the models and EPs. The figures suggest that the means of both the aggregate and full model for the logistic, random forest and boosting models are not the same while those of the neural and longitudinal models seem to be the same for all the EPs.

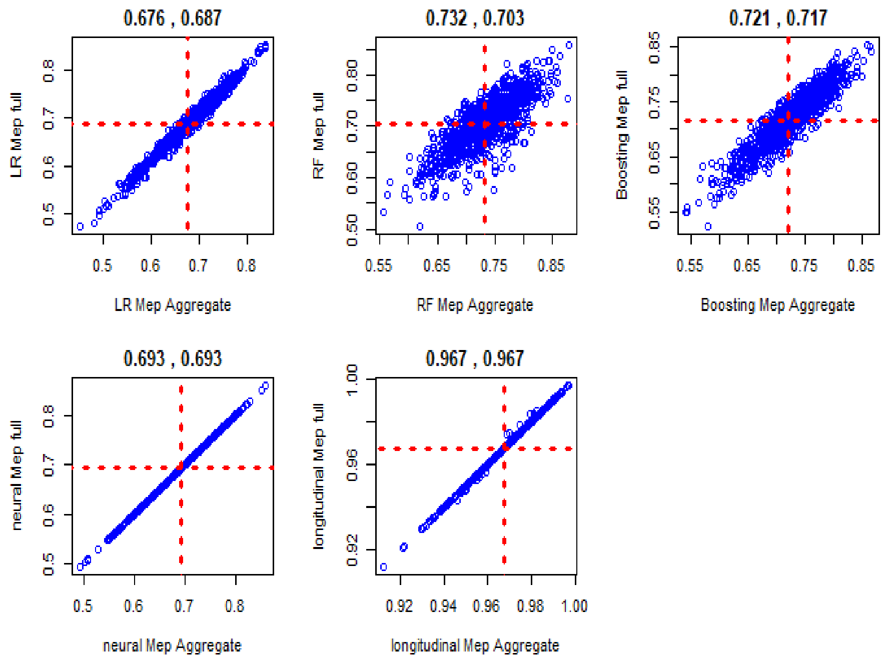


Figure 28: A graphical presentation of the aggregate model verses the full model, MEP

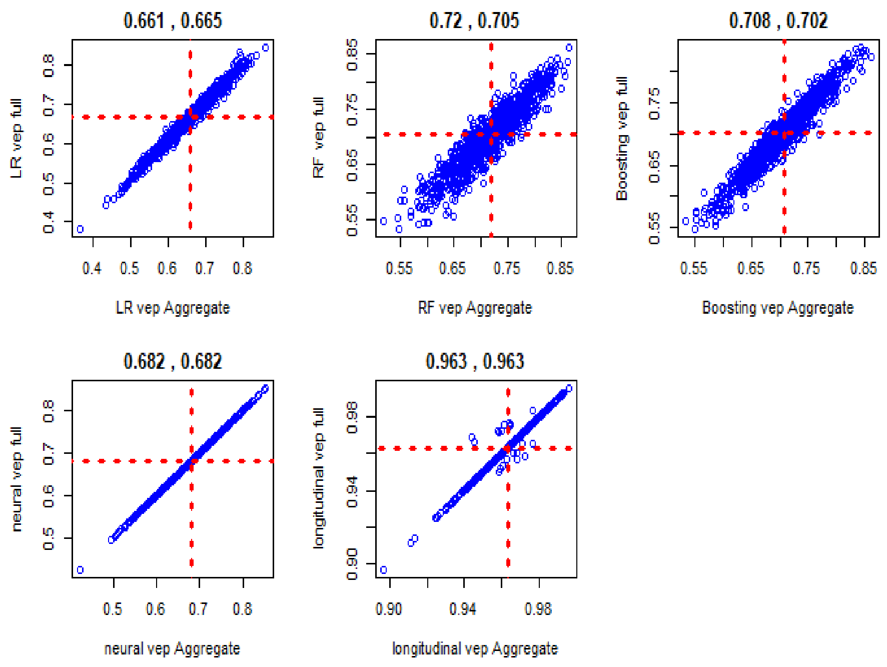


Figure 29: A graphical presentation of the aggregate model verses the full model, VEP

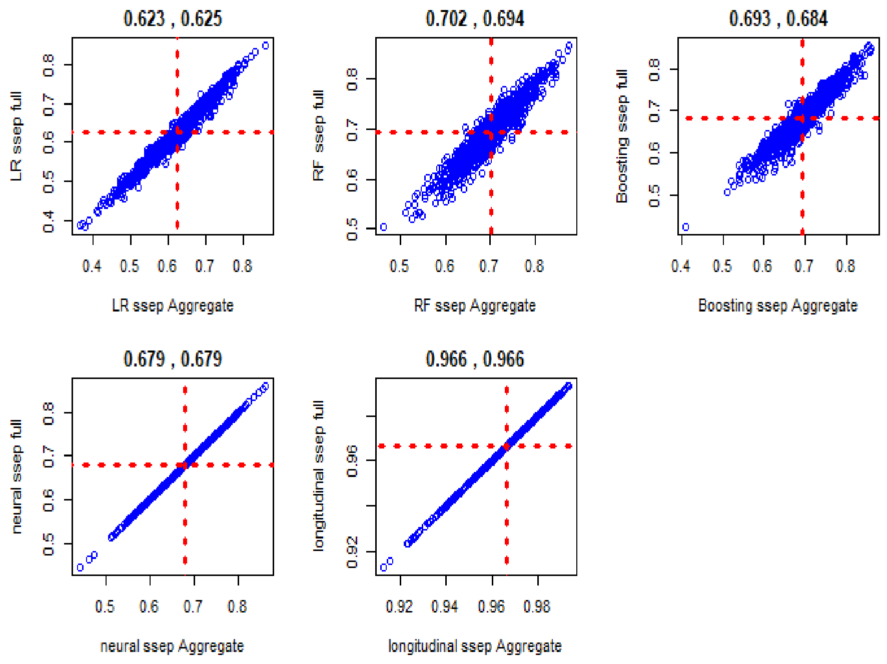


Figure 30: A graphical presentation of the aggregate model verses the full model, SSEP

The effects size between the aggregate and full models for all models and EPs is very small. The largest effects size among these is 0.03 belonging to the random forest model with MEP, implying that generally the aggregate and full models for all models and EPs are quite similar. We have however tested the null hypothesis that the aggregate and full models are the same, using the t test and the p_values are presented in the table 4. With the logistic model for MEP, the model that uses all latencies during analysis gives better results than when the left and the right side latencies are analysed separately.

When the VEP and SSEP data sets are used for the LR model, there is no evidence to reject the null that both models are similar and so any of the models could be used in these two data sets. With random forest, there was evidence that the aggregate model is better than the random forest full model at 0.05 level of significance for all the EPs. With boosting model, while using the MEP data set, it does not matter which model to use, but it matters with the VEP and the SSEP models where the aggregate model would yield better results compared to the full model at 0.05 level of significance.

For the neural and longitudinal models, there was no evidence against the null hypothesis that the aggregate and the full model are the same. It therefore does not matter whether we analyse any data set for any EP using the aggregate, or the full models of the neural (p_values ≥ 0.97) or the longitudinal model (p_values ≥ 0.92).

Model	**MEP**	**VEP**	**SSEP**
Logistic	0.000	0.150	0.630
Random Forest	0.000	0.000	0.000
Boosting	0.060	0.020	0.000
Neural	0.990	0.970	0.970
Longitudinal	0.990	0.920	1.000

Table 4: The p_values resulting from testing the equality of aggregate and full models with all EPs.

Since the longitudinal model is our best model, and we have shown that it does not matter the limbs or eyes analysis, we conclude that analysing the limbs or eyes separately will not improve the MS disability predictive performance of the model for all EPs.

7.5 Which EP provides best predictions for MS disability progression after two years?

From the earlier section we have shown that the longitudinal model outperforms all the others for all EPs. From the results of table 3, the average AUC defined by this model is very good and almost similar across all EPs (maximum effect size of 0.004). We have however gone ahead to test the null hypothesis that MEP is similar to VEP and SSEP and the p_values after multiplicity correction are presented in table 5.

Model	**MEP**	**VEP**	**SSEP**
Logistic, Aggregate	1	0.000	0.000
Logistic, all limbs or eyes	1	0.000	0.000
RF, Aggregate	1	0.000	0.000
RF, all limbs or eyes	1	0.760	0.000
Boosting, Aggregate	1	0.000	0.000
Boosting, all limbs or eyes	1	0.000	0.000
Neural network, Aggregate	1	0.000	0.000
Neural network, all limbs or eyes	1	0.000	0.000
Longitudinal model, Aggregate	1	0.000	0.073
Longitudinal model, all limbs or eyes	1	0.000	0.071

Table 5: Bonferroni adjusted p_values from t test done to test the equality of models across EPs with the MEP as baseline EP

The results presented in the above table, and in conjunction with the results of table 3 show that for most of the models, the different EPs have varying performance while predicting MS disease after two years. With the logistic model, both the aggregate and full models will best predict MS disease in the order of the MEP, VEP and SSEP. The same trend is followed with the RF, aggregate model. However With the RF, full model, there was no evidence against the null hypothesis that the MEP and the VEP data sets produce the same results. But the evidence was found with the SSEP. While using the RF, full model, one gets similar results when they use the VEP and the MEP, but gets worse results with the SSEP. Both the boosting and the neural network models produce different results with the different EPs in the performance descending order of MEP, VEP and SSEP. With the longitudinal model however, there was no evidence against the null hypothesis that the MEP and SSEP are similar for both the aggregate and full models. However there was evidence that MEP and SSEP are both superior to the VEP.

8 Discussion

We had four research questions for this work. We set out to find which latencies are key in conjunction with the baseline EDSS in predicting MS disability progression, which machine learning technique best predicts MS disability progression after two years, we also wanted to find which EP produces best predictions when used by a machine learning technique to predict disability progression, and the last quest was to find out if its really necessary to take into account the symmetrical property in the data and analyse specifically the left and the right sides latencies separately. This study is unique in its own style, majority studies have studied one EP, but we have had a chance to study three EPs in predicting the MS disability progression after two years. The cut off of two years is common in literature, but the idea is to predict the unresponsiveness in the treatment as early as possible so we can alter the treatment and hence we can always alter the duration (not always 2 years) depending on how early we want to detect treatment failure. While we have used several EPs, the patients in one EP are not necessarily the same patients studied in the other EP.

There have been debates as to whether EP has the necessary information that can be used in predicting MS, with some studies for and some studies against. While some studies have basically used the features extracted from the EPs, latencies, EDSS at baseline, gender and age, this study only used the latencies and the baseline EDSS to predict disability progression. [44] were the first to study MS with all the above literature variable, and also putting into consideration the time series data to improve the prediction, but only for the motor evoked potential.
Even though these studies have had all the data available, non of them considered the clustered longitudinal nature in the data during analysis. We have studied these data using most of the routine machine learning techniques, but also considered the clustered nature in the data and applied the longitudinal REEMtree model.

Our results show that all the latencies for MEP are important in MS disease prediction. In addition to the P100 latency which is expected by neurologists to be very important for the VEP, our results show that latency N75 is equally important in MS disability progression prediction. While neurologists expect N20 to be the most important latency for SSEP, there was no evidence to support this argument. Instead the P25 latency measured on the right side was selected to be very important as opposed to N20. Our results further show that the model that considers the clustered nature in the data, REEMtree emerges the best machine learning model for predicting disability progression. This best model had an roc AUC of 0.967 with the MEP and SSEP dataset and 0.963 with the VEP dataset. These results outperform the basic logistic classifier significantly by AUC of 0.29, 0.34 and 0.30 with the MEP, SSEP and VEP data sets respectively. This is one of the best and promising results realised ever. Even though [44] used time series data in addition to the literature variables, the best model achieved had an average AUC performance of 0.75 for the MEP data set. It may not be necessary to include all of the variables there is about MS, in fact our performance was reached after reducing the data by applying the lasso and the predictions were made using not more that three covariates for all the EPs with the baseline EDSS covariate in every model. The results without longitudinal modeling with the MEP data set also agreed with [44] that the random forest classifier (AUC 0.732) outperforms the baseline logistic classifier (AUC 0.679) significantly.

While the boosting model is theoretically superior to random forest, the RF model outperformed it for all the EPs. This could be a result of overfitting, the boosted model suffers overfitting if the parameter tuning is not done to select the appropriate number of trees to get the best performance. During analysis of this work, we have only performed hyper parameter tuning with the lasso model to select the best that gives us the least cross validation error. We performed feature selection on the entire data set, resulting into features that are important to both the train and the test set to be selected. This is a form of data leakage that can cause overfitting. Hyper parameter tuning was however not done for the other models discussed in this work and we used the train-test approach which does not support data leakage. Our data was enough to train models with this approach, evidenced by the small standard deviations resulting from 10000 ROC AUC as shown in table 3.
In machine learning, and specifically deep learning methods, the idea is for the machine to learn the rules that best represent the response variable in the data and its always advantageous to have as much data as possible. Much as we have tried to keep as much data as possible, by imputing the missing data with a random latency value, it did not make the neural net work model train well on the data. For all the EPs, the results from the neural network model are only second to the baseline logistic classifier.

On the question of which EP is best among the three, in theory, the neurologists often prefer the order of MEP, SSEP and VEP. According to our results, we have shown that the longitudinal model yields almost the same AUC for all EPs as represented in table 3. Our results further show that with majority of the models, the MEP is superior to all EPs, followed by VEP and lastly the SSEP. With our best model however, it is reasonable to put it that we always get similar conclusions regarding MS prediction after two years when we use both the MEP and the SSEP. This result, obtained with the longitudinal model concurs with the neurologists argument.

In literature, some studies have written about the need to take advantage of the symmetrical nature of the EP data set by looking at the left and the right limbs separately for the MEP and SSEP data sets and the left and the right eye for the VEP data set to finally ensemble the two models to better predict MS disability progression. Our results suggest that with our best model (longitudinal model), one does not have to go an extra mile of looking at the latencies for the two sides separately during analysis regardless of the EP being used. This result has also been tested to be true with the neural network models as shown in table 4. In future, when we have a lot of data on MS and we want to use neural network model, we shouldn't have to do separate analysis for the limbs or eyes before reaching a conclusion. This conclusion is however not correct when non longitudinal models are used. Using the random forest classifier demands that the limbs, or eyes are studied separately to get better final outcomes.

9 Conclusion

Multiple sclerosis is still a chronic disease with no cure affecting so many people especially in Europe and north America. Being able to know in advance the progression of the disease two years earlier will have a greater impact on patients health in managing the disease. Unfortunately, up to now, predicting MS disability progression has been a challenge. This research has explored a number of machine learning techniques that could support better prediction. We have used three EPs, namely motor evoked potential, the visual evoked potential and somatosensory evoked potential from a large data set for patient already undergoing treatment gathered by Rehabilitation & MS center in Overpelt, Belgium. We have maintained patient's visits with two years followups. Unlike other studies that have used all features extracted from the EPs, including latencies, peak-to-peak amplitude and dispersion pattern, we have only focused on latencies, in conjunction with the EDSS for this task.

First off, our results show that non linear classifiers out perform the linear logistic classifiers which have mostly been used in many recent papers. Among the non linear classifiers, the random forest classifier (AUC 0.732) out performed the linear classifier (AUC 0.679) significantly.
Our results further show that it is not enough to use non linear classifiers, models that take advantage of the clustered nature in the data yield better results by far when used with non linear classifiers. We have particularly used regression trees with random effects model to effectively predict disability progression with all EPs. This model has out performed all the other models significantly with all the EPs. Specifically, with the MEP data set, it has achieved the AUC of 0.967 ± 0.013 outperforming the second best RF model (0.732 ± 0.05) and the baseline logistic model (AUC $0.687\pm$ 0.06). With the VEP data set, it has achieved the AUC of 0.963 ± 0.013 outperforming the second best RF model (0.72 ± 0.06) and the baseline logistic model (AUC 0.661 ± 0.07). Finally it has achieved the AUC of 0.967 ± 0.012 with the SSEP data set outperforming the second best RF model (0.702 ± 0.06) and the baseline logistic model (AUC 0.623 ± 0.08).

Neurologists prefer the MEP to SSEP and VEP. With our best model, the results show that the MEP is only superior to VEP (p_value 0.000), it's predictive performance is similar to that of the SSEP (p_value 0.07) at 0.05 level of significance. The MEP data set however seems to be superior to VEP and the VEP is also superior to SSEP when using the RF and LR models. While this is true, it's important to note that the MEP had more final observations compared to either VEP or SSEP. This definitely made it easier to train a machine learning model with it than the other EPs and this could have contributed to it's good performance as compared to other EPs.
Finally, previous studies have preferred to study limbs (MEP) separately for better predictions. Even though our results agree that this is true for the RF classifier for all EPs (p_value 0.000), logistic classifier for the MEP data set (P_value 0.000) and boosting for the VEP and SSEP (P_values 0.02 and 0.000 respectively) data sets, it is not necessary while using the longitudinal, or the neural network models (P_value ≥ 0.92).

10 Ideas for future research

From the clustering section of this work, it was interesting to see that the latencies from the upper limb and the latencies from the lower limb for the MEP data set cluster separately. A certain trend is also seen from the other two EPs that latencies always cluster together regardless of the side. For non longitudinal models, it would be interesting to also think about analysing the data with each cluster separately and ensemble the results for better prediction.

The missing data in this research was handled in the simplest way possible. This could have been done better with multiple imputation methods which could further improve performance of MS predictive models. Future research can focus on using better imputation methods.

It would be enriching to have all the other literature variables in the data set. This was a predictive problem, but it would be interesting to see how other variables like age gender etc. relate with MS disability progression.

Finally, the longitudinal model out performed all the other models, may be because no hyper parameter tuning was done for the other models and its not required for the longitudinal model. Future research should study if tuning the hyper parameters improves the MS disability prediction.

Bibliography

[1] Richard Berk. An introduction to statistical learning from a regression perspective. In *Handbook of Quantitative Criminology*, pages 725–740. Springer, 2010.

[2] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[3] Josep Lluis Berral-García. A quick view on current techniques and machine learning algorithms for big data analytics. In *2016 18th international conference on transparent optical networks (ICTON)*, pages 1–4. IEEE, 2016.

[4] Hamparsum Bozdogan. Improving the performance of radial basis function (rbf) classification using information criteria zhenqiu liu and hamparsum bozdogan university of tennessee, knoxville, usa. In *Statistical Data Mining and Knowledge Discovery*, pages 225–248. Chapman and Hall/CRC, 2003.

[5] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[6] J Brooks-Bartlett. Probability concepts explained: Maximum likelihood estimation. *Retrieved from Towards Data Science: https://towardsdatascience.com/probability-conce pts-explained-maximum-likelihood-estimation-c7b4 342fdbb1*, 2018.

[7] Paul Browne, Dhia Chandraratna, Ceri Angood, Helen Tremlett, Chris Baker, Bruce V Taylor, and Alan J Thompson. Atlas of multiple sclerosis 2013: a growing global problem with widespread inequity. *Neurology*, 83(11):1022–1024, 2014.

[8] Jason Brownlee. Supervised and unsupervised machine learning algorithms. *Machine Learning Mastery*, 16(03), 2016.

[9] Francois Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[10] Christian Confavreux and Sandra Vukusic. Natural history of multiple sclerosis: a unifying concept. *Brain*, 129(3):606–616, 2006.

[11] Jan Salomon Cramer. The origins of logistic regression. 2002.

[12] Glenn De'Ath. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117, 2002.

[13] Alberto Gajofatto, Massimiliano Calabrese, Maria Donata Benedetti, and Salvatore Monaco. Clinical, mri, and csf markers of disability progression in multiple sclerosis. *Disease markers*, 35(6):687–699, 2013.

[14] Ahlem Hajjem, François Bellavance, and Denis Larocque. Mixed effects regression trees for clustered data. *Statistics & probability letters*, 81(4):451–459, 2011.

[15] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[16] Martin Hardmeier, Florian Hatz, Yvonne Naegelin, Darren Hight, Christian Schindler, Ludwig Kappos, Margitta Seeck, Christoph M Michel, and Peter Fuhr. Improved characterization of visual evoked potentials in multiple sclerosis by topographic analysis. *Brain topography*, 27(2):318–327, 2014.

[17] Frank E Harrell Jr and Maintainer Frank E Harrell Jr. Package 'hmisc'. *CRAN2018*, pages 235–6, 2015.

[18] Hironobu Hayashi and Masahiko Kawaguchi. Intraoperative monitoring of the brain. In *Textbook of Neuroanesthesia and Neurocritical Care*, pages 43–61. Springer, 2019.

[19] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[20] Tomas Kalincik, Gary Cutter, Tim Spelman, Vilija Jokubaitis, Eva Havrdova, Dana Horakova, Maria Trojano, Guillermo Izquierdo, Marc Girard, Pierre Duquette, et al. Defining reliable disability outcomes in multiple sclerosis. *Brain*, 138(11):3287–3298, 2015.

[21] Hanni SM Kiiski, Sinead Ni Riada, Edmund C Lalor, Nuno R Goncalves, Hugh Nolan, Robert Whelan, Roisin Lonergan, Siobhan Kelly, Marie Claire O'Brien, Katie Kinsella, et al. Delayed p100-like latencies in multiple sclerosis: A preliminary investigation using visual evoked spread spectrum analysis. *PloS one*, 11(1), 2016.

[22] Gisela Kobelt, Alan Thompson, Jenny Berg, Mia Gannedahl, Jennifer Eriksson, MSCOI Study Group, and European Multiple Sclerosis Platform. New insights into the burden and costs of multiple sclerosis in europe. *Multiple Sclerosis Journal*, 23(8):1123–1136, 2017.

[23] John F Kurtzke. Rating neurologic impairment in multiple sclerosis: an expanded disability status scale (edss). *Neurology*, 33(11):1444–1444, 1983.

[24] Letizia Leocani, Simone Guerrieri, and Giancarlo Comi. Visual evoked potentials as a biomarker in multiple sclerosis and associated optic neuritis. *Journal of Neuro-Ophthalmology*, 38(3):350–357, 2018.

[25] Letizia Leocani, Marco Rovaris, Filippo Martinelli Boneschi, Stefania Medaglini, Paolo Rossi, Vittorio Martinelli, Stefano Amadio, and Giancarlo

Comi. Multimodal evoked potentials to assess the evolution of multiple sclerosis: a longitudinal study. *Journal of Neurology, Neurosurgery & Psychiatry*, 77(9):1030–1035, 2006.

[26] Roland Martin, Henry F McFarland, and JM Boggs. Immunological aspects of experimental allergic encephalomyelitis and multiple sclerosis. *Critical reviews in clinical laboratory sciences*, 32(2):121–182, 1995.

[27] V Martinelli, G Dalla Costa, MJ Messina, Giovanni Di Maggio, Francesca Sangalli, L Moiola, M Rodegher, B Colombo, R Furlan, L Leocani, et al. Multiple biomarkers improve the prediction of multiple sclerosis in clinically isolated syndromes. *Acta Neurologica Scandinavica*, 136(5):454–461, 2017.

[28] Xavier Montalban, Ralf Gold, Alan J Thompson, Susana Otero-Romero, Maria Pia Amato, Dhia Chandraratna, Michel Clanet, Giancarlo Comi, Tobias Derfuss, Franz Fazekas, et al. Ectrims/ean guideline on the pharmacological treatment of people with multiple sclerosis. *Multiple Sclerosis Journal*, 24(2):96–120, 2018.

[29] Marc R Nuwer, James W Packwood, Lawrence W Myers, and George W Ellison. Evoked potentials predict the clinical changes in a multiple sclerosis drug study. *Neurology*, 37(11):1754–1754, 1987.

[30] J Vernon Odom, Michael Bach, Colin Barber, Mitchell Brigell, Michael F Marmor, Alma Patrizia Tormene, Graham E Holder, et al. Visual evoked potentials standard (2004). *Documenta ophthalmologica*, 108(2):115–123, 2004.

[31] Seong Ho Park, Jin Mo Goo, and Chan-Hee Jo. Receiver operating characteristic (roc) curve: practical review for radiologists. *Korean journal of radiology*, 5(1):11–18, 2004.

[32] R Pelayo, X Montalban, T Minoves, D Moncho, J Rio, C Nos, C Tur, J Castillo, A Horga, M Comabella, et al. Do multimodal evoked potentials add information to mri in clinically isolated syndromes? *Multiple Sclerosis Journal*, 16(1):55–61, 2010.

[33] Sudarshini Ramanathan, Kerry Lenton, Therese Burke, Lavier Gomes, Karen Storchenegger, Con Yiannikas, and Steve Vucic. The utility of multimodal evoked potentials in multiple sclerosis prognostication. *Journal of Clinical Neuroscience*, 20(11):1576–1581, 2013.

[34] Mark Robert Segal. Tree-structured methods for longitudinal data. *Journal of the American Statistical Association*, 87(418):407–418, 1992.

[35] Rebecca J Sela and Jeffrey S Simonoff. Re-em trees: a new data mining approach for longitudinal data. 2009.

[36] Rebecca J Sela and Jeffrey S Simonoff. Re-em trees: a data mining approach for longitudinal and clustered data. *Machine learning*, 86(2):169–207, 2012.

[37] Mireia Sospedra and Roland Martin. Immunology of multiple sclerosis. *Annu. Rev. Immunol.*, 23:683–747, 2005.

[38] Ewa Stawowczyk, Krzysztof Piotr Malinowski, Paweł Kawalec, and Paweł Moćko. The indirect costs of multiple sclerosis: systematic review and meta-analysis. *Expert review of pharmacoeconomics & outcomes research*, 15(5):759–786, 2015.

[39] Kate Tilling, Michael Lawton, Neil Robertson, Helen Tremlett, Feng Zhu, Katharine Harding, Joel Oger, and Yoav Ben-Shlomo. Modelling disease progression in relapsing-remitting onset multiple sclerosis using multilevel models applied to longitudinal data from two natural history cohorts and one treated cohort. *Health Technology Assessment (Winchester, England)*, 20(81):1, 2016.

[40] Mar Tintoré, A Rovira, J Rio, C Tur, R Pelayo, C Nos, N Téllez, H Perkal, M Comabella, J Sastre-Garriga, et al. Do oligoclonal bands add information to mri in first attacks of multiple sclerosis? *Neurology*, 70(13 Part 2):1079–1083, 2008.

[41] P Walsh, N Kane, and S Butler. The clinical role of evoked potentials. *Journal of neurology, neurosurgery & psychiatry*, 76(suppl 2):ii16–ii22, 2005.

[42] Brian G Weinshenker. The natural history of multiple sclerosis: update 1998. In *Seminars in neurology*, volume 18, pages 301–307. © 1998 by Thieme Medical Publishers, Inc., 1998.

[43] Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193, 2015.

[44] Jan Yperman, Thijs Becker, Dirk Valkenborg, Veronica Popescu, Niels Hellings, Bart Van Wijmeersch, and Liesbet M Peeters. Machine learning analysis of motor evoked potential time series to predict disability progression in multiple sclerosis. *BMC neurology*, 20(1):1–15, 2020.

11 Appendix

```
## importing the required R libraries
library (reshape)
library (ISLR)
library(Hmisc)
library (glmnet)
library (boot)
library (gbm)
library (randomForest)
library(AUC)
library(neuralnet)
library(REEMtree)
library(Biobase)
library(multtest, verbose = FALSE)
require(ggplot2)

## importing the data
edss <- read.csv(file= "G:/My Drive/Noah/data/edss.csv", header=TRUE
                 , sep="," )
mep <- read.csv(file= "G:/My Drive/Noah/data/mep.csv", header=TRUE,
                sep="," )
ssep_lower <- read.csv(file= "G:/My Drive/Noah/data/ssep_lower.csv",
                       header=TRUE, sep="," )
ssep_upper <- read.csv(file= "G:/My Drive/Noah/data/ssep_upper.csv",
                       header=TRUE, sep="," )
vep <- read.csv(file= "G:/My Drive/Noah/data/vep.csv", header=TRUE,
                sep="," )
## function to output missing data percentage
project_missing <- function(p_data){
  all.mis = matrix(0,nrow=dim(p_data)[2], ncol = 3)
  names.project = colnames(p_data)
  for(colmns in 1:dim(p_data)[2]){
    coluum <- p_data[,colmns]
    if(class(coluum) != "factor"){
      col.miss <- coluum[coluum < 0]
    }
    else
      col.miss <- coluum[coluum == ""]
    all.mis[colmns, 1] <- names.project[colmns]
    all.mis[colmns, 2] <- length(col.miss)
    all.mis[colmns, 3] <- round((length(col.miss)/(dim(p_data)[1]))
                                *100, digit = 2)
  }
  all.mis.names = c("Variable", "Missing", "Percentage")
  colnames(all.mis) <- all.mis.names
  all.mis
}
```

```
# dealing with missing data
tomiss = function(data.miss){
  for(row in 1:dim(data.miss)[1]){
    for(column in 1:(dim(data.miss)[2]-2)){
      if(data.miss[row, column] < 0){
        data.miss[row, column] = NA
      }
    }
  }
  return(data.miss)
}
mep.miss <- tomiss(mep)
vep.miss <- tomiss(vep)
ssep_lower.miss <- tomiss(ssep_lower)
ssep_upper.miss <- tomiss(ssep_upper)

# Let us solve the missing data
# for mep
set.seed(671290)
mep.miss$imputed_ahl <- with(mep.miss, impute(AH_L, 'random'))
set.seed(671290)
mep.miss$imputed_apbl <- with(mep.miss, impute(APB_L, 'random'))
set.seed(671290)
mep.miss$imputed_ahr <- with(mep.miss, impute(AH_R, 'random'))
set.seed(671290)
mep.miss$imputed_apbr <- with(mep.miss, impute(APB_R, 'random'))
mep.imputed <- subset(mep.miss, select = -c(AH_L,APB_L,AH_R,APB_R))
mep.imputed <- rename(mep.imputed, c(imputed_ahl="AH_L",
                                     imputed_apbl="APB_L", imputed_ahr="AH_R",
                                     imputed_apbr="APB_R" ))
mep.imputed = mep.imputed[, c(3,4,5,6,1,2)]
# for other EPs, similar approach is used
## data management
# Reshaping the edss data
edss_1 <- edss
edss_1$date = as.Date(edss_1$date, "%d/%m/%Y")
edss_1 <- edss_1[order(edss_1$patient_id, edss_1$date), ]
edss_1$v1 = 1
edss_1$times <- ave(edss_1$v1, edss_1$patient_id, FUN = seq_along)
edss_1 <- edss_1[-4]
edss.reshape <- reshape(edss_1,
                        timevar = "times",
                        idvar = c("patient_id"),
                        direction = "wide")

# counting unique individuals in all datasets
data.unique = function(given){
  given_unique = given
  given_unique$v1 = 1
  given_unique$times <- ave(given_unique$v1, given_unique$patient_id,
                            FUN = seq_along)
  given_unique = given_unique[given_unique$times ==1,]
  dim(given_unique)
}
data.unique(mep)
data.unique(vep)
data.unique(ssep_upper)
data.unique(ssep_lower)
data.unique(edss)

## getting baseline EDSS
mydata_all = function(mep){
  mep$date = as.Date(mep$date, "%d/%m/%Y")
  meedss <- merge(mep,edss.reshape, by= "patient_id")
  mepdim = dim(mep)[2]
  mep_base = meedss[, c(1:mepdim)]
  moreclms <-seq(from=(mepdim+2),to=dim(meedss)[2],by =2)
  for(rows in 1:dim(meedss)[1]){
    kcol = c()
    for (columns in moreclms){
      kcol[columns] <- abs(meedss[rows,dim(mep)[2]]- meedss[rows,
                                                        columns])
    }
    kcol_col = order(kcol)
    kcol_col.1 = kcol_col[1]
    if(kcol[kcol_col.1] <= 365){
      mep_base[rows, mepdim] <- meedss[rows, kcol_col.1]
      mep_base[rows, mepdim+1] <- meedss[rows, (kcol_col.1-1)]
    }
  }
  mep_base = mep_base[order(mep_base$patient_id, mep_base$date),]
  mep_base.names <- colnames(mep_base)
  mep_base.names[dim(mep_base)[2]] <- "edss"
  colnames(mep_base) <- mep_base.names
  mep_full <- na.omit(mep_base)
  mep_part <- mep_base[!complete.cases(mep_base),]
  out = list(mep_part, mep_full)
  return(out)
}
mep_incomplete = mydata_all(mep.imputed)[[1]]
mep_complete = mydata_all(mep.imputed)[[2]]
vep_incomplete = mydata_all(vep.miss)[[1]]
vep_complete = mydata_all(vep.miss)[[2]]
ssep_upper_incomplete = mydata_all(ssep_upper.miss)[[1]]
ssep_upper_complete = mydata_all(ssep_upper.miss)[[2]]
ssep_lower_incomplete = mydata_all(ssep_lower.miss)[[1]]
```

```
ssep_lower_complete = mydata_all(ssep_lower.miss)[[2]]
# percentage without baseline edss
dim(mydata_all(mep.imputed)[[1]])[1]/dim(mep)[1]
dim(mydata_all(vep.miss)[[1]])[1]/dim(vep)[1]
dim(mydata_all(ssep_upper.miss)[[1]])[1]/dim(ssep_upper)[1]
dim(mydata_all(ssep_lower.miss)[[1]])[1]/dim(ssep_lower)[1]

## putting the x and y variables together
final.clean = function(mep_complete){
  mep_1 <- merge(mep_complete,edss.reshape, by="patient_id")
  mep_2 = mep_1
  all.dim = (dim(mep_complete)[2])+1
  all.dim_date = (dim(mep_complete)[2])-1
  for(rows in 1:dim(mep_1)[1]){
    for (columns in all.dim:dim(mep_1)[2]){
      if (!is.na(mep_1[rows,columns])){
        if (mep_1[rows,columns] == mep_1[rows, all.dim_date]){
          moreclms <-seq(from=columns,to=dim(mep_1)[2],by =2)
          kcol = c()
          if(length(moreclms) > 1){
            for(nu in 2:length(moreclms)){
              if(!is.na(mep_1[rows, moreclms[nu]])){
                kcol[nu] <- abs(abs(mep_1[rows, moreclms[nu]] -
                                      mep_1[rows,columns])-730)
              }
            }
          }
        }
      }
    }
    if(length(kcol) >1){
      kcol_col = order(kcol)
      kcol_col.1 = kcol_col[1]
      close_column = (moreclms[kcol_col.1])-1
      if(kcol[kcol_col.1] <= 183){
        mep_2[rows, dim(mep_1)[2]+1] <- mep_1[rows, close_column]
      }
    }
  }
  all.names = colnames(mep_2)
  all.names[dim(mep_2)[2]] <- "edssT"
  colnames(mep_2) <- all.names
  mep_3 = mep_2[,c(1:dim(mep_complete)[2], dim(mep_1)[2] +1)]
  mep_3 = na.omit(mep_3)
  mep_3$edss.diff <- mep_3$edssT-mep_3$edss
  mep_3$progress[mep_3$edss.diff >=1 & mep_3$edss <= 5.5] <- 1
  mep_3$progress[mep_3$edss.diff >=0.5 & mep_3$edss > 5.5] <- 1
  mep_3$progress[is.na(mep_3$progress)] <- 0
  mep_3$progress <- factor(mep_3$progress,
                           levels = c(0,1),
                           labels = c("No", "Yes"))
  return(mep_3)}
mep.final <- final.clean(mep_complete)
vep.final <- final.clean(vep_complete)
ssep_upper.final <- final.clean(ssep_upper_complete)
ssep_lower.final <- final.clean(ssep_lower_complete)
table1 <- table(mep.final$progress)
table2 <- table(vep.final$progress)
table3 <- table(ssep_upper.final$progress)
table4 <- table(ssep_lower.final$progress)
# final with x and y
dim(mep.final)
dim(vep.final)
dim(ssep_upper.final)
dim(ssep_lower.final)
# proportion given with follow up after 2 year
dim(mep.final)[1]/dim(mep)[1]
dim(vep.final)[1]/dim(vep)[1]
dim(ssep_upper.final)[1]/dim(ssep_upper)[1]
dim(ssep_lower.final)[1]/dim(ssep_lower)[1]

## unsupervised PCA
par(mfrow = c(1,1))
mep.final_pca <- mep.final[, c(2:5,7)]
vep.final_pca <- vep.final[, c(2:13,15)]
SSEP_Upper.final_pca <- ssep_upper.final[, c(2:9,11)]
unsupervised = function(clean_data){
  pc.cr <- princomp(clean_data,scores=TRUE,cor = TRUE, scale=T)
  loadings(pc.cr)
  summary(pc.cr)
  pc_summary = list(loadings(pc.cr),summary(pc.cr))
  biplot(pc.cr)
  Ux1<-as.vector(pc.cr$scores[,1])
  Ux2<-as.vector(pc.cr$scores[,2])
  plot(Ux1,Ux2, col =4, ylab = "Comp2", xlab = "Comp1")
  return(pc_summary)
}
meppca = unsupervised(mep.final_pca)
veppca = unsupervised(vep.final_pca)
sseppca = unsupervised(SSEP_Upper.final_pca)

## unsupervised clustering
mep.final_clus <- mep.final[, c(2:5,7,10)]
vep.final_clus <- vep.final[, c(2:13,15,18)]
SSEP_Upper.final_clus <- ssep_upper.final[, c(2:9,11,14)]
unsupervised_clus = function(clean_data){
  prog = dim(clean_data)[2]
  sd.data = clean_data[, -c(prog)]
  ms.labs = clean_data[, prog]
  sd.data=scale(sd.data)
  sd.data.t = scale(t(sd.data))
  sd.data.obs = scale(sd.data)
  par(mfrow =c(1,2))
  data.dist=dist(sd.data)
  data.dist.t=dist(sd.data.t)
  data.dist.obs=dist(sd.data.obs)
  plot(hclust(data.dist.t),  xlab ="", sub ="",
       ylab ="")
  plot(hclust(data.dist.obs),  xlab ="", sub ="",
       ylab ="")
}
unsupervised_clus(mep.final_clus)
unsupervised_clus(vep.final_clus)
unsupervised_clus(SSEP_Upper.final_clus)

## model selection
model.select = function(project.data){
  mod.unwant <- dim(project.data)[2]
  mep.final.sel <- project.data[,-c(1,mod.unwant-4,mod.unwant-2,
                                  mod.unwant-1, mod.unwant )]
  mod.want <- dim(mep.final.sel)[2]-1
  classlabel = as.numeric(project.data[, mod.unwant]) -1
  mep.final.sel = scale(mep.final.sel)
  set.seed(123456)
  lasso1.bin <- glmnet(mep.final.sel,classlabel,alpha=1, family
                     ="binomial")
  set.seed(1234346)
  CVlasso1.bin <-cv.glmnet(mep.final.sel,classlabel,alpha=1,
                         family ="binomial",nfolds =10)
  ### all coefficients for optimal lambda
  tmp_coeffs.bin <- coef(CVlasso1.bin, s = "lambda.min")
  coeff.bin <- as.data.frame(as.matrix(tmp_coeffs.bin))
  coeff.bin1<-coeff.bin[,1][-c(1)]
  ### only non-zero coefficients for optimal lambda
  n_coeff.bin <- data.frame(name = tmp_coeffs.bin@Dimnames[[1]]
                          [tmp_coeffs.bin@i + 1], coefficient =
                            tmp_coeffs.bin@x)
  # The lasso model prediction
  ## predicting the class labels, but with the same data used
  #to build the model
  ## need to separate train and test data
  class.predict <- predict(CVlasso1.bin, mep.final.sel,
                         s="lambda.min",type = "class")
  ## confusion matrix
  conf.mat <- table(classlabel,class.predict)
  mis.error <- mean(class.predict!=classlabel)
  model.select.out = list(conf.mat, mis.error, n_coeff.bin)
  return(model.select.out)
}
mep.select.model.l <- model.select(mep.final.l)
mep.select.model.r <- model.select(mep.final.r)
mep.select.model.all <- model.select(mep.final)
vep.select.model.l <- model.select(vep.final.l)
vep.select.model.r <- model.select(vep.final.r)
vep.select.model.all <- model.select(vep.final)
ssep_upper.select.model.l <- model.select(ssep_upper.final.l)
ssep_upper.select.model.r <- model.select(ssep_upper.final.r)
ssep_upper.select.model.all <- model.select(ssep_upper.final)

## MEP models
all.row <- dim(mep.final)[1]
mod.unwant <- dim(mep.final)[2]
# lets scale numerical variables
mod.dim <- ((dim(mep.final)[2])-3)
for(scale.variable in 2:mod.dim){
  if(scale.variable!=(mod.dim-1)){
    mep.final[, scale.variable] <- scale(as.numeric(mep.final
                                                    [, scale.variable]))
  }
}
max.len = round(.70*all.row)
predict.sets = 100
mep.final.sel <- mep.final[,-c(mod.unwant-4,mod.unwant-2,
                               mod.unwant-1, mod.unwant )]
mep.final.sel$v1 = 1
mep.final.sel$times <- ave(mep.final.sel$v1, mep.final.sel
                           $patient_id, FUN = seq_along)
mep.final.sel = mep.final.sel[, -c(7)]
progress = mep.final[, mod.unwant]
mep.final.use = data.frame(cbind(progress,mep.final.sel))
final.score = matrix(0,100,10)
final.score.all = matrix(0,100,10)
final.score.rf = matrix(0,100,10)
final.score.rf.all = matrix(0,100,10)
final.score.bos = matrix(0,100,10)
final.score.bos.all = matrix(0,100,10)
final.score.neural = matrix(0,100,10)
final.score.neural.all = matrix(0,100,10)
for(preds in 1:predict.sets){
  train_obs = sample(all.row, max.len)
  train = mep.final.use[train_obs, ]
  test = mep.final.use[-train_obs, ]
```

```
# logistic regression
glm.fit.l=glm(progress~AH_L+APB_L+edss, data=train ,
              family =binomial)
glm.fit.r=glm(progress~AH_R+APB_R+edss, data=train ,
              family =binomial)
glm.fit.all=glm(progress~AH_R+APB_R+edss, data=train ,
                family =binomial)
# Random forest
rf.mep.l =randomForest(progress~.-AH_R -APB_R -patient_id
                       -times, data=train,importance =TRUE)
rf.mep.r =randomForest(progress~.-AH_L -APB_L -patient_id
                       -times, data=train,importance =TRUE)
rf.mep.all =randomForest(progress~ AH_R+APB_R+edss,
                         data=train,importance =TRUE)
# boosting
progress.train = ifelse(train$progress == "No", 0, 1)
boost.boston.l =gbm(progress.train~.-progress -AH_R -APB_R
                    -patient_id -times,data=train, distribution=
                      "bernoulli",n.trees =5000,
                    interaction.depth =4, shrinkage =0.001,
                    verbose =F)
boost.boston.r =gbm(progress.train~.-progress -AH_L
                    -APB_L -patient_id -times, data=train, distribution=
                      "bernoulli",n.trees =5000 ,
                    interaction.depth =4, shrinkage =0.001,
                    verbose =F)
boost.boston.all =gbm(progress.train~.-progress -AH_L
                      -APB_L -patient_id -times, data=train, distribution=
                        "bernoulli",n.trees =5000 ,
                      interaction.depth =4, shrinkage =0.001,
                      verbose =F)
# neural network model
train2 = cbind(progress.train, train)
train2 = train2[, -c(2,3,9)]
test2 = test
test2$progress.train = ifelse(test$progress == "No", 0, 1)
test2 = test2[, -c(1,2,8)]
neural.l=neuralnet(progress.train~AH_L+APB_L+edss, data=train2,
                   hidden=3, act.fct = "logistic",
                   linear.output = FALSE)
neural.r=neuralnet(progress.train~AH_R+APB_R+edss, data=train2 ,
                   hidden=3,act.fct = "logistic",
                   linear.output = FALSE)
neural.all=neuralnet(progress.train~AH_R+APB_R+edss, data=train2
                     ,hidden=3, act.fct = "logistic",
                     linear.output = FALSE)

test.len = dim(test)[1]
```

```
for(tests.pred in 1:10){
  # logistic
  test_set = sample(test.len, test.len, replace = T)
  glm.probs.l = predict(glm.fit.l,test[test_set,],type = "response")
  glm.probs.r = predict(glm.fit.r,test[test_set,],type = "response")
  glm.probs.lr <- (as.numeric(glm.probs.r)+as.numeric(glm.probs.l))/2
  glm.probs.all = predict(glm.fit.all,test[test_set,],
                          type = "response")
  roc_obj<- auc(roc(glm.probs.lr, test[test_set,]$progress))
  roc_obj.all<- auc(roc(glm.probs.all, test[test_set,]$progress))
  final.score[preds,tests.pred] = roc_obj
  final.score.all[preds,tests.pred] = roc_obj.all
  # random forest
  yhat.rf.l = predict (rf.mep.l ,newdata =test[test_set,],
                       type = "prob")[,2]
  yhat.rf.r = predict (rf.mep.r ,newdata =test[test_set,],
                       type = "prob")[,2]
  yhat.rf.all = predict (rf.mep.all ,newdata =test[test_set,],
                         type = "prob")[,2]
  yhat.rf.lr <- (yhat.rf.l+yhat.rf.r)/2
  roc_obj.rf<- auc(roc(yhat.rf.lr,test[test_set,]$progress))
  roc_obj.rf.all<- auc(roc(yhat.rf.all,test[test_set,]$progress))
  final.score.rf[preds,tests.pred] = roc_obj.rf
  final.score.rf.all[preds,tests.pred] = roc_obj.rf.all
  # boosting
  yhat.boost.l=predict(boost.boston.l ,newdata = test[test_set,],
                       n.trees =5000, type = "response")
  yhat.boost.r=predict(boost.boston.r ,newdata = test[test_set,],
                       n.trees =5000, type = "response")
  yhat.boost.r.all=predict(boost.boston.all ,newdata = test[test_set,],
                           n.trees =5000, type = "response")
  yhat.bos.lr <- (as.numeric(yhat.boost.l)+as.numeric(yhat.boost.r))/2
  roc_obj.bos<- auc(roc(yhat.bos.lr,test[test_set,]$progress))
  roc_obj.bos.all<- auc(roc(yhat.boost.r.all,test[test_set,]$progress))
  final.score.bos[preds,tests.pred] = roc_obj.bos
  final.score.bos.all[preds,tests.pred] = roc_obj.bos.all
  # neural
  yhat.neural.l = compute(neural.l,test[test_set,])
  yhat.neural.r = compute(neural.r,test[test_set,])
  yhat.neural.all = compute(neural.all,test[test_set,])
  yhat.neural.lr <- (as.numeric(yhat.neural.l$net.result)+
                       as.numeric(yhat.neural.r$net.result))/2
  roc_obj.neural<- auc(roc(yhat.neural.lr,test[test_set,]$progress))
  roc_obj.neural.all<- auc(roc(yhat.neural.all$net.result,
                               test[test_set,]$progress))
  final.score.neural[preds,tests.pred] = roc_obj.neural
  final.score.neural.all[preds,tests.pred] = roc_obj.neural.all
}
```

```
}
all.score.mp <- c()
all.score.rf.mp <- c()
all.score.bos.mp <- c()
all.score.all.mp <- c()
all.score.rf.all.mp <- c()
all.score.bos.all.mp <- c()
all.score.neural.mp <- c()
all.score.neural.all.mp <-c()
for(nfrows in 1:dim(final.score)[1]){
  for(ncrows in 1:dim(final.score)[2]){
    all.score.mp <- append(all.score.mp, final.score
                          [nfrows,ncrows])
    all.score.rf.mp <- append(all.score.rf.mp,
                                final.score.rf[nfrows,ncrows])
    all.score.bos.mp <- append(all.score.bos.mp,
                                final.score.bos[nfrows,ncrows])
    all.score.neural.mp <- append(all.score.neural.mp,
                                   final.score.neural[nfrows,ncrows])
    all.score.all.mp <- append(all.score.all.mp,
                                final.score.all[nfrows,ncrows])
    all.score.rf.all.mp <- append(all.score.rf.all.mp,
                                   final.score.rf.all[nfrows,ncrows])
    all.score.bos.all.mp <- append(all.score.bos.all.mp,
                                    final.score.bos.all[nfrows,ncrows])
    all.score.neural.all.mp <- append(all.score.neural.mp,
                                       final.score.neural.all[nfrows,ncrows])
  }
}
final.perfomance.lr<- c(mean(all.score.mp), sd(all.score.mp))
final.perfomance.lr.all<- c(mean(all.score.all.mp), sd(all.score.all.mp))
final.perfomance.rf<- c(mean(all.score.rf.mp), sd(all.score.rf.mp))
final.perfomance.rf.all<- c(mean(all.score.rf.all.mp),
                            sd(all.score.rf.all.mp))
final.perfomance.bos<- c(mean(all.score.bos.mp), sd(all.score.bos.mp))
final.perfomance.bos.all<- c(mean(all.score.bos.all.mp),
                             sd(all.score.bos.all.mp))
final.perfomance.neural<- c(mean(all.score.neural.mp),
                            sd(all.score.neural.mp))
final.perfomance.neural.all<- c(mean(all.score.neural.all.mp),
                                sd(all.score.neural.all.mp))

## MEP longitudinal model
mod.unwant <- dim(mep.final)[2]
# lets scale numerical variables
mod.dim <- ((dim(mep.final)[2])-3)
for(scale.variable in 2:mod.dim){
  if(scale.variable!=(mod.dim-1)){
    mep.final[, scale.variable] <- scale(as.numeric(mep.final
                                                   [, scale.variable]))
  }
}
mep.final.sel <- mep.final[,-c(mod.unwant-4,mod.unwant-2,
                               mod.unwant-1, mod.unwant )]
mep.final.sel$v1 = 1
mep.final.sel$times <- ave(mep.final.sel$v1, mep.final.sel$
                             patient_id, FUN = seq_along)
# for distinct to sample from
mep.final.distinct <- mep.final.sel[mep.final.sel$times == 1,]
mep.final.sel = mep.final.sel[, -c(7)]
mep.final.distinct = mep.final.distinct[, c(1,7)]
all.row <- dim(mep.final.distinct)[1]
# we now sample from the distinct set
max.len = round(.80*all.row)
predict.sets = 100
progress = mep.final[, mod.unwant]
mep.final.use = data.frame(cbind(progress,mep.final.sel))
final.score.long = matrix(0,100,10)
final.score.long.all = matrix(0,100,10)
for(preds in 1:predict.sets){
  train_obs = sample(all.row, max.len)
  train_distinct = mep.final.distinct[train_obs,]
  test_distinct = mep.final.distinct[-train_obs,]
  train = merge(mep.final.use, train_distinct, by.x= "patient_id",
                by.y = "patient_id")
  test = merge(mep.final.use,test_distinct, by.x= "patient_id",
               by.y = "patient_id")
  progress.train = ifelse(train$progress == "No", 0, 1)
  train2 = cbind(progress.train, train)
  # longitudinal model
  reem.l=REEMtree(progress.train~AH_L+APB_L+edss+times,
                  data=train2, random=~1|patient_id)
  reem.r=REEMtree(progress.train~AH_R+APB_R+edss+times,
                  data=train2 ,random=~1|patient_id)
  reem.all=REEMtree(progress.train~AH_R+APB_R+edss+times,
                    data=train2, random=~1|patient_id)
  test.len = dim(test_distinct)[1]
  test.max =  round(.80*test.len)
  for(tests.pred in 1:10){
    test_set = sample(test.len, test.max)
    test2_obs = test_distinct[test_set,]
    test2_obs$new = 1
    test2_obs = subset(test2_obs, select=c("patient_id", "new"))
    test2 = merge(test,test2_obs, by.x = "patient_id",
                  by.y = "patient_id")
    test2$progress.train = ifelse(test2$progress == "No", 0, 1)
    # longitudinal model
    yhat.long.l = predict(reem.l, test2, id=test2$patient_id,
                          EstimateRandomEffects=TRUE)
    yhat.long.r = predict(reem.r, test2, id=test2$patient_id,
                          EstimateRandomEffects=TRUE)
    yhat.long.all = predict(reem.all, test2, id=test2$patient_id,
                            EstimateRandomEffects=TRUE)
    yhat.long.lr <- (yhat.long.l+yhat.long.r)/2
    roc_obj.long<- auc(roc(yhat.long.lr, test2$progress))
    roc_obj.long.all<- auc(roc(yhat.long.all,test2$progress))
    final.score.long[preds,tests.pred] = roc_obj.long
    final.score.long.all[preds,tests.pred] = roc_obj.long.all
  }
}
all.score.long.mp <- c()
all.score.long.all.mp <-c()
for(nfrows in 1:dim(final.score.long)[1]){
  for(ncrows in 1:dim(final.score.long)[2]){
    all.score.long.mp <- append(all.score.long.mp,
                                final.score.long[nfrows,ncrows])
    all.score.long.all.mp <- append(all.score.long.all.mp,
                                    final.score.long.all[nfrows,ncrows])
  }
}
final.perfomance.long<- c(mean(all.score.long.mp),
                          sd(all.score.long.mp))
final.perfomance.long.all<- c(mean(all.score.long.all.mp),
                              sd(all.score.long.all.mp))
final.perfomance.long
final.perfomance.long.all

### The VEP and SSEP models have been left out,
### but they follow the same procedure

## Model tests
# test for the difference between patial and full models for MP
t.test(all.score.mp,all.score.all.mp)
t.test(all.score.rf.mp,all.score.rf.all.mp)
t.test(all.score.bos.mp,all.score.bos.all.mp)
t.test(all.score.neural.mp,all.score.neural.all.mp)
t.test(all.score.long.mp,all.score.long.all.mp)
# tests across best models
t.test(all.score.long.mp,all.score.rf.mp)

# test for the difference between patial and full models for VP
t.test(all.score.vp,all.score.all.vp)
t.test(all.score.rf.vp,all.score.rf.all.vp)
t.test(all.score.bos.vp,all.score.bos.all.vp)
```

```
t.test(all.score.neural.vp,all.score.neural.all.vp)
t.test(all.score.long.vp,all.score.long.all.vp)
# tests across best  models
t.test(all.score.long.vp,all.score.rf.vp)

# test for the difference between patial and full models for sP
t.test(all.score.sp,all.score.all.sp)
t.test(all.score.rf.sp,all.score.rf.all.sp)
t.test(all.score.bos.sp,all.score.bos.all.sp)
t.test(all.score.neural.sp,all.score.neural.all.sp)
t.test(all.score.long.sp,all.score.long.all.sp)
# tests across best models
t.test(all.score.long.sp,all.score.rf.sp)


## Tests across EPs
# test for the EPs separately
mep_models = cbind(all.score.long.all.mp, all.score.long.mp,
                   all.score.neural.mp, all.score.neural.all.mp[1:1000],
                   all.score.bos.mp, all.score.bos.all.mp, all.score.rf.mp,
                   all.score.rf.all.mp, all.score.mp, all.score.all.mp)

vep_models = cbind(all.score.long.all.vp, all.score.long.vp,
                   all.score.neural.vp, all.score.neural.all.vp[1:1000],
                   all.score.bos.vp, all.score.bos.all.vp, all.score.rf.vp,
                   all.score.rf.all.vp, all.score.vp, all.score.all.vp)

ssep_models = cbind(all.score.long.all.sp, all.score.long.sp,
                    all.score.neural.sp, all.score.neural.all.sp[1:1000],
                    all.score.bos.sp, all.score.bos.all.sp, all.score.rf.sp,
                    all.score.rf.all.sp, all.score.sp, all.score.all.sp)

multipletest = function(fmodels){
  obsr = fmodels[,1]
  model_pvalues = c(1:9)
  for(models in 2:10){
    model_ttest = t.test(obsr,fmodels[,models])
    model_pvalues[(models-1)] <- model_ttest$p.value
  }
  rawp <- model_pvalues
  bonf = mt.rawp2adjp(rawp, proc = c("Bonferroni"))
  holm = mt.rawp2adjp(rawp, proc = c('Holm'))
  bh = mt.rawp2adjp(rawp,proc=c('BH'))
  by = mt.rawp2adjp(rawp, proc=c('BY'))
  allp =cbind(rawp, bonf$adjp[order(bonf$index),2], holm$adjp
              [order(holm$index),2], bh$adjp[order(bh$index),2],
              by$adjp[order(by$index),2])
  mt.plot(allp,plottype="pvsr", proc=c("rawp","Bonferroni","Holm",
                                       "BH","BY"),leg=c(2,0.95),lty=1,
          col=1:5,lwd=2)
  moout = mt.reject(cbind(rawp,bonf$adjp[order(bonf$index),2],
                          holm$adjp[order(holm$index),2],bh$adjp
                          [order(bh$index),2],by$adjp[order(by$index),2]),
                    seq(0,1,0.05))$r
  return(moout)
}
par(mfrow = c(1,3))
#multipletest(mep_models)
#multipletest(vep_models)
#multipletest(ssep_models)

## tests across EPs
LRaggmodel = cbind(all.score.mp, all.score.vp, all.score.sp)
LRfullmodel = cbind(all.score.all.mp, all.score.all.vp, all.score.all.sp)
RFaggmodel = cbind(all.score.rf.mp, all.score.rf.vp, all.score.rf.sp)
RFfullmodel = cbind(all.score.rf.all.mp, all.score.rf.all.vp,
                    all.score.rf.all.sp)
Boostaggmodel = cbind(all.score.bos.mp, all.score.bos.vp, all.score.bos.sp)
Boostfullmodel = cbind(all.score.bos.all.mp, all.score.bos.all.vp,
                       all.score.bos.all.sp)
NNaggmodel = cbind(all.score.neural.mp, all.score.neural.vp,
                   all.score.neural.sp)
NNfullmodel = cbind(all.score.neural.all.mp,
                    all.score.neural.all.vp, all.score.neural.all.sp)
lgaggmodel = cbind(all.score.long.mp, all.score.long.vp, all.score.long.sp)
lgfullmodel = cbind(all.score.long.all.mp,
                    all.score.long.all.vp, all.score.long.all.sp)

multipletestaccross = function(fmodels){
  obsr = fmodels[,1]
  model_pvalues = c(1:2)
  for(models in 2:3){
    model_ttest = t.test(obsr,fmodels[,models])
    model_pvalues[(models-1)] <- model_ttest$p.value
  }
  rawp <- model_pvalues
  bonf = mt.rawp2adjp(rawp, proc = c("Bonferroni"))
  holm = mt.rawp2adjp(rawp, proc = c('Holm'))
  bh = mt.rawp2adjp(rawp,proc=c('BH'))
  allp =cbind(rawp, bonf$adjp[order(bonf$index),2],
            holm$adjp[order(holm$index),2],
            bh$adjp[order(bh$index),2])
  mt.plot(allp,plottype="pvsr", proc=c("rawp","Bonferroni",
                                       "Holm","BH"),leg=c(2,0.95),
          lty=1,col=1:5,lwd=2)
  moout = mt.reject(cbind(rawp,bonf$adjp[order(bonf$index),2],
                        holm$adjp[order(holm$index),2],bh$adjp
                        [order(bh$index),2]),seq(0,1,0.05))$r
  print(allp)
  return(moout)
}
par(mfrow = c(1,3))
multipletestaccross(LRaggmodel)
multipletestaccross(LRfullmodel)
multipletestaccross(RFaggmodel)
multipletestaccross(RFfullmodel)
multipletestaccross(Boostaggmodel)
multipletestaccross(Boostfullmodel)
multipletestaccross(NNaggmodel)
multipletestaccross(NNfullmodel)
multipletestaccross(lgaggmodel)
multipletestaccross(lgfullmodel)
```