Universiteit Leiden

Master Computer Science

Predicting Disease Progression in Huntington's Disease

Name:	Jasper Ouwerkerk
Student ID:	s2494876
Date:	13/07/2021
Specialisation:	Bioinformatics
1st supervisor:	Dr. Eleni Mina
2nd supervisor:	Dr. Katherine Wolstencroft

Thesis Project in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

Abstract

Enroll-HD is a longitudinal study of Huntington Disease (HD) patients. HD is a neurodegenerative disease which is caused by a mutation on the HTT gene, causing an aberrant CAG repeat at the N-terminus. The Enroll-HD dataset is a high quality dataset which has not yet been fully explored. In this study we propose a data-driven approach to give a prognosis on disease progression and to deduct which variables impact disease progression. In this study another model is created to give an advice on driving capability. Multiple machine learning algorithms were trained and tuned, namely random forest, support vector machine, convolutional, recurrent, long short-term memory, gated recurrent, and feed forward neural networks. These algorithms were applied to allow for a longitudinal personalized prediction per patient on driving capability and disease progression. It was found that generally the GRU had the best performance. The driving capability was predicted accurately in 91% of the positive (able to drive) and 85% of the negative (unable to drive) samples. This system allows for a more nuanced advice on driving capability and can help clinicians in advising patients on their driving capability. The disease progression was modelled in terms of total motor score (TMS), total functional capacity (TFC), cognitive measures (SDMT, and SWRT), and a combination of these variables (cUHDRS). These models showed that, aside from the cUHDRS itself, the cUHDRS progression is mainly influenced by the cognitive measures. This showcases that TMS and TFC might be less important in predicting the progression of the cUHDRS even though it is a measure of motor, functional, and cognitive symptom progression. It was also found that different components of the TMS and TFC vary in influence on the model for both the cUHDRS, TMS, and TFC progression. Suggesting that components of the TMS and TFC have a varying impact on the disease progression. The SDMT and SWRT were mainly influenced by the SDMT and SWRT in the last year respectively. In conclusion, accurate predictions could be made on disease progression and driving capability. It was also shown that some variables have a bigger or smaller impact than previously thought.

Acknowledgement

I would like to thank Dr. Eleni Mina for her feedback on the analysis of the dataset and for her guidance during the project. I would also like to thank Dr. Susanne T. de Bot, Dr. Willeke M.C. van Roon, MSc Stephanie Feleus and MSc Kasper F. van der Zwaan from the Leiden University Medical Center for their expertise and input on Huntington Disease and the results. Lastly, I would like to thank my academic supervisor Dr. Katherine Wolstencroft for her feedback, time and for giving me the opportunity to work on this project.

Chapter 1

Introduction

1.1 Huntington's Disease

Huntington's disease is an incurable neurodegenerative disease causing involuntary movements, cognitive impairment, psychiatric and behavioral problems and progressive weight loss.[1] HD is mainly caused by a mutation in the *HTT* gene. The mutation causes an extended CAG repeat at the N-terminus of the *HTT* gene, which results in a long PolyQ (glutamine) tail in the Huntingtin protein.[1,2,3] Wild type HTT has 6-35 CAG repeats and mutant *HTT* (*mHTT*) has more than 35 CAG repeats. Symptom onset for 40 or more repeats is between 30-50 and more than 60 CAG repeats cause juvenile HD, which is a severe form of HD that manifests very early in life.[4]

The number of CAG repeats on the *mHTT* is inversely correlated with the age at onset (AAO) and can therefore give a good prediction of disease onset. The number of CAG repeats accounts for 47% to 72% of the variation in the disease onset in different HD populations and the remaining factors are caused by interactions with variations in other genes and environmental factors.[5]

In addition to the AAO, the CAG repeat size is also a good determinant for age at death, however it was found that the time between AAO and age at death (disease duration), is independent of the CAG repeat size.[6] It was also found that "two-thirds of the rate of functional, motor, and cognitive progression in HD is determined by the same factors that also determine age at onset, with CAG repeat–dependent mechanisms having by far the largest effect" Aziz et al.[5] (2018). The remaining one-third is still unknown. This might indicate that other variables exist, not included in the previous analysis, that affect disease progression/duration and that a closer examination of clinical data might provide further insight into assessing the disease progression and duration in HD. Determining the remaining factors that influence disease progression and thus, being able to model disease progression can be beneficial for clinical trials and for future disease prognosis and diagnosis of individual patients.[5,6]

Currently, nearly all research on disease progression is conducted using traditional statistical models, which can be limiting. Previous research Aziz et al.[5] (2018) on model-

ing disease progression, using linear mixed models, predicted the longitudinal development of symptoms using mostly the age and the number of CAG repeats on the *mHTT* gene.[5]

In another study 39 variables of interest were used to create imaging and clinical markers of premanifest HD progressions, which can be used in clinical trials as outcome measures..[7]

In these papers, models are generally fed a limited number of variables which are of interest to the researcher. This might introduce bias and might cause them to discard relevant information. Also, these models have assumptions about the data, like normality, equal variance, and absence of disproportionately influential observations in the data. Complying with these assumptions generally results in filtering out participants and longitudinal data, which might introduce bias and might limit the model's predictive power, e.g. variables and patients might be filtered out that play a large role in the analysis, but they are simply discarded.[5,6]

To overcome the limitations explained above we propose a machine learning approach. Using machine learning we want to improve the understanding of HD and assist future research and clinical trials. Machine learning algorithms and AI are increasingly used in medicine and healthcare[8,9] and are beginning to achieve or even surpass human performance. For example, previous research has shown that an AI can potentially accelerate rare disease diagnoses, by calculating disease probabilities based on patient symptoms.[10] In another paper it was found by McKinney et al.[11] (2020) that a machine learning algorithm can classify breast cancer, based on images, better than radiologists. This might suggest that machine learning could also be useful for researching HD, however to our knowledge there have only been a few attempts to apply machine learning in HD research, e.g. evaluating the frequency and factors associated with psychosis in HD,[12] predicting the development of suicidal ideas in HD patients[13] and predicting the size of the CAG-expansion based on phenotypical data.[14]

1.2 Goals and Problem Definitions

To further the understanding of HD our main goal is to model disease progression in HD a data-driven approach, to study which variables contribute the most to disease progression.

This can be achieved by applying machine learning on the Enroll-HD dataset, which is a growing worldwide longitudinal study consisting of 21,116 participants. The Enroll-HD dataset has barely been analysed using machine learning and this large dataset is perfect for machine learning models, which require a lot of data.[15] Here five machine/deep learning methods are proposed, namely Support Vector Machine, Random Forest, Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks to model the longitudinal HD data from the Enroll-HD study.

To model disease progression the composite Unified Huntington Disease Rating Scale (cUHDRS) is used, which is an indicator for disease progression.[16] The cUHDRS is a

combined score of Motor Score (TMS), Functional Score (TFC), total correct answers on the Stroop Word Reading Test (SWRT), and total correct answers on the Symbol Digit Modality Test (SDMT). The TMS and TFC score are the sum of other variables that measure a specific motoric/functional component, e.g. tapping your left finger (fingtapl) or how well a patient can do their domestic chores. All the components of the TMS are shown in appendix A and the TFC components consist of: Occupation, Finances, Domestic chores, ADL, and Care level.

The cUHDRS was developed by Schobel et al.[16] (2017) to 'identify an improved measure of clinical progression in early HD' and the cUHDRS is now also used as a primary outcome in clinical trials.[17] In addition to being a good measure of disease progression, it has also been found by Estevez-Fraga et al.[17] (2021), that cUHDRS correlates with the progression of imaging bio-markers, which is an extra validation of its biological relevance in developing clinical trials.[17]

Modelling the progression of cUHDRS can have an impact on quality of life, since the onset or change of symptoms can be foreseen. This can ensure that a patient is treated at the right time and to identify the optimal moment for intervention. In addition, what impacts a patient's disease progression is also deducted, i.e. a personalized impact panel. Here we want to find out whether the variables making up the score are able to predict the cUHDRS (disease progression) or not and whether new knowledge can be identified as in other variables that might contribute more to disease progression that were previously unknown.

In addition to disease progression, the driving capability is modelled to give an advice on driving capability for HD patients. This problem arises from patients in the clinic that asked whether an advice could be given on when their disease becomes a burden to their driving performance. Currently, the driving capability is a binary indicator whether the patient is driving or not, which can be used to make an estimation on their driving capability compared to other patients. This project should illustrate whether machine learning is a feasible method for helping clinicians and HD patients as an advisory system. It should also be a simple case, to develop the methods and show feasibility of the proposed machine learning models for modelling disease progression.

HD progression has typically been model by linear mixed models and not so much by machine learning models. The machine learning models also allow us to analyze the dataset using a data-driven approach. With this approach we can use all variables within the dataset to see how much each variable affects the output of the model. Therefore, we propose using machine learning models to predict driving capability and disease progression, defined by cUHDRS, in HD patients. With this research we want to introduce the HD research field to machine learning approaches and with these models we potentially want to answer the following research questions:

1. Can ML models provide an accurate advice on driving capability to HD patients and clinicians?
2. Can ML models provide a personalized prognosis on HD progression, defined by the cUHDRS, to HD patients and clinicians?

3. Do the variables making up the cUHDRS also affect the progression of cUHDRS the most?
4. Which variables affect the components of the cUHDRS the most?

1.3 Structure

First, the data used in this project is discussed in chapter 2. Secondly, all background information is explained to understand how neural networks and machine learning models work in chapter 3. Thirdly, the proposed models used to answer the research questions are explained in chapter 4. Fourthly, how the data is pre-processed and how each model is trained and evaluated is explained in chapter 5. Next, the results of these models are shown in chapter 6 and lastly the results are discussed, concluded, and a future prospect is given in chapter 7.

Chapter 2

Data

2.1 Enroll-HD

In this project the Enroll-HD dataset is used, which is a growing worldwide longitudinal study of HD patients. One of the many objectives of Enroll-HD is to facilitate disease modeling studies, assisting in the identification of beneficial interventions, and promoting interrogatory studies that may provide clues to the parthenogenesis of HD. The study collects baseline and follow-up data from multiple sites worldwide of control, pre-manifest, and manifest patients.

Enroll-HD is built upon three studies, namely Ad Hoc, REGISTRY, and Enroll. The Ad Hoc study was conducted before REGISTRY, which was active between 2004-2015 and originated from the European Huntington Disease Network (EHDN). The REGISRTY study is included in the Enroll-HD study under protocol 2 and 3. The most recent and biggest study included in Enroll-HD is the Enroll study, which started in 2012 and is still on-going. The Enroll study has research sites in North America, Europe, Australasia, and Latin America. The sizes of these studies are shown in table 2.1. In addition to a large number of participants/visits, the studies also include a lot of variables, including motor, functional, cognitive, and behavioural assessments and information about nutritional and medication supplements. The studies also have some overlap in terms of variables measured and patients participated, see figures 2.1 and 2.2 respectively. As shown in figure 2.1, most variables are shared between registry (65.7%) and all three studies share 30% of the variables. In terms of patients, see figure 2.2, 295 patients are shared between all studies and most new patients are included in the Enroll study (78.7%).

In short, the dataset is a rich source of high quality information about longitudinal data of HD patients, which can be utilized by machine learning algorithms. However, in order to utilize the dataset using machine learning the data has to be pre-processed first.[15,18]

Table 2.1: *The number of participants and visits for each study according to the latest version (PDS5) of the Enroll-HD dataset.*[15]

Study	Participants	Visits	Variables
Enroll	21,116	55,975	304
Registry	6,247	14,737	287
Ad Hoc	302	970	109

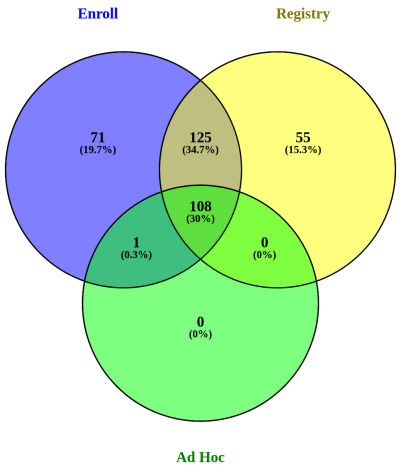


Figure 2.1: *A venn diagram of the columns, i.e. measured variables in each study.*

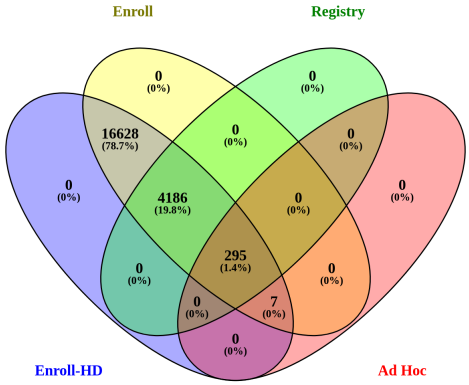


Figure 2.2: *A venn diagram of the patients, i.e. the unique subject ids in each study.*

2.2 Pre-Processing

In this project the Fifth Periodic Dataset (PDS5) of Enroll-HD is used. In this dataset around 47.15% of all values were missing values. This can be attributed to many factors, e.g. errors, inconsistencies, the variable is not applicable for a specific patient and simply missing data. This is a problem since machine learning models generally can not handle missing values. Therefore, the dataset had to be pre-processed before using machine learning algorithms.

In the introductory research project a workflow was developed to pre-process the PDS4 version of the Enroll-HD dataset. Here we re-applied the same workflow for the new PDS5 version. The workflow includes, separating HD and control patients, detecting outliers, feature engineering and imputation using machine learning models (linear regression, random forest, and K-nearest Neighbours). In this workflow only the enroll dataset is used, since enroll is the largest dataset and describes all patients found in Enroll-HD,

see figure 2.2. Also, combining the other studies by including all variables would result in many missing values or the reduction of variables, by only keeping variables shared between the studies, see figure 2.1.

To summarize a flowchart of the workflow of the introductory research project is shown in figure 2.3. From this workflow two datasets arise, namely the pre-imputed dataset and the imputed dataset. Here the pre-imputed dataset is the dataset before imputing the dataset using the machine learning models.

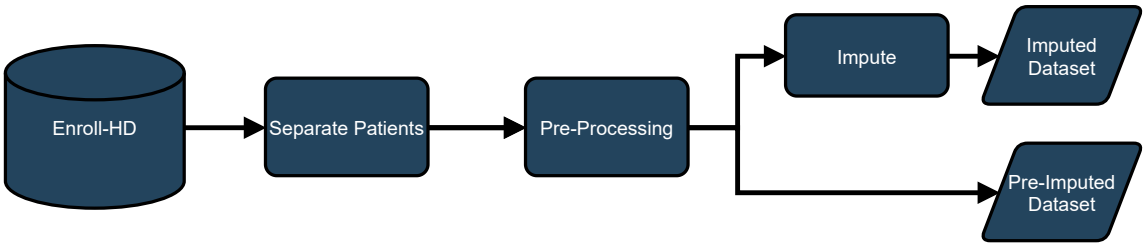


Figure 2.3: *Flowchart of the workflow created in the introductory research project.*

2.3 Dataset Statistics

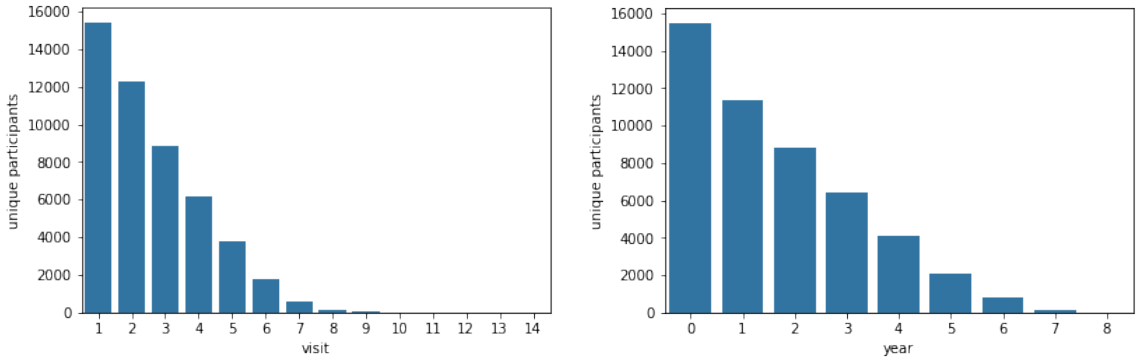
Some basic statistics of these datasets are shown in table 2.2. This table shows that the imputed dataset has no missing values and the pre-imputed dataset only has 6.84% missing values. The distribution of all the variables in the imputed dataset are included in the appendices B, C, and D.

Table 2.2: *Basic statistics of the pre-imputed and imputed dataset, including the number of participants, visits, and variables.*

Dataset	Variables	Participants	Visits	Visits/Participant	Missing (%)
Pre-Imputed	498	15427	49082	3.2 (+/-1.8)	6.84
Imputed	498	15427	49082	3.2 (+/-1.8)	0.0

Table, 2.2 also shows that the participants have an average of 3.2 visits (sd: 1.8). The exact number of participants per visit can be seen in figure 2.4a. This figure shows that there are participants that have a total of 14 visits, however figure 2.4b shows that the longest time between the first and last visit is 8 years. Even though each visit should be 1 year apart, the figures 2.4a and 2.4 indicate that this is not the case.

The exact distribution of the years between the current and next visit is shown in figure 2.5. This figure shows that indeed most visits are 1 year apart, however in some cases the visits can only be 1 month apart or even 5 years apart. However, it is essential that the time between each visit is consistent. How this is achieved is explained in the next section.



(a) *Number of unique participants per visit.* **(b)** *Number of unique participants per year. The year is calculated using equation 5.2.*

Figure 2.4: *The distribution of the number of participants per visit. In figure 2.4a the number of visits is shown and in figure 2.4b the number of yearly visits is shown.*

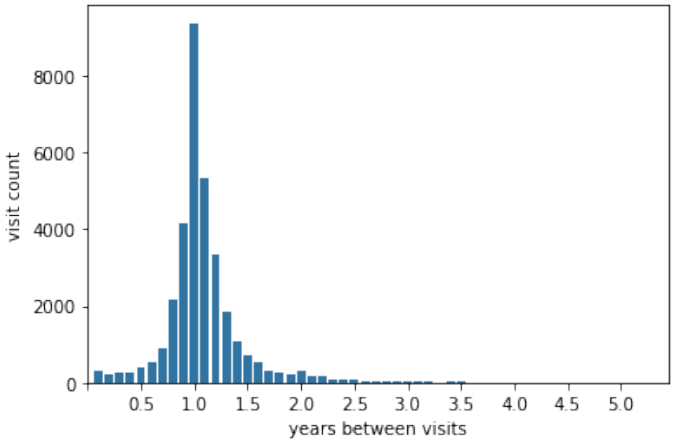


Figure 2.5: *The distribution of years between the current and next visit.*

Chapter 3

Background

3.1 Machine Learning (ML) and Deep Learning (DL)

There are many types of machine and deep learning algorithms that can solve different problems, revolving around pattern recognition. How these algorithms learn these patterns can be very different, however how they are trained is very similar. In this chapter the general strategies of training machine and deep learning algorithms is explained and how the proposed models work themselves is further elaborated in chapter 4.

3.2 Training

In this section, it is explained how machine and deep learning can be applied to solve pattern recognition problems. To make such an algorithm useful it has to be "trained", which refers to optimizing their parameters, e.g. weights and bias values, to fit a preferred output using a specific input. So in order to train such an algorithm input data is required with known outputs, also known as labeled data. In this case Enroll-HD data (input) is used to predict for example the age of onset (output/label). These predicted values from the model are then used to calculate the error/loss between the predicted and real outputs/labels. This error/loss is then used to update the model's parameters to better fit the labels, until the model does not improve anymore. This can be the root mean squared error (RMSE) for regression problems and cross entropy (CE) for classification tasks, the formulas to calculate the error for one predicted label can be seen below.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.1}$$

$$CE_{M=2} = \begin{cases} -log(\hat{p}) & \text{if } y = 1 \\ -log(1-\hat{p}) & \text{if } y = 0 \end{cases} \tag{3.2}$$

$$CE_{M>2} = -\sum_{c=1}^{M} y_{o,c} \times log(p_{o,c}) \tag{3.3}$$

For the SE, y_i is the predicted value and $\hat{y}_i$ is the real value (label), i.e. the squared difference. For the cross entropy loss where the number of classes is 2 ($M = 2$), $\hat{p}$ is the predicted label and y is the true label. This makes sense, since $-log(\hat{P})$ (the loss) increases when it approaches 0 and $-log(1-\hat{p})$ increases when it approaches 1. For the cross entropy where $M>2$, y equals 0 (correct) or 1 (incorrect) depending on whether o is equal to the label c and p is the output value (probability) that instance o equals c. In short, the cross entropy loss is used to measure how well a set of estimated class probabilities matches a target class, which is calculated using the softmax activation function, see equation 4.4.

3.3 Train and Test Sets

In machine learning the dataset is divided into two datasets, namely the train and test set. Here the training set is used to fit the algorithm. After training the model on the training set it is evaluated on the test set. At the evaluation step it is calculated how accurate the model is for predicting the output labels for the test set. This is done using measurements like accuracy, recall, precision, and F1-score for classification problems and the MSE for regression problems. So the backpropagation algorithm is never performed on the test dataset. This is split up since it prevents overfitting and allows to evaluate if the model is generalised. When a model is overfitted it means that the model "learns" patterns in the training data which are specific for that dataset, making the model not ideal for other datasets. The test set shows that a model is overfitting when the performance on the training set keeps increasing and the performance on the test set decreases. When this happens, training should stop, this is known as early stopping. With early stopping a patience is given, which are the number of consecutive iterations of unimproved loss on the test set needed to stop training.[19]

3.3.1 Class Imbalance

When dividing the dataset into train and test sets it is important to have a train and test set that both represent the original dataset. For example the distribution of predictable labels/classes should be close to the original dataset in both the train and test set. This is important, because the model should be able to predict the patterns within the whole dataset. Therefore, the train and test should be as representative to the original dataset as possible, e.g. the distribution of the classes between the original, train, and test

datasets needs to be the same. This is generally achieved by randomly sampling from the dataset to create the train and test dataset.[19]

Another problem to keep in mind is the class imbalance in the dataset. When dealing with a classification problem the number of classes in the dataset might include 90% of label A and only 10% of label B. When training on this dataset the model is biased to learn to predict label A more, since most of the time it would encounter such a sample. Therefore, the loss for predicting all classes are weighted by their presence in the training dataset. A lower presence would mean a higher weight, i.e. a higher loss.[20] This weight (w_c) is calculated using equation 3.4, where n are the number of samples in the dataset, C the total number of unique classes, and n_c the number of samples labelled with class c. This equation was inspired by King and Zeng[21] (2001).

$$w_c = \frac{n}{(C * n_c)} \tag{3.4}$$

3.3.2 Holdout Validation & K-Fold Cross Validation

There are two methods of splitting the train and test set, namely holdout validation and K-fold cross validation. Holdout evaluation is simply the splitting of the dataset into one train subset and one test subset. Common split percentages are 80/20, 67/33 or 50/50, where a split of 80/20 is commonly used when there is not much data to train the model on.

An alternative to the holdout evaluation method is the K-fold cross validation method. With this method the dataset is split into K subsets, where K generally has a value of 10. After splitting the dataset K models are trained on K-1 subsets and the left out subset is used as a test set. This results in K evaluation scores, which can give a good indication of how precise the evaluation score is by looking at the standard deviation of the evaluation score. The only disadvantage to this method is that it requires the training of multiple models, which sometimes might be infeasible since training a model can take a very long time depending on the size of the dataset.

In the introductory research project the K-fold cross validation method was used. In this project the holdout evaluation method is used, since neural networks can have a long training time.[19]

3.4 Regularization and Overfitting

There are several methods that reduce overfitting of the models. These are *l1* regularization, *l2* regularization, Dropout, and early stopping discussed in section 3.3. In this research only the early stopping and *l2* regularization techniques are used. With *l2* regularization a penalty is added to the normal loss function, which is shown in equation 3.5. Here the error is normally calculated using the real labels (y) and the predicted labels ($\hat{y}$) and the penalty is calculated using the size of the regularization factor (λ) times half

the squared sum of the weights ($\frac{1}{2}\sum_{i=1}^{n} w_i^2$). This penalty forces the model to keep the weights as small as possible, which regularizes the model, while still allowing the model to fit to the data depending on the size of the regularization factor (λ).[19]

$$loss = error(y, \hat{y}) + \lambda \frac{1}{2} \sum_{i=1}^{n} w_i^2 \tag{3.5}$$

3.5 Evaluation Methods

In the previous sections some evaluation methods were mentioned, namely accuracy, recall, precision and F1-score. In this section these measurements are explained.

3.5.1 Classification Metrics

Firstly, the accuracy is simply the fraction of correctly predicted labels, true positives (TP) and true negatives (TN), and the total labels, TP + TN + false positives (FP) + false negatives (FN), as shown in equation 3.6. The accuracy is a simple, but misleading metric, since it does not evaluate whether the model is actually learning. For example, a model could achieve a very high accuracy by only predicting one class, where 90% of the observations are labelled with class 1 and the other 10% are labelled with class 2.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.6}$$

A similar metric to the accuracy is the F1-score. The F1-score is the harmonic mean of the precision and recall (a.k.a. sensitivity), see equation 3.7.[19]

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.7}$$

Here the recall is defined as the fraction of "relevant" labels that were actually correct, i.e. how many of the actual positive labels (TP + FN) were actually correctly predicted (TP), which is calculated using equation 3.8.

$$Sensitivity = Recall = \frac{TP}{TP + FN} \tag{3.8}$$

The precision is defined as the fraction of predicted positive labels that were actually correct, i.e. how many of the predicted positive labels (TP + FP) were actually correct (TP), which is calculated using equation 3.9.

$$Precision = \frac{TP}{TP + FP} \tag{3.9}$$

Both recall and precision give a better perspective of what the model is actually learning, namely the recall shows how many positive labels are detected by the model and the

precision shows how accurately it detects positive labels. These measures can therefore show if a model is actually learning, for example when the recall is very high and the precision is very low it will be clear that the model is most likely predicting a single class. However, to show the overall performance of the model the F1-score is used, since it is easier to interpret a single value.

Another well-established classification metric is the area under the receiver operating characteristics also known as AUROC or the area under the curve (AUC) and the ROC-curve. The ROC-curve is a plot showing the recall / true positive rate (TPR) versus the false positive rate (FPR), which is the ratio of actual negative labels predicted as positive labels, at different classification threshold values. An example of a ROC-curve is shown in figure 3.1. Here the blue line represents a trained model and the dashed line represents a random model. The ROC-curve shows that the trained model performs much better than the random model, since it is near the top left corner, i.e. it has a high TPR at a low FPR. The performance of each model is translated into a single metric namely, the AUC which indicates how well the model can distinguish the classes. Aside, from the AUC, the ROC-curve can also indicate which classification threshold is most optimal for the task at hand. This depends on whether TPR should be maximized or FPR should be minimized. For example, when diagnosing patients with a devastating disease it would be favorable to reduce falsely diagnosing patients, i.e. reduce the FPR. When no particular measure should be optimized the optimal threshold can be determined by finding the threshold that maximizes the $TPR - FPR$.

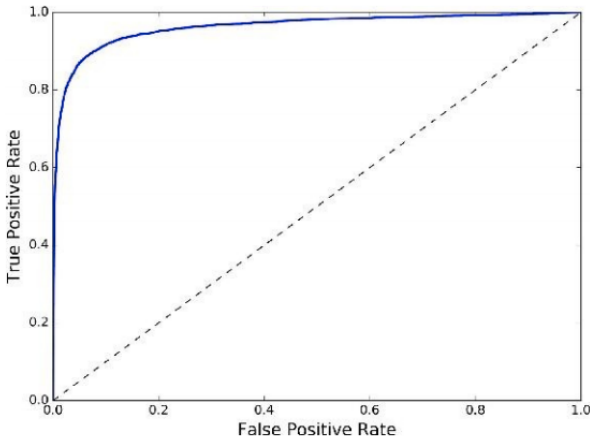


Figure 3.1: *An example of a ROC-curve.*[19]

Another, visual metric is the confusion matrix. A confusion matrix displays the actual number of true positives, false positives, true negatives and false negatives in one figure, see figure 3.2. This visualization directly shows which class is hard to predict.

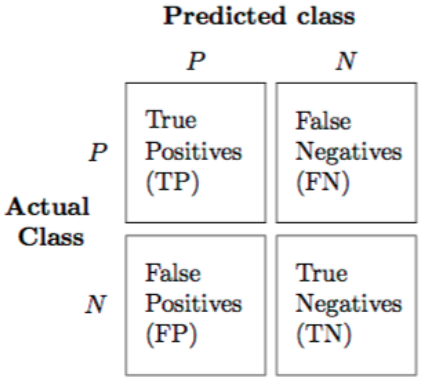


Figure 3.2: *An example of a confusion matrix.*[22]

These performance measures are all related to (ordinal) classification problems, however for regression problems these scores are not applicable. For regression problems other metrics are used, which are explained in the next section.

3.5.2 Regression Metrics

The R^2 compares the trained model to a baseline model which always predicts the mean. The R^2 score ranges from $(-\infty, 1]$. When the R^2 score is 1 it means that the model perfectly fits the data, 0 means that the model is just as good as taking the mean and lower than 0 means the model is worse than just taking the mean.

$$\begin{aligned} SS_{tot} &= \sum_{i=1}^{n}(y_i - \overline{y})^2 \\ SS_{res} &= \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \\ R^2 &= 1 - \frac{SS_{res}}{SS_{tot}} \end{aligned} \tag{3.10}$$

In addition to the R^2 score, the mean absolute error MAE and the RMSE are also good measures for evaluation. The RMSE is also used as a measure of loss which has already been discussed, see equation 3.1. The MAE, see equation 3.11, is a bit more interpretable than the RMSE, since it uses the absolute difference instead of the squared difference between the real value (y) and the predicted value ($\hat{y}$), i.e. the MAE is the average absolute prediction error.

The RMSE is always greater or equal to the MAE and if the RMSE is close to the MAE then the model generally does not overestimate or underestimate the prediction value, which gives a nice performance oversight of the model.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.11}$$

Chapter 4

Proposed Models

4.1 Neural Networks

First the "standard" (feed forward) neural network (NN) is explained. A NN takes variables/features as input data and can output one or multiple float numbers, known as a regression model or the output can be one or multiple classes/labels, which is known as classification model. An example of input and output data for a feed forward neural network for a classification problem can be seen below in table 4.1. As shown in the table multiple input types are also possible for the network to take into account. In table 4.1 the input variable types are: a date, a continuous value that can represent a concentration, a binary categorical value representing the presence of a feature or gender, a categorical value representing one of more than three possibilities, and a discrete value, a value ranging from 0 to infinity, which can represent for example age. However, all these values are transformed into numerical values, since a neural network can only process numbers.

Table 4.1: *An example of input data for a classification problem.*

Var 1	**Var 2**	**Var 3**	**Var 4**	**Var 5**	**Output (class)**
13-10-2020	0.23	True (1)	Low (1)	26	Sick (0)
01-01-2020	5.25	False (0)	High (3)	50	Healthy (1)

The idea/theory for neural networks has been around for quite some time, however they only became very popular in the last decade. This is because these networks require a lot of high quality data and are computationally intensive to use.[23,24,19]

4.1.1 Design

A neural network is simply put a layered network of neurons, where data is processed through the network layer by layer, see figure 4.1. As shown in the figure each neuron from the previous layer is connected to all the neurons in the next layer. Each connection/line has a weight and each neuron has a bias, where the neurons follow a linear function: $y = b + \sum_{i=1}^{n} x_i * w_i$, where n is the number of incoming connections, x_i is the variable from a neuron in the previous layer, w_i is the weight of the connection/line between the previous and next neuron and b is the bias of the output neuron. In this case a neuron in the hidden layer receives four inputs from the previous layer (input layer). A neuron in a neural network is actually exactly the same as a linear regression model, another machine learning technique. Such a model also takes input variables and calculates the weighted sum plus the bias and the result is the output of the model. However, in a neural network such models are connected with one another and all the input values are aggregated into one value in each neuron which is done using so called activation functions. These activation functions transform all the values into one value that falls within a desired range, depending on the activation function used, see figure 4.2. In this figure the activation function simply takes the sum of all output values. These activated values are then again processed to the next layer, the output layer, and these values are also aggregated and activated which gives a particular output. In this case there is only one output neuron, so this network solves a binary classification problem or a regression problem where the expected output is one value.[23,24,19]

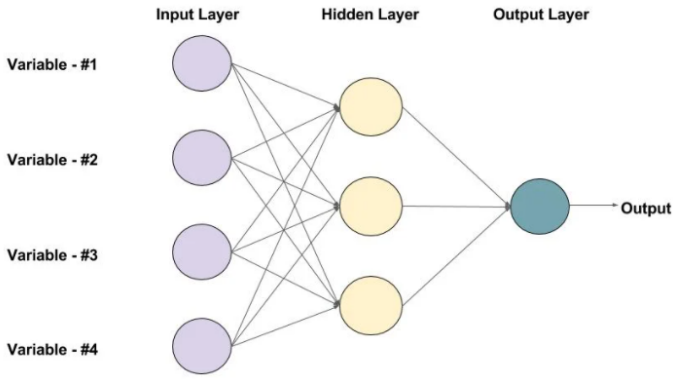


Figure 4.1: *A neural network with one hidden layer of three neurons.*[25]

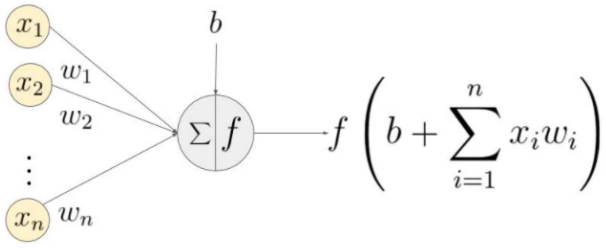


Figure 4.2: *A neuron in a neural network with an activation function that aggregates all the input values into one value.*[25]

4.1.2 Activation Functions

Activation functions used in this study include linear activation, the rectified linear unit (RELU), tanh, sigmoid, and softmax. The linear activation function is simply the weighted sum of the input values shown in figure 4.2 and 4.3a. RELU is an activation function that follows:

$$f(x) = \max(0, x) \tag{4.1}$$

which transforms the output to fit $[0, \infty)$, which is shown in figure 4.3b.[26]
The tanh function

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.2}$$

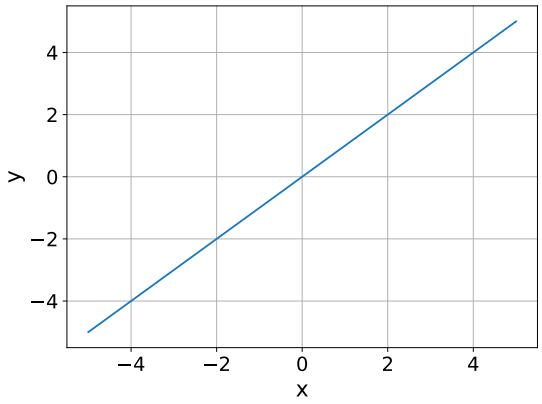
transforms the output to fit $(-1, 1)$, which is shown in figure 4.3c. The tanh function is commonly used in recurrent neural networks (RNNs).[19]
Sigmoid follows the function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.3}$$

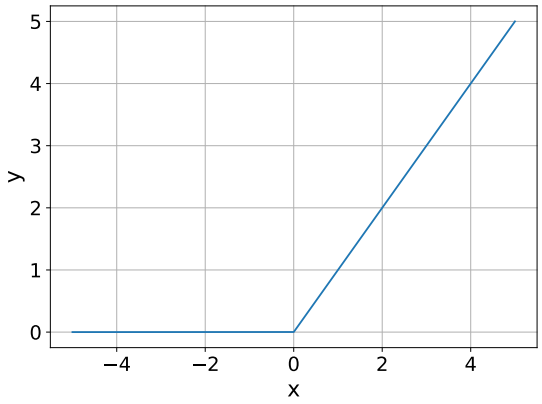
which transforms the output to fit $(0, 1)$, this is shown in figure 4.3d. This is very handy for predicting binary outputs like sick or healthy.[27] When trying to predict from multiple classes softmax is used. Softmax simply computes the exponential of each output neuron (linear output value) and then normalizes them using the sum of all the exponentials, see below:

$$\hat{P}_k = \sigma(s(x))_k = \frac{\exp(s_k(x))}{\sum_{j=1}^{K} \exp(s_j(x))} \tag{4.4}$$

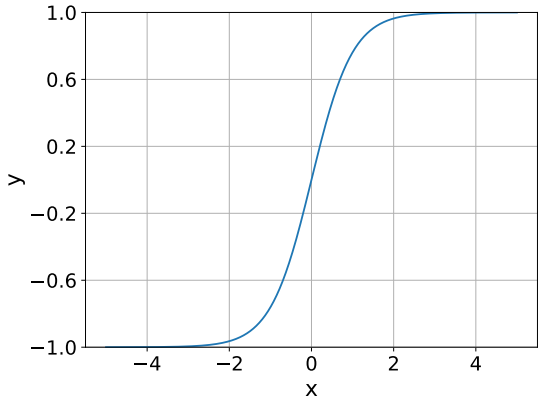
where K the number of classes, $s(x)$ a vector containing the scores (linear output value) of each class/neuron for the instance x (input) and $\sigma(s(x))_k$ is the estimated probability that the instance x belongs to class k, given the scores of each class for that instance. This function calculates the probability for each class (summing up to 1), which could be handy if multiple outputs are possible, however when there is one correct output the highest probability is simply taken using an argmax function. For example for the scores in the set 2, 1, 5, 4, 3 the softmax function transforms the values to sum up to 1, see figure 4.3e. In this case the class corresponding to value 3 will be predicted when the argmax is applied.[19]



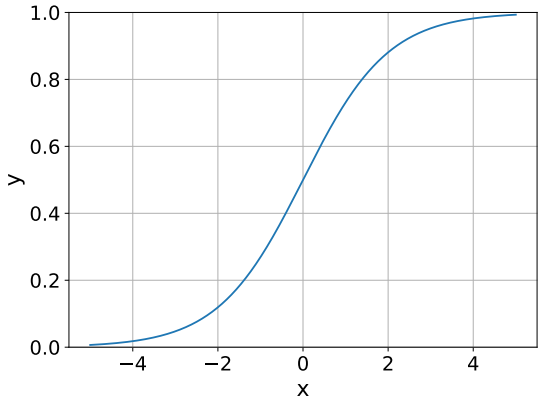
(a) *The linear activation for x in $(-5, 5)$.*



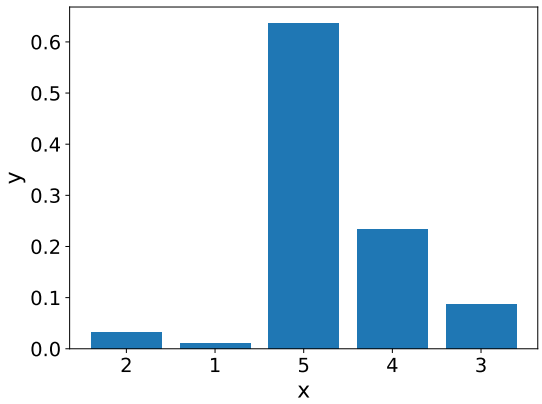
(b) *The RELU activation for x in $(-5, 5)$.*



(c) *The tanh activation for x in $(-5, 5)$.*



(d) *The sigmoid activation for x in $(-5, 5)$.*



(e) *The softmax activation for the x values [2, 1, 5, 4, 3] respectively.*

Figure 4.3: *Five activation functions used in this project.*

4.1.3 Fitting

To fit a model for a specific problem, the neural network's weights and biases are optimized using the backpropagation algorithm. Backpropagation is an algorithm that allows to compute the gradient of the network's error for each parameter (θ_j) (weights and biases). The gradient is a direction pointing towards a value for the current bias or weight that is most optimal from the current point of view. This requires the algorithm to calculate the direction of the slope and how much the loss will change when taking a step in that direction, see figure 4.5. This is done using partial derivatives, noted as $\frac{\partial}{\partial\theta_j}Loss(\theta)$, here the partial derivative is calculated with regards to the loss of all the parameters (θ).

The size of the steps taken is determined by the learning rate (lr). This learning rate is generally around 0.0001, but that depends on the dataset and neural network. When the learning rate is too low it takes too much time to ever find the local minima for the weights. It can also cause the parameter to get stuck in a local minima very quickly when there are many minima's. This is where the gradient is 0, but the value for the weight is not the most optimal to reduce the loss as much as possible, see figure 4.4. When the learning rate is very high, e.g. lr=1 training becomes very unstable, see figure 4.6. This makes it almost impossible to ever find a minimum. Therefore, it is key to find an optimal learning rate.

The previously mentioned backpropagation step can use different optimization algorithms to determine what to do with the gradients to optimize the weights. For this, mini-batch Gradient Descent (GD) is generally used. Both algorithms are illustrated below.

1. Take a mini-batch from the dataset, i.e. rows where the number of rows is known as the batch size. This mini-batch allows the parameters to jump out of local minima, since the landscape of the minima's changes with each new sample.
2. Predict for each instance/row in the mini-batch the given output with the current network. This is known as the forward pass, since the information goes through the network.
3. Measure the loss of the batch accordingly using the formulas described above, see equation 3.1, 3.2 and 3.3.
4. Compute for each connection in the previous layer how much each weight and bias contributes to the error using the chain rule. This is known as the backward pass, since the error is calculated from the output layer to the input layer.
5. Perform GD to tweak the weights and biases using the calculated gradients of the mini-batch.

This algorithm is performed on the whole dataset using the mini-batches, where each mini-batch is sampled from the unsampled dataset. Training one iteration on a mini-batch is called an epoch and training on all the samples is done in an iteration.[19,28]

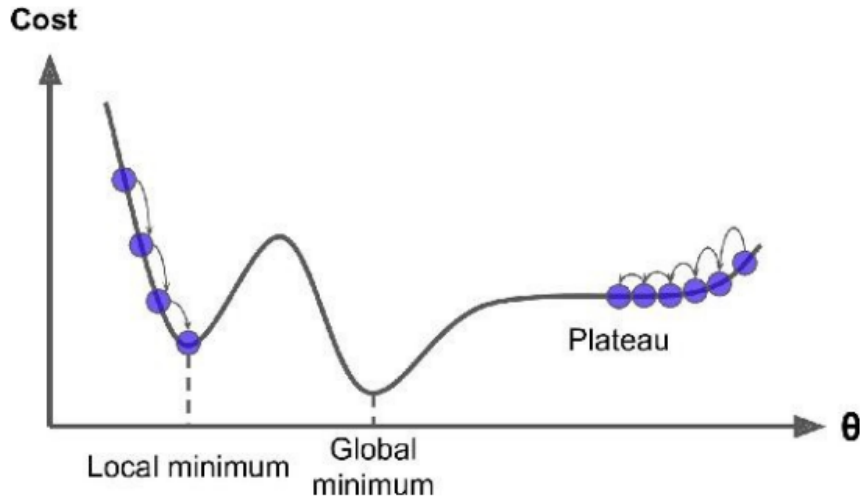


Figure 4.4: *This figure shows the gradient for two different points. Here the y-axis is the loss and the x-axis is the value of the weight.*[19]

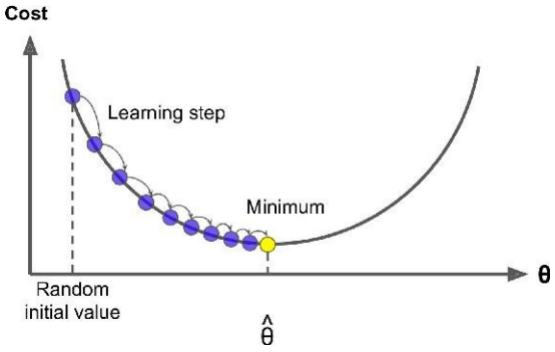


Figure 4.5: *This figure shows the gradient descent algorithm for one parameter (θ). Here the y-axis is the loss and the x-axis is the value of the parameter.*[19]

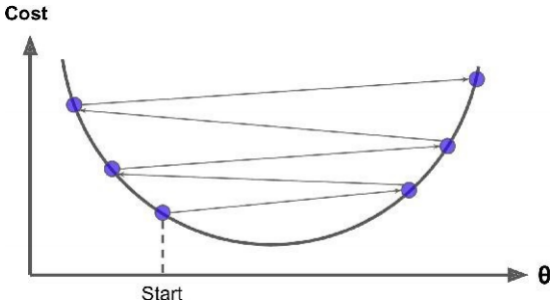


Figure 4.6: *This figure shows the gradient descent algorithm for one parameter (θ) where the learning rate is too big causing big steps which makes it very hard to reach the minima.*[19]

4.2 Recurrent Neural Networks

In this section, the difference between RNNs and the feed forward NN is highlighted. Firstly, a RNN is typically used for data where previous events are relevant for future events, e.g. stock price predictions or disease prognosis, i.e. sequential data. Secondly, the architecture of a RNN is different from a NN, namely a RNN can be seen as a stacked NN. An example of a recurrent neuron is shown in figure 4.7, which shows on the left a recurrent neuron and on the right the unfolded recurrent neuron. The recurrent neuron is actually processing each time step using the input data of the current time step $x_{(t)}$

and the processed output of the previous time step $h_{(t-1)}$ also known as the hidden state. Here the input (x) has the same weight $w_{x(t)}$ for each time step and the hidden state also has a trainable weight w_h. Note that both the input and hidden state have the same weights for each time frame, which allows the network to use a flexible number of time frames. Here the hidden state of the current time frame $h_{(t)}$ is a function of the input of that time frame and the hidden state of the previous time step: $h_{(t)} = f(h_{(t-1)}, x_{(t)})$, here $h_{(t-1)}$ for t=0 is typically set to 0. How $h_{(t-1)}, x_{(t)}$ are used in function f to calculate $h_{(t)}$ depends on the cell type used, which is explained in the next section. Lastly, the inner workings of a recurrent neuron are different from a normal neuron. In this study 3 types of recurrent neurons are discussed, namely the SimpleRNN neuron, the LSTM neuron, and the GRU neuron.[19]

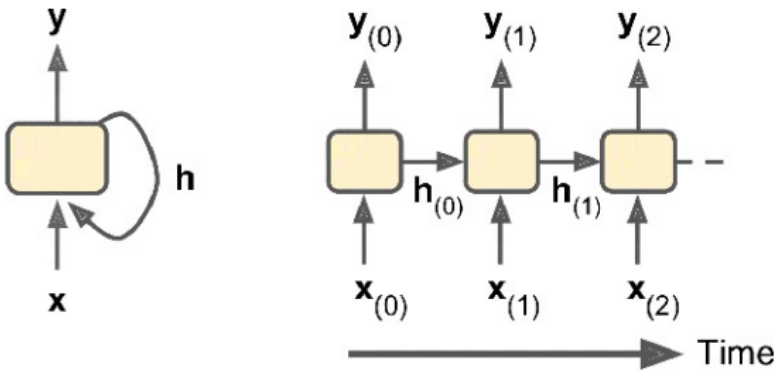


Figure 4.7: *An example of a recurrent neuron on the left, which can use the processed output $h_{(t-1)}$ of the previous time frame $x_{(t-1)}$. Note that a single neuron is displayed in this figure using the same weights for each time step.*[19]

4.2.1 SimpleRNN Cell

In this section the simpleRNN cell is explained. In figure 4.8 an example of a simpleRNN cell is shown. The parameter used in this cell are listed below:

- x_t is a one dimensional input vector of m features.
- h_{t-1} is the hidden state of the previous RNN cell.
- h_t is the hidden state of the current RNN cell, i.e. the output, which is the same as o_t, however o_t can be used as an output for the next layer.
- b_h is the bias vector for the RNN neuron, i.e. the bias for all the cells.
- W_x is the weight vector of the input of the RNN neuron, i.e. the input weight for all the cells.
- W_h is the weight vector of the hidden state of the RNN neuron, i.e. for all the cells.

Now that all parameters are known the hidden state (h_t) is calculated as follows:

$$h_t = tanh(W_h h_{t-1} + W_x x_t + b_h) \tag{4.5}$$

Note that to calculate the current hidden state h_t the previous hidden state $h_t - 1$ and current time step input x_t is used. Note that the next hidden state h_{t+1} uses the current hidden state h_t, therefore the next hidden state h_{t+1} is also depending on the previous hidden state h_{t-1} and the current time frame input x_t, i.e. the cell 'memorizes' the previous time step.[19]

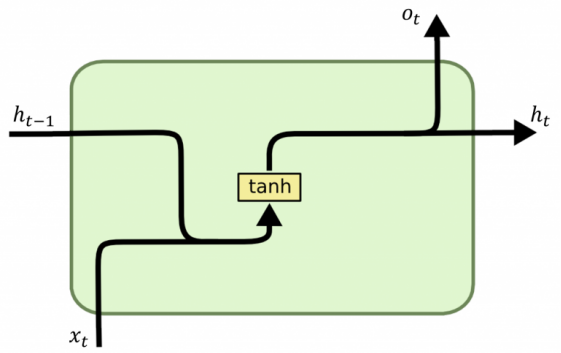


Figure 4.8: *An example of a simpleRNN cell.*[29]

A simpleRNN can learn very short patterns "typically about 10 steps long, but this varies depending on the task" Géron[19] (2019), p500. This is due to the fact that a simpleRNN quickly 'forgets' the important features from the previous time steps. Therefore, the LSTM and GRU might be a better option for longer sequences.

4.2.2 Long Short-Term Memory (LSTM) Cell

The LSTM cell can handle much longer sequences and is also more complex than the simpleRNN cell, since it has much more trainable parameters than a simpleRNN cell and so called gates, see figure 4.9. Also the hidden state is split up into a short-term state h_t and a long-term state c_t. The LSTM cell works better on longer sequences, since it can learn what to store in the long-term state c_t, what to discard and what to read from it. As shown in figure 4.9 the previous long-term state c_{t-1} first discards some memory in the forget gate and then adds some new memories in the addition gate, which were selected using the input gate. After that the long-term memory c_{t-1} goes two ways (1) it is fed to the next cell without any transformation as the new long-term memory c_t and (2) it is transformed using a tanh activation function and together with the processed short-term memory h_{t-1} transformed into the output of the cell y_t and the next short-term state h_t which is propagated to the next cell.[19]

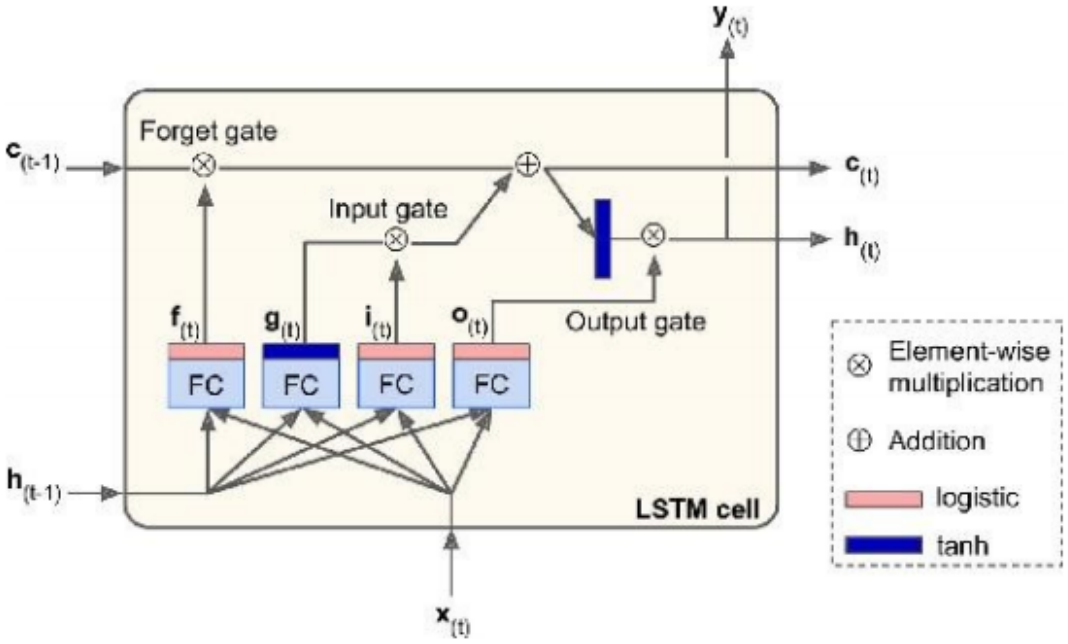


Figure 4.9: *An example of a LSTM cell.*[19]

The forgetting and memorizing of information is simply achieved by using addition and multiplication using the long-term memory c_{t-1}, the short-term memory h_{t-1}, and the input vector x_t, with the functions $f_t, g_t, i_t,$ and o_t. How these functions use the long-term memory c_{t-1}, short-term memory h_{t-1}, and input vector x_t to calculate the new long-term memory c_t and the new short-term memory h_t is listed below. Firstly, function f,

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{4.6}$$

here b_f is the bias of function f, W_{xf} the weight matrix of f for input x_t, and W_{hf} the weight matrix of f for the short-term state h_{t-1}. Again W_{xf} and W_{hf} is the same for the same cell between the time steps. The output of function f is activated using a sigmoid function, denoted by a σ, transforming the output to a range between 0 and 1. This output is then factored with c_{t-1} at the forget gate to 'forget' or 'remember' some long-term memory, i.e. reducing the values in c_{t-1} or increasing the values in c_{t-1} respectively. Next is function g,

$$g_t = tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g) \tag{4.7}$$

here b_g is the bias of function g, W_{xg} the weight matrix of g for input x_t and W_{hg} the weight matrix of g for the short-term state h_{t-1}. Again W_{xg} and W_{hg} is the same for the same cell between the time steps. The output of function g is activated using tanh transforming the output to a range between -1 and 1. This output is later 'forgotten' or 'remembered' by multiplying the outputs of g_t with the output of i_t at the input gate. How i_t is calculated is shown below

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{4.8}$$

here b_i is the bias of function i, W_{xi} the weight matrix of i for input x_t and W_{hi} the weight matrix of i for the short-term state h_{t-1}. Again W_{xi} and W_{hi} is the same for the same cell between the time steps. The output of function i is activated using a sigmoid function (σ) transforming the output to a range between 0 and 1. This output is then factored with the output of g_t at the input gate to 'forget' or 'remember' the output from g_t. Which is then added to $f_t \times c_{t-1}$, which is the final long-term memory output of the cell c_t. Finally, the function o_t is used to create the short-term memory h_t. How o_t is calculated is shown below,

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{4.9}$$

here b_o is the bias of function o, W_{xo} the weight matrix of o for input x_t and W_{ho} the weight matrix of o for the short-term state h_{t-1}. Again W_{xo} and W_{ho} is the same for the same cell between the time steps. The output of function o is activated using a sigmoid function (σ) transforming the output to a range between 0 and 1. The output of o_t is then factored with the tanh of the long-term memory of the current cell c_t, which is calculated as: $tanh(c_t) = tanh((c_{t-1} \times f_t) + (g_t \times i_t))$

In short the current long-term memory c_t is calculated as:

$$c_t = f_t \cdot c_{t-1} + g_t \cdot i_t \tag{4.10}$$

And the current short-term memory h_t is calculated as:

$$h_t = tanh(c_t) \cdot o_t \tag{4.11}$$

Which variables are memorized and forgotten is all determined by the trainable parameters of the functions f, g, i and o, the trainable parameters include the biases of the functions b, the weights of the input vector W_x and the weights of the short-term state W_h.[19]

4.2.3 Gated Recurrent Unit (GRU) Cell

Finally, Gated Recurrent Unit (GRU) cell was proposed by Cho et al.[30] (2014). The GRU is a simplified version of the LSTM cell since it has less trainable parameters, making it faster to train, however it generally achieves the same results. The GRU cell is shown in figure 4.10, which shows that the short-term state (h_{t-1}) and long-term state (c_{t-1}) from the LSTM cell are combined into one hidden state (h_{t-1}) and that there are 3 main functions within the cell, namely r_t, z_t and g_t. These functions achieve the same effect as the LSTM functions. To start with, function r_t,

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{4.12}$$

here b_r is the bias of function r, W_{xr} the weight matrix of r for input x_t, and W_{hr} the weight matrix of r for the hidden state h_{t-1}. Again W_{xr} and W_{hr} is the same for the same cell between the time steps. The output of function r is activated using a sigmoid

function (σ) transforming the output to a range between 0 and 1. This output is then factored with h_{t-1} to determine how much information needs to be forgotten/remembered from previous information, i.e. reducing/increasing the values in h_{t-1}. Next, function z_t is used to determine how much information from previous time steps (h_{t-1}) needs to remain in the future hidden state (h_t). How z_t is calculated is shown below.

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{4.13}$$

here b_z is the bias of function z, W_{xz} the weight matrix of z for input x_t, and W_{hz} the weight matrix of z for the hidden state h_{t-1}. Again the weights in W_{xz} and W_{hz} are the same for the same cell between the time steps. The output of function z is activated using a sigmoid function (σ) transforming the output to a range between 0 and 1. Finally, g_t is calculated to determine how to transform the previous hidden state h_{t-1}, the formula is shown below.

$$g_t = tanh(W_{xg}x_t + W_{hg}(r_t h_{t-1}) + b_g) \tag{4.14}$$

here b_g is the bias of function g, W_{xg} the weight matrix of g for input x_t, and W_{hg} the weight matrix of g for the hidden state h_{t-1}. Again the weights in W_{xg} and W_{gz} are the same for the same cell between the time steps. The output of function g is activated using a tanh function transforming the output to a range between -1 and 1. This output is then used together with z_t to calculate h_t and y_t, which is shown below.[19]

$$y_t = h_t = (1 - z_t)g_t + h_{t-1}z_t \tag{4.15}$$

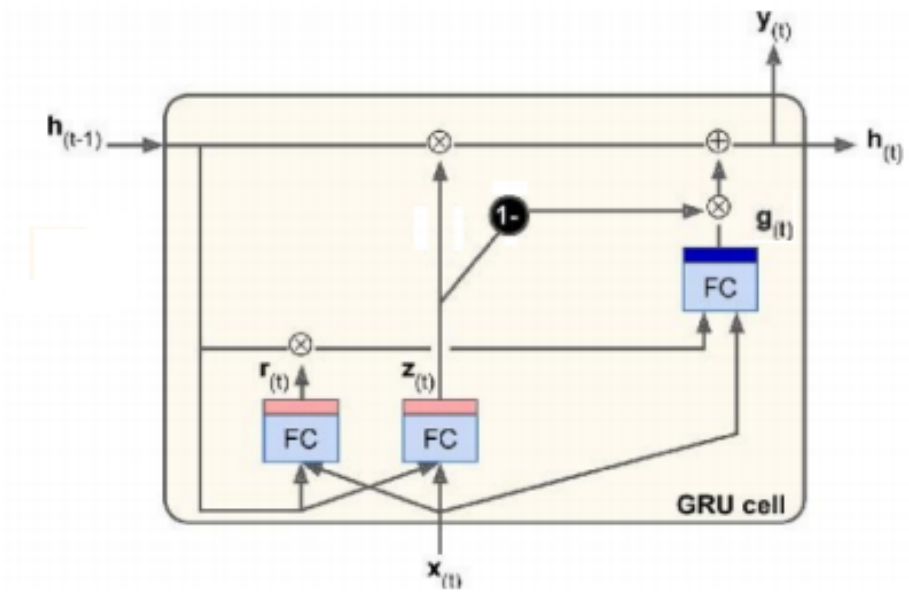


Figure 4.10: *An example of a GRU cell.*[19]

4.3 Convolutional Neural Networks (CNN)

Another type of neural networks are convolutional neural networks (CNNs), which are also almost the same as normal neural networks, however the normal neurons are replaced with convolutional filters. These convolutional filters can work on two dimensional data, like longitudinal data or image data. An example how these convolutional filters work is shown in figure 4.11. This figure shows that filter f moves over the input matrix with a step size (stride) of 1 to create a feature map. The values of the feature map are calculated by multiplying the filter's trainable weights with the input. In this example the filter is two dimensional and has 9 (3×3) trainable weights the image is also zero padded to ensure that the input size equals the output size. In this case only the filter is shown, however many more filters can be trained alongside this filter to increase the number of useful patterns.

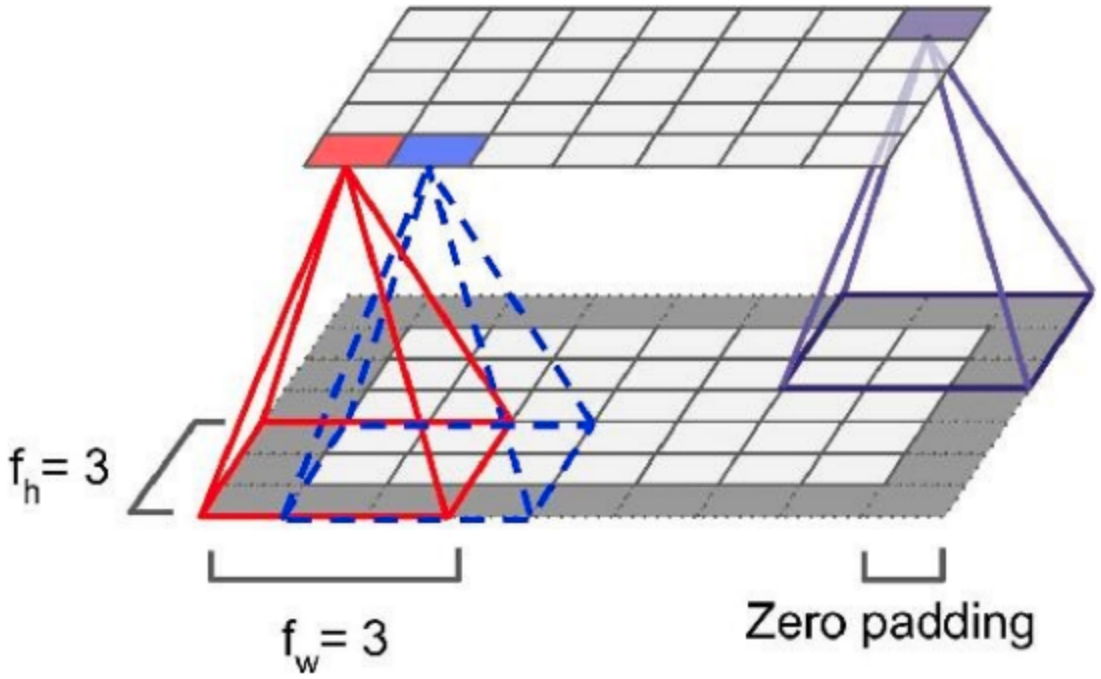


Figure 4.11: *Two examples of an one dimensional convolution.*[19]

The above example shows how a two dimensional convolutional layer works, however for time series a one dimensional convolution can also be used. In this case the width of the filter is always the width of the input vector and only the height can vary. This means that the filters only move over the time sequence in the input matrix.

After a convolution layer (series of convolutional filters), the output of that layer is generally pooled. These pooling layers take a sub-sample of the input matrix to reduce the computational intensity. The pooling layers work similar to the convolution layers, however there are no trainable parameters and the filters use an aggregation function to

sub-sample the input. An example of such a pooling layer is shown in figure 4.12, where the maximum function is used as an aggregation function.[19]

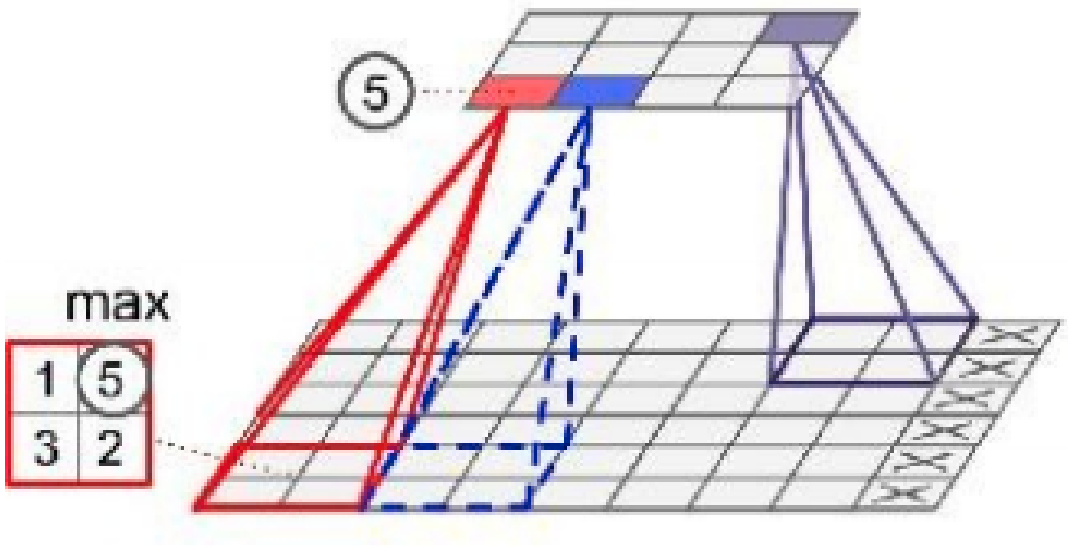


Figure 4.12: *An example of a max pooling layer.*[19]

4.4 Machine Learning Models

4.4.1 Random Forest

Random forest is a collection of decision trees which are trained via the bagging method. A decision tree is made up of nodes, branches and leafs. The nodes represent a question or statement, the branches are the possible answers/directions that connect the nodes in the tree and the leafs represent the predicted outcome, see figure 4.13 for an example of a decision tree. A decision is usually constructed from top to bottom where each condition is chosen based on how well it can separate the data to predict the correct label. [31] The "best" condition can be defined by the Gini Impurity measure (GI), where a low GI indicates a good condition. The GI measure calculates the probability that a random instance is incorrectly classified in the subsets created by the condition. This is calculated in equation 4.16, where C is the number of classes and p(i) is the chance of picking a data point with class i. For example, assume that the first condition, caghigh >50 in figure 4.13, splits 90% of class 3 in finances and the other 10% is wrongly classified as another class. This would result in a GI measure of 0.9 * (1-0.9) + 0.1 * (1-0.1) = 0.18 for the finances=3 leaf. The GI of the decision tree is the loss/error which the decision

tree tries to minimize, in order to make the best predictions.[32]

$$G = \sum_{i=1}^{C} p(i) * (1 - p(i)) \tag{4.16}$$

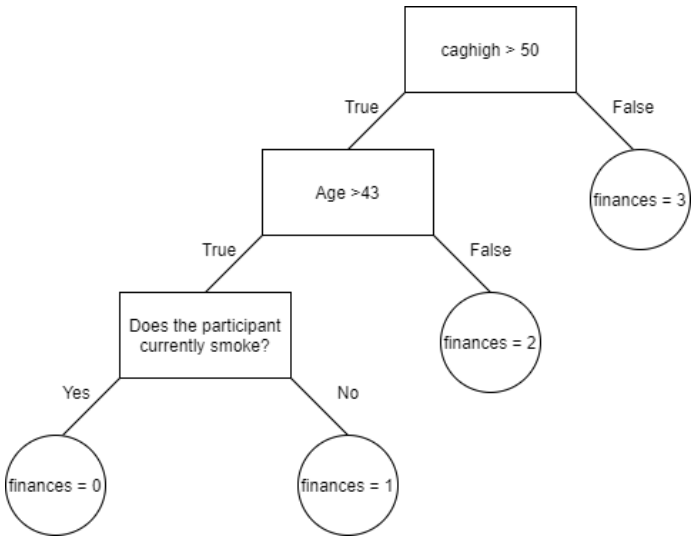


Figure 4.13: *Example of a simple decision tree that predicts the finances variable. Here the rectangles are the nodes and the circles are the leafs.*

The bagging method is used to obtain a diverse set of models (decision trees) by training the models on randomly sampled subsets, with replacement, on the training set. After training each predictor the random forest model makes a prediction by aggregating the predictions of all models. Here the most frequent prediction is taken for classification problems and the mean is taken for regression problems. One of the advantages of random forest is that the random forest generally has a similar bias but a lower variance than one decision tree trained on the whole training set.[19]

4.4.2 Linear Support Vector Machine (SVM)

The fundamental idea behind a Support Vector Machine (SVM) is that a hyperplane can be drawn that separates two classes for classification or that tries to fit as many observations as possible on the hyperplane and within the margin for regression. This hyperplane is drawn based on support vectors, which are the data points nearest to the hyperplane. The space between the support vectors and the hyperplane is called the margin.[19]

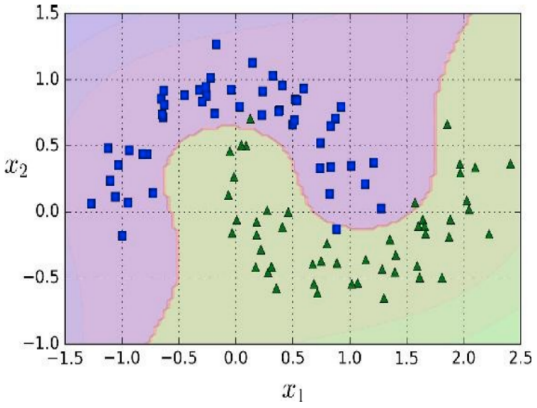


Figure 4.14: *An example of a SVM classification here the SVM tries to increase the distance between the hyperplane.*

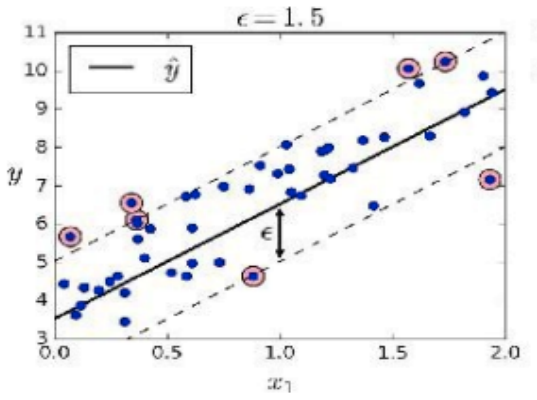


Figure 4.15: *An example of a SVM regression here the size of the margin is determined by epsilon (ϵ).*

4.5 Baseline/Constant Model

To have a good performance evaluation a baseline model is also evaluated, namely the constant model. This model simply predicts the latest known label, e.g. to predict the label at time point 1 the label at time point 0 is used, which is illustrated in the figure below. Here i is the position of the label in the input matrix given to the model.

$$\begin{bmatrix} x_{00} & x_{10} & \dots & x_{n0} \\ x_{01} & x_{11} & \dots & x_{n1} \\ x_{02} & x_{12} & \dots & x_{n2} \\ \dots & \dots & \dots & \dots \\ x_{0t-1} & x_{1t-1} & \dots & x_{nt-1} \end{bmatrix} \xrightarrow[predict]{} \begin{bmatrix} x_{i0} \\ x_{i1} \\ x_{i2} \\ \dots \\ x_{it-1} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \\ \dots \\ \hat{y}_t \end{bmatrix}$$

Figure 4.16: *An example of how the baseline model predicts the next time step using the label at position i in the previous time step.*

Chapter 5

Methods

5.1 Software & Data Availability

All the work was done in *Python 3.8.5* and *Jupyter Notebook 1.0.0* using the libraries *numpy 1.19.5* and *pandas 1.1.3* for handling the data, *tensorflow 2.4.1* and *sklearn 0.24.1* for training the models and *seaborn 0.11.1*, *Venny 2.1.0*, and *matplotlib 3.3.3* for plotting figures and finally *shap 0.39.0* also for plotting and interpreting the models. In this report the fifth Enroll-HD periodic dataset (PDS5) was used, which is available at: `https://enroll-hd.org/for-researchers/access-data`. The code used in this research is available at: `https://git.liacs.nl/s2494876/thesis`.

5.2 Predicting Disease Progression/Prognosis

As mentioned in the goals of the project (section 1.2), the primary goal is to give a prognosis on the progression of the cUHDRS and highlight if the variables affecting the cUHDRS are the same as the variables that make up this score (TMS, TFC, SWRT, SDMT). After that a deeper analyses is done on the components making up the cUHDRS to show which variables affect the components making up the cUHDRS. This is done by predicting one time step ahead in time, circa one year ahead in time, for the cUHDRS and the variables making up the score. For the driving capability the current time point is predicted, since an advice on driving capability needs to be given on the current situation for the patient. How the data is transformed is explained in more detail in the next section Model Input.

5.3 Pre-Processing

In addition to previous pre-processing and imputation steps, more pre-processing steps are needed to use machine learning and deep learning models for longitudinal data, these

steps include reshaping the data and transforming the input to let the machine learning and deep learning models understand what time step to predict. Also, some general pre-processing steps are explained, these include: feature selection, train and test split, data scaling, and assigning train and test labels, i.e. what to predict. In the following sections these steps are elaborated in the order of how the data is processed.

5.3.1 Feature Engineering

Most feature engineering was done in the previous project, however two additional features were created, namely the cUHDRS variable and the cognitive score (cogscore), to make the variables included in the cognitive assessment more interpretable.

The cUHDRS can be created using equation 5.1, which was deducted from Schobel et al.[16] (2017). The cUHDRS is made up of the Motor Score (TMS), Function Score (TFC), Symbol Digit Modality Test (SDMT), and the Stroop Word Reading Test (SWRT).

$$\text{cUHDRS} = \frac{\text{TFC} - 10.4}{1.9} - \frac{\text{TMS} - 29.7}{14.9} + \frac{\text{SDMT} - 28.4}{11.3} + \frac{\text{SWRT} + 66.1}{20.1} + 10 \tag{5.1}$$

The cognitive score was created by taking the first principal component (PC) of a principal component analyses (PCA) on all the main cognitive variables, namely: sdmt1, verfct5, scnt1, swrt1, sit1, trla1, trlb1, and verflt05. These variables measure either the number of correct answers or the time needed to complete a cognitive test. The PCA and how much each PC explains the variance in percentages is shown in figure 5.1. This figure shows that the the first PC (PC1) explains the variance for 77.7% which was sufficient for our use case.

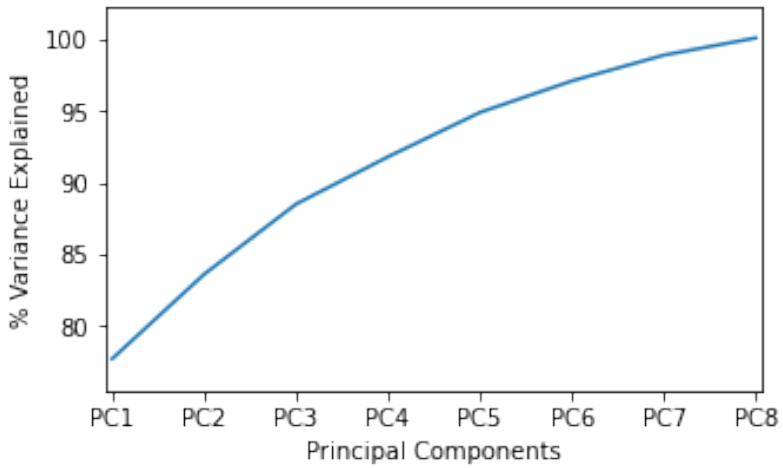


Figure 5.1: *The explained variance in percentage for each PC. This first PC (PC1) explains the most variance with 77.7%.*

5.3.2 Feature Selection

Before modeling some variables are discarded beforehand. Variables that are discarded include variables that refer to a date, whether a specific test was done during a visit, and low quality variables. Low quality variables include variables that can be inferred from other variables, e.g. whether a patient drinks alcohol (alcab) can be inferred from the number of alcohol units they drink per week (alcunits) or variables that are almost always equal to one value. All the selected variables and their statistics are shown in appendix B and the distribution of the selected variables is shown in appendices C and D. In appendices B and C all the categorical variables are displayed as single variables to make the distribution and statistics of these variables more interpretable. However, some of these variables are encoded using dummy encoding, since the variables contain missing values. Dummy encoding ensures that missing values are pre-processed in a way that the models can interpret them. An example of a dummy encoding of the categorical variable 'handedness' is shown in table 5.1. Which variables are transformed using this encoding are shown in appendix E. In total 191 variables are given as input to the models after dummy encoding.

Table 5.1: *Illustration of dummy encoding, here each row represents a visit.*

Handedness		**handedness_1**	**handedness_2**	**handedness_3**
right (1)		1	0	0
left (2)	→	0	1	0
mixed (3)		0	0	1
NaN		0	0	0

5.3.3 Feature Selection - Drive

When modelling the driving capability one feature is removed from the dataset, namely the functional assessment score (fascore). The fascore is the sum of all variables measures in the functional assessment form, which includes the drive variable. All variables making up this score are also added to the data, which makes it quite simple to deduct the drive variable, namely the fascore minus the sum of all components would equal the drive variable. Therefore, the fascore is removed and of course the driving capability is also removed since the current time step is predicted.

5.3.4 Reshaping the Data

To allow for accurate and faster training of the NN models it is necessary to reshape the datasets in a way that each participant has the same number of visits and that the time between the visits is consistent. In our case there are around 1.4-5.0 visits for each participant and the visits are generally 1 years apart. With this in mind, it was decided to

have a maximum of five visits that are 1 year apart, i.e. the dataset is transformed from $(\#participants \times \#visits, \#features)$ to $(\#participants, 5, \#features)$. To achieve this the index of each existing visit is first calculated using the equation below, where v_{pi} is the new visit index of visit i for participant p, d is the number of days from the baseline visit (visdy) and t is the desired number of days between each visit.

$$v_{pi} = \lfloor (d_{pi} - d_{p0})/t \rceil \tag{5.2}$$

In our case $t = 365$, this ensures that all patients have a first visit and that the next visits are around 1 years apart. As mentioned before not all participants have the same number of visits or time between visits. Therefore, any remaining missing visits are masked. This is done by filling the visit vector with an arbitrary mask value, which the RNN can detect and skip using a masking layer, making training faster. This is further elaborated in section 5.4 General Neural Network Design.

5.3.5 Train and Test Split

To create the train and test subsets the holdout strategy is used, i.e. the dataset is split into two by randomly picking 80% of the patients for the training set and the other 20% are assigned to the test set. However this splitting is done on groups of samples. These groups are defined by the number of missing visits for the patient. This will ensure that both the training set represents the test set. For the progression models a sample/patient is discarded if only one visit is available, since there is no visit to train on in that case.

5.3.6 Model Input

To ensure that all the proposed models can be compared fairly, the models need to be trained on the same labels. Since, the predictions on all the time steps are of interest, the models have to predict a value at each time point. This is quite straightforward for RNNs, since the RNN loops over each time point one by one and can output the prediction made at each iteration, which results in a prediction for each time point, without the model being able to look ahead into the next time point, where the label is known, see figure 4.7.

$$\begin{bmatrix} x_{00} & x_{10} & \dots & x_{n0} \\ x_{01} & x_{11} & \dots & x_{n1} \\ x_{02} & x_{12} & \dots & x_{n2} \\ \dots & \dots & \dots & \dots \\ x_{0t-1} & x_{1t-1} & \dots & x_{nt-1} \end{bmatrix} \xrightarrow[predict]{} \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \\ \hat{y_3} \\ \dots \\ \hat{y_t} \end{bmatrix}$$

Figure 5.2: *Example of how a RNN predicts the labels ($\hat{y}$). Note that for time step (t) a value (the label) from the next time step ($t+1$) is predicted, since the models are trained to predict one time step ahead for the cUHDRS and for the driving capability the current time step is predicted.*

This is not the case for CNNs and the ML models, since those models look at the whole input directly and predict t labels or even only one, since not all machine learning models support multi-output problems. This is illustrated in figure 5.2, where there are n input variables and t time steps. This is a problem, since to allow for a fair comparison between RNNs, CNNs and ML models the models need to be trained on the same input data and labels. To achieve this the input for the NNs, CNNs, and ML models are transformed. This is done by duplicating the input matrix in figure 5.2 into t copies, one for each time point. After that the time point that has to be predicted is masked using a masking value m, indicating to the model which time step has to be predicted, see figure 5.3. Which time point needs to be predicted is learned by the models, however this might cause some issues when a time point is missing, since missing visits are also masked using the same value m. An example of such a case is shown in figure 5.4, where time point 1 is missing. This would indicate that the first time point has to be predicted ($\hat{y_1}$). Therefore, all missing time points are filled in with the last non-missing visit.

$$\begin{bmatrix} x_{00} & x_{10} & \dots & x_{n0} \\ m & m & \dots & m \\ m & m & \dots & m \\ \dots & \dots & \dots & \dots \\ m & m & \dots & m \end{bmatrix} \xrightarrow[predict]{} \hat{y_1} \quad \begin{bmatrix} x_{00} & x_{10} & \dots & x_{n0} \\ x_{01} & x_{11} & \dots & x_{n1} \\ m & m & \dots & m \\ \dots & \dots & \dots & \dots \\ m & m & \dots & m \end{bmatrix} \xrightarrow[predict]{} \hat{y_2}$$

Figure 5.3: *Example of how CNN and ML get their input like a RNN to predict time point 1 (left) and 2 (right). Note that ML models require one dimensional data, so the matrix in this example will be flattened when provided to the model.*

$$\begin{bmatrix} x_{00} & x_{10} & \dots & x_{n0} \\ m & m & \dots & m \\ m & m & \dots & m \\ \dots & \dots & \dots & \dots \\ m & m & \dots & m \end{bmatrix} \xrightarrow[fill]{} \begin{bmatrix} x_{00} & x_{10} & \dots & x_{n0} \\ x_{00} & x_{10} & \dots & x_{n0} \\ m & m & \dots & m \\ \dots & \dots & \dots & \dots \\ m & m & \dots & m \end{bmatrix} \xrightarrow[predict]{} \hat{y_2}$$

Figure 5.4: *Example of how a missing visit is filled in, when $\hat{y_2}$ needs to be predicted for the NN, CNN, and ML models.*

5.3.7 Data Scaling

Before any modelling can take place the continuous input data is scaled to a range of -1 to 1 using the MinMaxScaler from sklearn.[33] This function transforms the input data as shown in equation 5.3. Scaling the data is essential before training a NN, since a NN generally does not perform well when the features are on different scales.[19]

$$x' = \frac{x - min(x)}{max(x) - min(x)} \times 2 - 1 \tag{5.3}$$

5.3.8 Assigning Labels and Sample Weights

When assigning labels for each time point the desired value at the next visit is taken. In this way the model tries to predict a value at the next visit for each time point.

However, in some cases there is no data for a patient at that specific time point which means that the label is missing and masked with the masking value m. In such a case the weight for that label is set to 0, i.e. it is not used to calculate the loss of the model. This ensures that the model is not trained on the masking value m.

A second problem is that a label might be imputed. However, it was decided not to train on imputed labels, since such data can contain errors and is therefore of lower quality. In such a case the sample weight is also set to 0. Whether a label is imputed or not can be deducted from the pre-imputed dataset.

Lastly, when predicting a class variable, e.g. the driving capability a sample weight is calculated based on the frequency of the different classes. To calculate this weight equation 3.4 is used. This class weight is calculated based on the pre-imputed dataset, since the frequency of each class can be different for the imputed dataset.

5.3.9 Overview

To summarize a flowchart of the pre-processing steps is shown in figure 5.5. Here the pre-imputed and imputed dataset was created in the previous introductory research project, see figure 2.3. Firstly, both datasets are reshaped to make longitudinal predictions, described in section 5.3.4. Secondly, both datasets are split in the train and test sets, described in section 5.3.5. Thirdly, both datasets are transformed to fit the specific model

that needs to be trained on, described in section 5.3.6. After that the imputed dataset is normalized/scaled, described in section 5.3.7, and the sample weights are calculated based on the pre-imputed and imputed dataset, described in section 5.3.8.

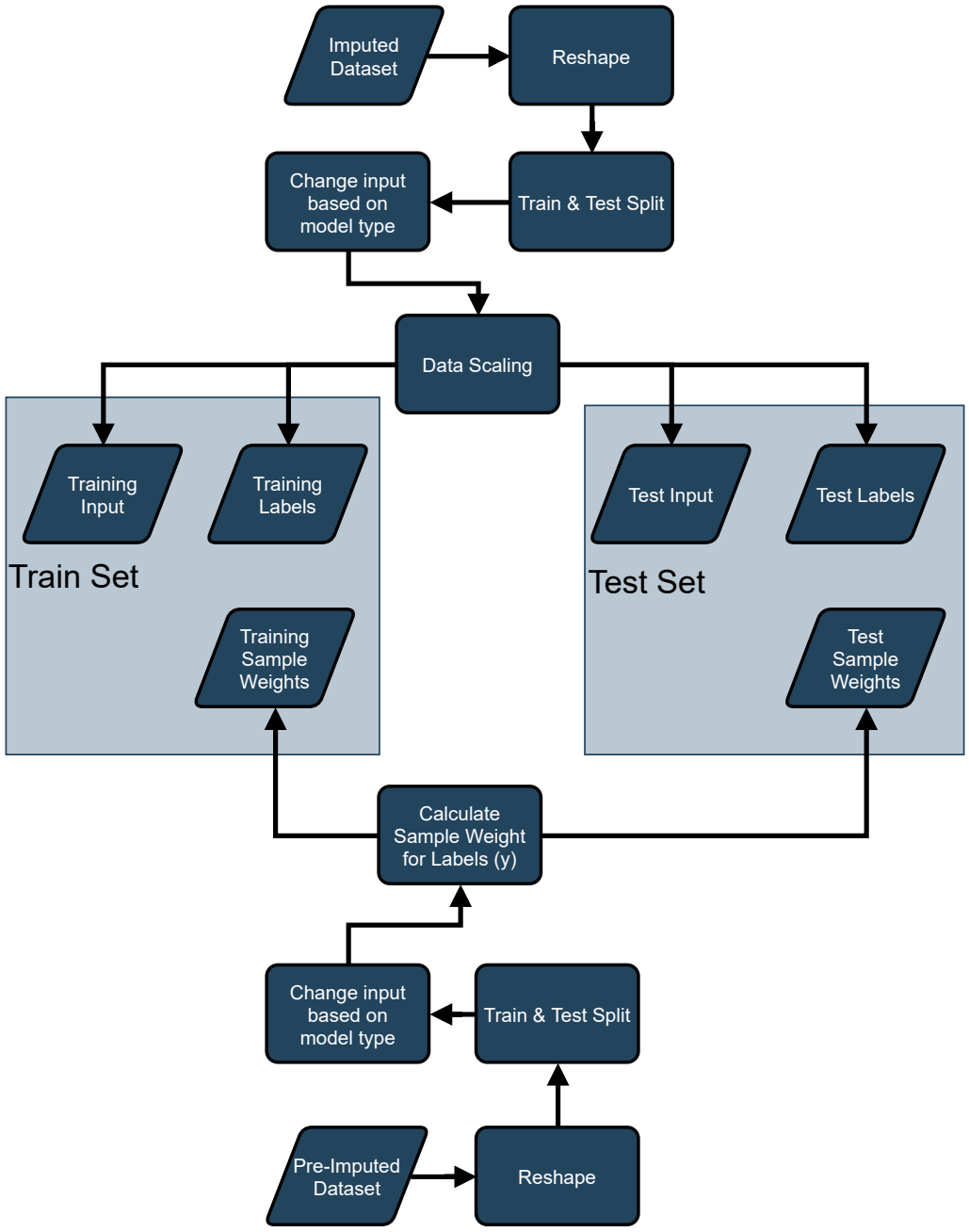


Figure 5.5: *The general workflow of pre-processing the pre-imputed and imputed dataset to train all the models. Here specific feature selection/engineering steps required for the different labels are ignored.*

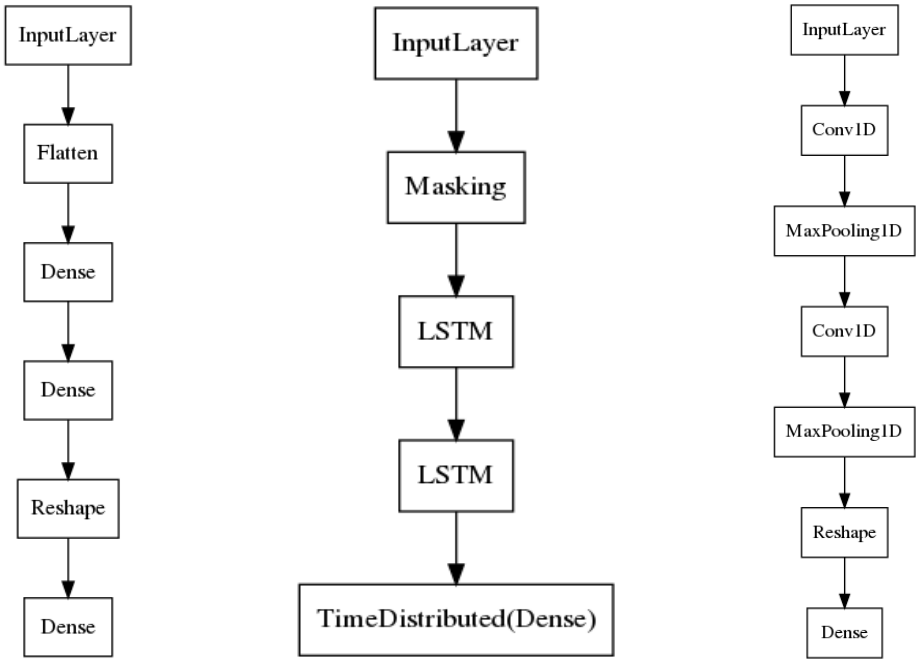
5.4 General Neural Network Design

In this section the chosen network design is illustrated for the RNNs, the CNNs and the normal NNs. The design of the networks is an important factor when creating networks. In this case the networks were designed by only using the network specific layers until the last layer. The last layer is a normal dense layer to make a prediction. An example of the network design, with three layers (a hyper-parameter), for each type of network can be seen below in figure 5.6.

The NN design, shown in figure 5.6a, starts with a normal input layer followed by a flattening layer, since the NN can not handle two dimensional data. After that the input flows through the dense layers (two layers in this example) and the output of those layers is activated using a RELU function, see figure 4.3b. After that the output of the last dense layer is reshaped to the proper shape and the last dense layer creates the final prediction using either a linear activation function (figure 4.3a) for a regression problem or a sigmoid/softmax activation function (figure 4.3d/4.3e) for a binary/multi-class classification. The last dense layer uses the same design for both the RNNs and the CNNs.

The RNN design, shown in figure 5.6b, starts with a normal input layer followed by a masking layer. The masking layer tells the RNN to skip missing visits, i.e. visits which are completely filled with masking value m. This masking layer is followed by recurrent layers, in this example LSTM layers are used, which are always activated using a tanh function (figure 4.3c) and the network ends with a time distributed dense layer. The time distributed layer simply tells the model to apply the dense layer to each time step to predict the desired value.

The CNN design, shown in figure 5.6c, starts with a normal input layer followed by a convolutional layer and a max pooling layer. This is repeated for the number of layers given, i.e. the combination of the convolution layer and the max pooling layer is considered as one layer. Here the convoltion layers are always activated with a RELU (figure 4.3b). After the convolution and pooling layers the output is reshaped into the proper shape and ends with the dense layer as the output layer.



(a) *The NN design, the input and output need to be reshaped since a NN can not handle two dimensional data.*

(b) *The RNN design with a LSTM layer as the recurrent layer.*

(c) *The (1D) CNN design, after each convolutional layer a maxpooling layer is used.*

Figure 5.6: *The designs of the NN, RNN, and the CNN when two layers are used.*

5.5 Hyperparameter Tuning

5.5.1 Neural Networks

All the neural network hyperparameters are shown in table 5.2, this table shows that only the hyperparameters L2 regularization, and the hidden size/filters are tuned. Here the hidden size indicates the number of neurons in each layer for the NN and the RNN and the filters indicates the number of filters per layer in the CNN. The static hyperparameters for the networks are the maximum iterations, the patience, batch size, the learning rate, the optimizer, and the number of layers and the CNN has three additional static hyperparameters, namely the filter size, the padding and the stride, which are shown in table 5.3. Any unnamed hyperparameters are set to the default value, which results in 6 neural network models per type, i.e. NN, CNN, SimpleRNN, LSTM, and GRU.

Table 5.2: *All the hyperparameters tested for all the types of neural networks.*

Hyperparameter	Range	Step
Maxiter	10,000	-
Patience	100	-
Batch Size	128	-
Learning Rate	10^{-5}	-
Optimizer	Adam	-
#Layers	3	-
L2 regularization	10^{-7}-10^{-3}	$\times 10^2$
Hidden Size/Filters	128-256	$\times 2$

Table 5.3: *All the constant hyperparameters specific for the CNNs.*

Hyperparameter	Value
Filter Size	3
Filter Strides	1
Padding	same (zero)

5.5.2 Linear Support Vector Machine (SVM)

All the SVM hyperparameters are shown in table 5.4, this table shows that the parameters Epsilon and Penalty (C) are tuned. One static hyperparameter is given, the maximum iterations, which is set to 10,000. All combinations of these parameters are tuned which results in 9 SVM models. All other unnamed hyperparameters are set to the default value.

Table 5.4: *All the hyperparameters tested for all the SVMs.*

Hyperparameter	Range	Step
Maxiter	10,000	-
Epsilon	$10^{-2} - 10^{0}$	$\times 10^1$
Penalty (C)	$10^{-1} - 10^{1}$	$\times 10^1$

5.5.3 Random Forest

All the Random Forest hyperparameters are shown in table 5.5, this table shows that the Criterion and n_estimators parameters are tuned. Here the criterion is either Gini or Entropy when predicting the driving capability and for the other variables the MSE is used. All combinations of these parameters are tuned which results in 5/10 Random Forest models. All other unnamed hyperparameters are set to the default value.

Table 5.5: *All the hyperparameters tested for all the random forest models. Here the criterion is either Gini or Entropy when predicting the driving capability and for the other variables*

Hyperparameter	Range	Step
Criterion	MSE / Gini or Entropy	-
n_estimators	100-1600	×2

5.6 Analyses

5.6.1 Overall Performance

After all the models have been trained and tuned a performance evaluation is done. In this evaluation the performance of all tuned models are shown and based on the best performing tuned models it is decided which type of model performed the best, e.g. LSTM or CNN. The models are evaluated on average performance, temporal performance (performance for each year ahead in time), and performance per status group (performance based on how the label changed over time). The models that were defined as the best model based on previous criteria are further analyzed using the SHAP algorithm and more performance evaluations, e.g. ROC curve and confusion matrix.

5.6.2 Interpreting the Model (SHAP)

One of the downsides of NNs is that they are not very interpretable on their own. Therefore, SHapley Additive exPlanation (SHAP) is used to make the best performing model more interpretable. SHAP is a method of measuring a sample's (patient's) variable importance on the predicted outcome, i.e. a personalized impact panel. This is done by measuring how a feature affects the expected outcome of a model, e.g. if the feature has a negative effect the SHAP value of that feature will be negative. This means that the sum of the expected value and all the SHAP values should equal the predicted value. Here the global feature importance of each variable is measured by taking the mean absolute SHAP value of each feature for each prediction.[34]

5.6.3 Overview

To summarize a flowchart of the hyperparameter tuning and analysis steps are shown in figure 5.7. First all the models are trained tuned on different hyperparameters on the train set. Next, all the trained models are evaluated on the test. Based on these evaluation metrics the best model is chosen, which include the average results, the temporal results, and the results based on different status groups. When the best model is selected the SHAP analysis is done to deduct which variables were important in predicting the output.

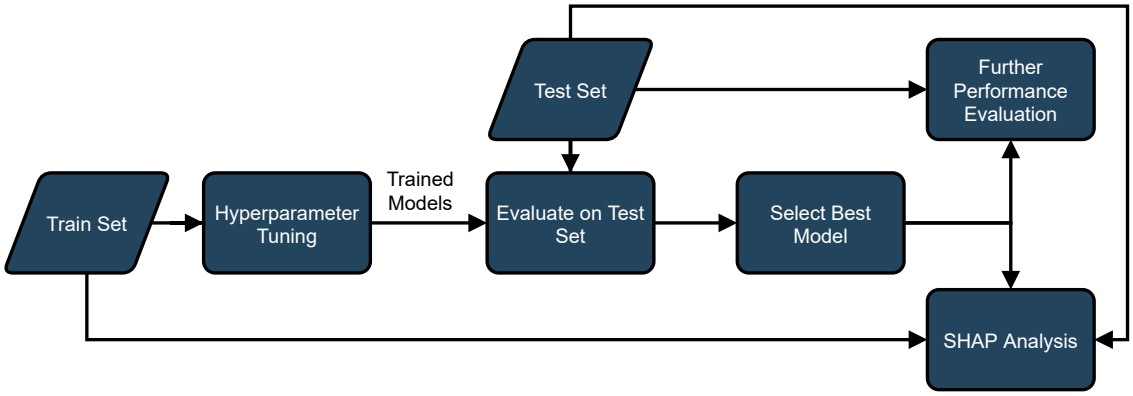


Figure 5.7: *Flowchart of the hyperparameter tuning and analysis steps.*

Chapter 6

Results

Here all the results are presented for modeling the driving capability and the disease progression. First the results for modelling the driving capability are shown. Secondly, the results for modelling the cUHDRS progression are shown, which is followed by the TMS, TFC, SDMT, and the SWRT. First, for each variable modelled the statistics of the train and test set are presented. This is followed by the results of all the tuned models. Here the best performing model is deducted based on the performance of each machine learning model. With this best model the SHAP values are calculated and presented to deduct which variables affect the outcome of the model.

6.1 Driving Capability

6.1.1 Model Performance

First the driving capability was modelled. The driving variable is equal to 0 when a HD patient is unable to drive and the driving variable is equal to 1 when a HD patient is able to drive. Here the drive variable was predicted for each current visit, while also considering information from previous visits. The distribution of the labels in both the training and test set are shown in figure 6.1. In total around 58% of the training and test data is labeled. Some visits are not labelled, since not all patients visited the clinic each year, see figure 2.4b. In total 50% of the labels were masked simply because the drive variable was not filled in and the models are not trained on imputed data. The shape of the train and test sets are shown in table 6.1. Here the dataset for the RNN models is smaller than the dataset for the other models, since the samples had to be duplicated for each time step for the non-RNN models, which is descried in the Model Input section.

All the models were evaluated on three performance measures, namely the AUC, F1, and accuracy score. All the models and their evaluation metrics are shown in appendix F. All the models are also evaluated to define which model works best for each output. To define the best model the average metrics, the metrics per time step, and the metrics

Table 6.1: *All the train and test stats of for modelling the driving capability.*

Model	Subset	Shape x	Shape y & sample weights
RNN	Train	(12396, 5, 191)	(12396, 5, 1)
RNN	Test	(3031, 5, 191)	(3031, 5, 1)
Other	Train	(61980, 5, 191)	(61980, 1, 1)
Other	Test	(15155, 5, 191)	(15155, 1, 1)

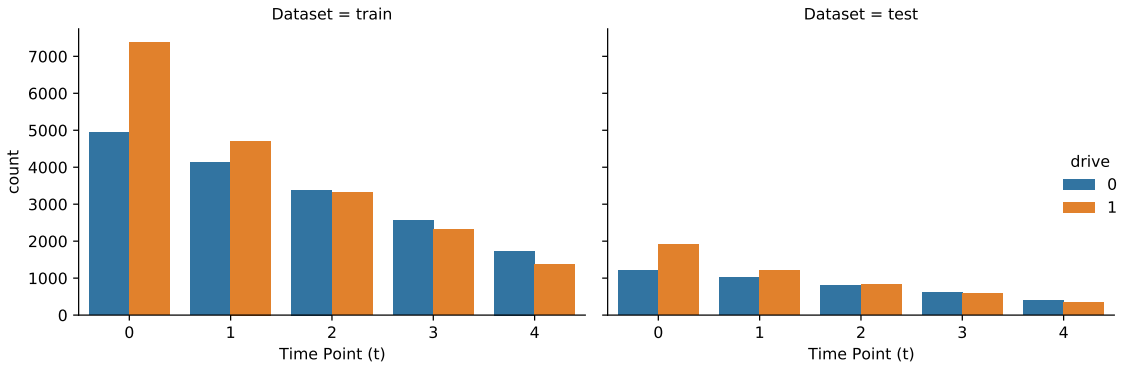


Figure 6.1: *Labels of the train and test set for the driving capability.*

per status group are shown. Here the best model can be deducted by evaluating which model performed best at each evaluation step.

Below the average over all visits of all the metrics are shown in figure 6.2. This figure shows that all the neural network models perform the best and their values are comparable to all metrics, however the GRU has the highest F1 and Accuracy score. Another thing to note is that the RF and SVM models always perform worse than the neural network models, especially in the AUC metric with a difference around 0.08 in AUC.

Figure 6.3 shows that the AUC is very similar along all neural network models between all time points. The metrics for the neural networks are different in terms of F1 and accuracy, however the difference is negligible.

Lastly, the performance per progression group is shown in figure 6.4. Here the groups are defined by the change in the driving capability between the first visit and last visit. The different groups are defined as follows, the stable group includes patients which had no change in the driving capability between the first and last visit, the recovered group includes patients which did not drive in the first visit and started driving again in the last visit, and lastly the progressive group includes patients which drove in the first visit and stopped driving in the last visit. Figure 6.4 shows that the stable group (n=2693) is best

predicted by the neural network models. The recovered group (n=28) is best predicted by the NN in terms of AUC, the CNN or NN in terms of F1, and the SVM in terms of accuracy. This difference in best models might be caused by the small size of this group (n=28). The progressive group (n=267) is best predicted by the neural network models, specifically the CNN, however the difference is negligible. Here the sample size is again quite low (n=267). These low sample sizes show that the models do not have many examples like this to train on, which might explain why the metrics for these models are lower than the larger stable group.

Considering that the GRU had the highest AUC, F1, and accuracy on average it is decided to further analyze the GRU model, even though all the neural networks perform similarly in all other aspects.

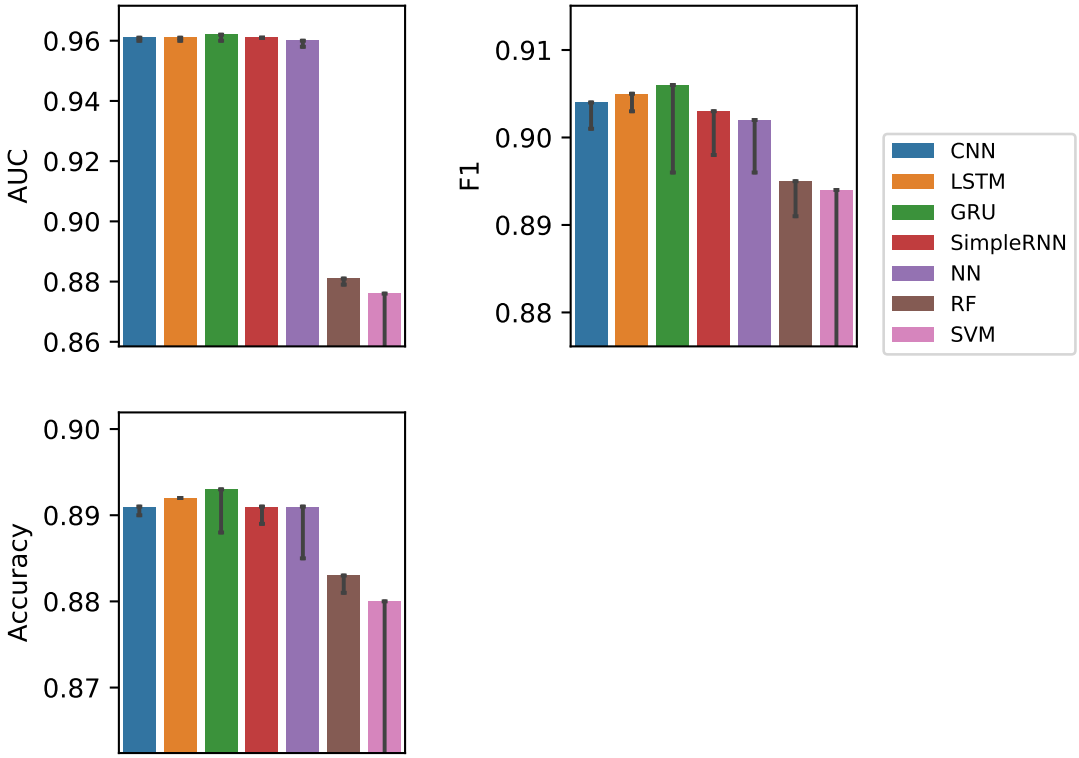


Figure 6.2: *The AUC, F1, and accuracy score of all the tuned models for the driving capability. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

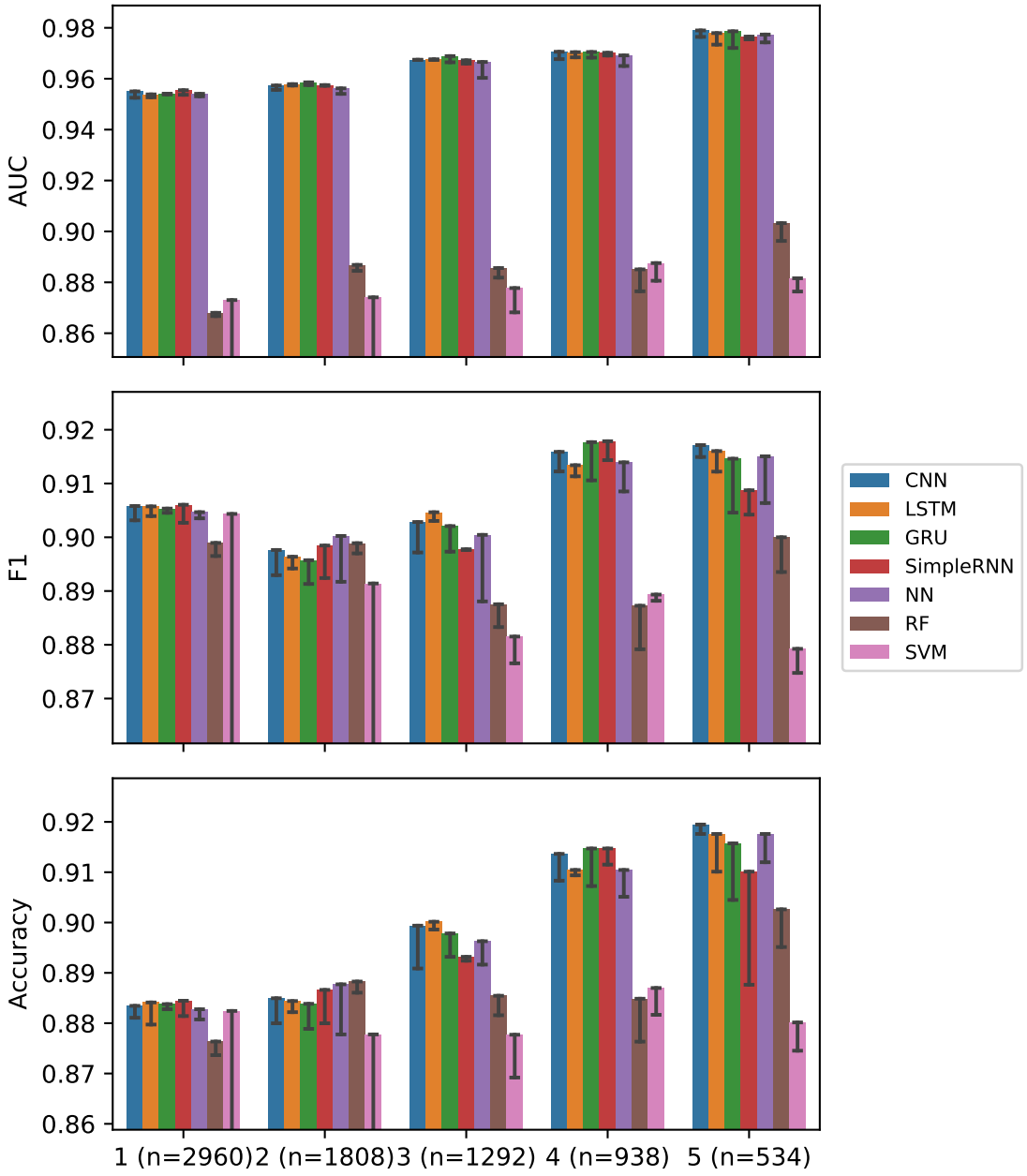


Figure 6.3: *The AUC, F1, and accuracy score of all the tuned models for the driving capability over all the time points. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

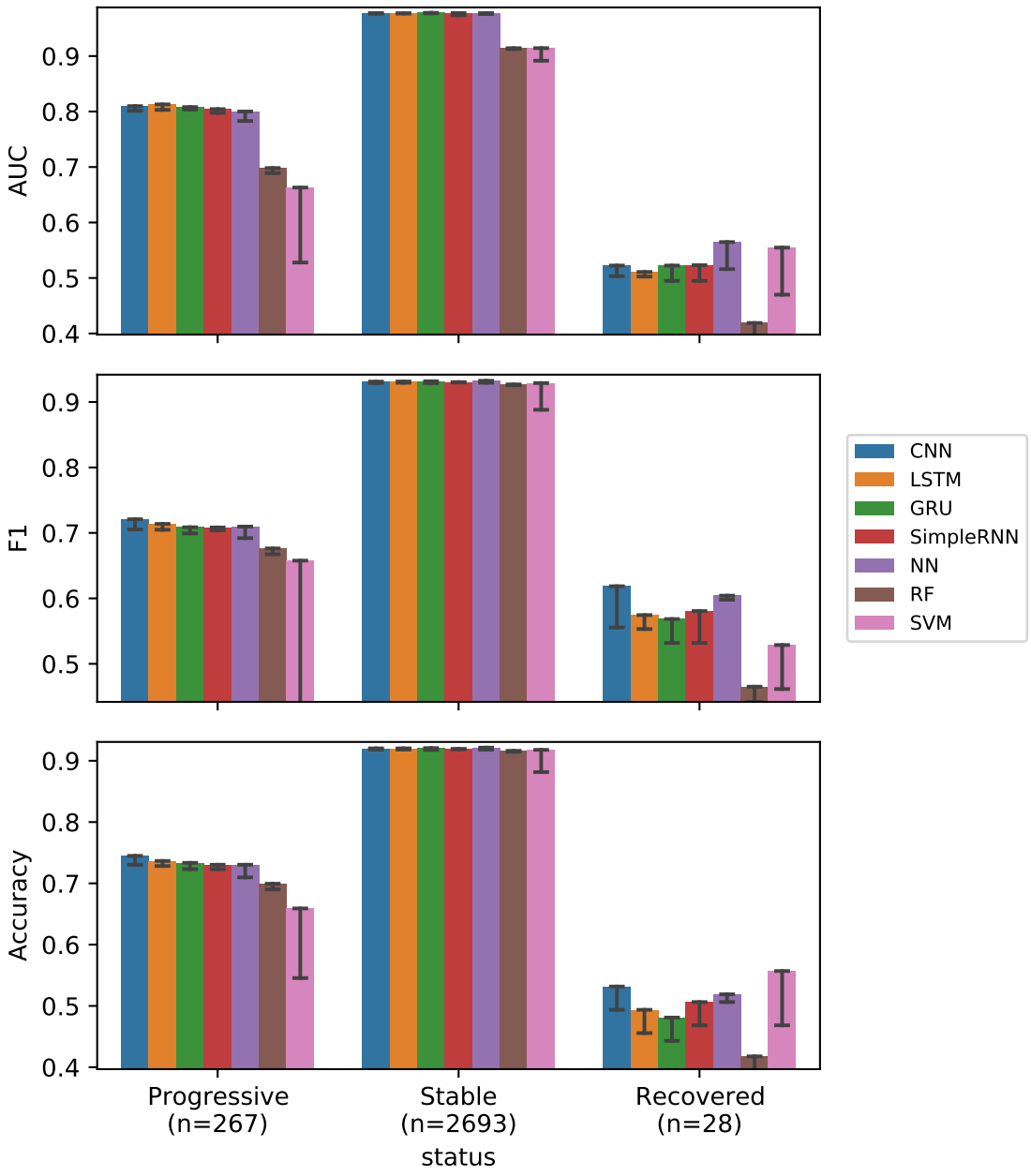


Figure 6.4: *The AUC, F1, and accuracy score of all the tuned models for the driving capability over all the status groups consisting of stable (no change in driving capability between first and last visit), progressive (patient drove a car in the first visit, but stopped in the last visit), and recovered (the patient did not drive a car in the first visit and started driving again in the last visit). Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

6.1.2 Model Interpretation

In this section the GRU model, which performed the best to predict the driving capability, is further analyzed. In figure 6.5, the confusion matrix is shown. This figure shows that the percentage of true positives (TP) and true negatives (TN) is quite similar. Around 91% percent of the positive samples (1, able to drive) are actually predicted as positive and around 85% percent of the negative samples (0, unable to drive) are actually predicted as negative. This means that the model is somewhat better at predicting true samples than negative samples. This might be due to the higher number of training samples for the positive label than the negative labels, see figure 6.1.

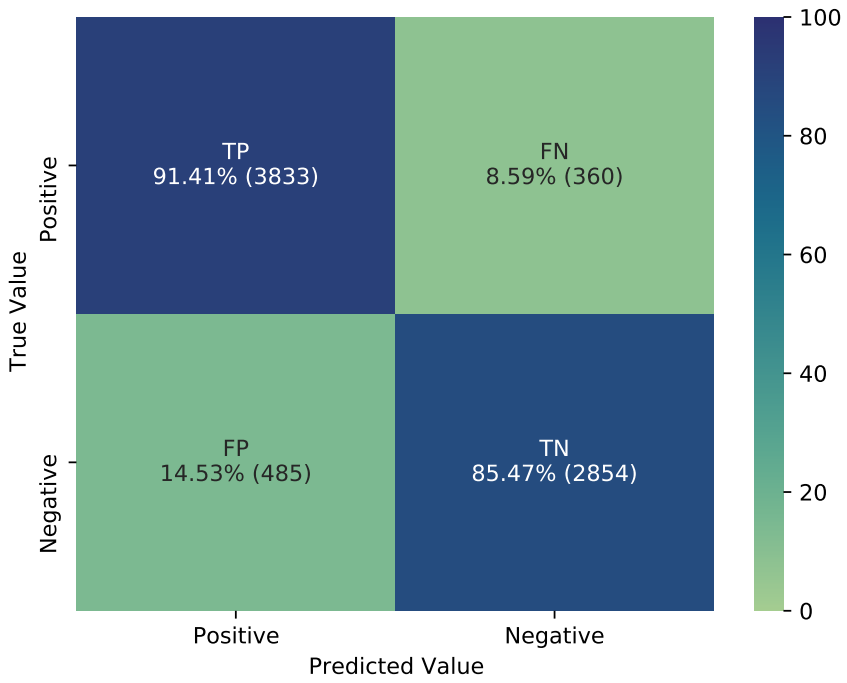


Figure 6.5: *The confusion matrix on the test set. Here the percentages refer to the percentage of positive samples (able to drive) predicted as a positive (TP) and the percentage of positive samples predicted as a negative (FN). The same is true for the negative samples (unable to drive).*

In figure 6.6 the personalized predictions at each time step are plotted for one patient, regarding their ability to drive. Note that the true labels are either 0 or 1 and the predicted labels can range from 0 to 1, where a value above the threshold of 0.5 are predicted as label 1 and a value below the threshold is predicted as 0. In figure 6.6 the labels are also plotted against a heatmap which is made up of ten rows/bands with an accuracy score, depicted by the color. This accuracy is deducted by taking all samples

from the test set which were predicted within in this band and calculating the accuracy within this sample, which results in an accuracy score for each row, making the prediction more interpretable, i.e. a predicted value close to the threshold is less accurate than a predicted value close to 0 or 1.

This figure also shows that at one time step the predicted value does not match the actual label, however this does not necessarily imply a prediction error. It could also indicate a wrong assessment by the clinic. In this case the patient might still have been able to drive, according to the model, which would be an improvement of their quality of life, since the patient has more freedom of movement. On the other hand it might be unsafe for the patient and others to let the patient drive. In such a case it might be better that the clinician and patient have a deeper look into the patients driving capability. Note that this model is an advisory system. No definitive answers should be drawn from these predictions.

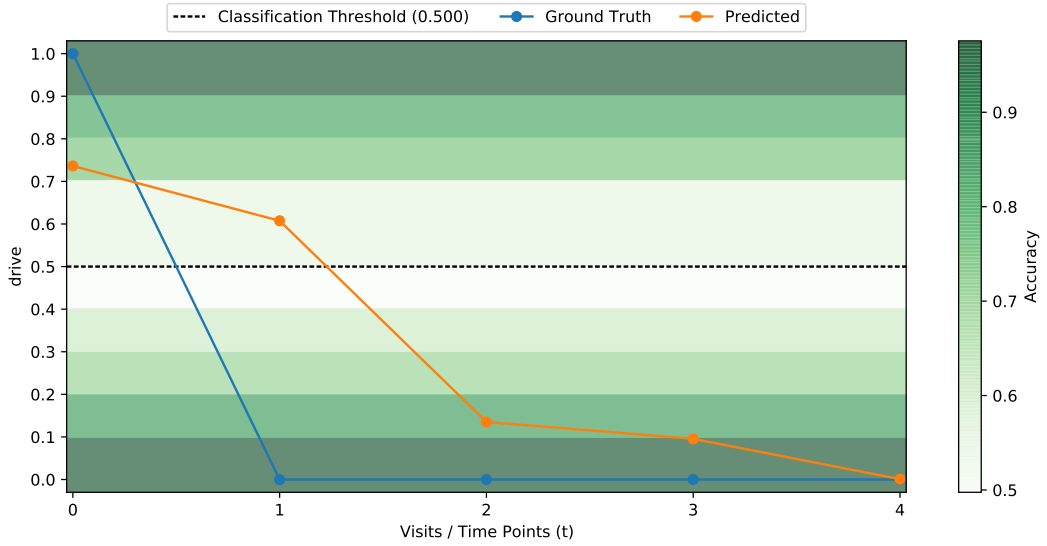


Figure 6.6: *An example of estimating the driving capability of a patient at each time step. Here one misclassification is shown at time step 1, however this can also be interpreted differently.*

Theoretically these uncertain cases can be negated by deciding what is more important, the safety of others and the patients or the freedom of the movement of the patient. How these theoretical cases would affect patients is shown in the ROC curve, see figure 6.7. Here the ROC curve of the model on the test set is shown.

The safety of the patient and others can be prioritized by setting the classification threshold to 0.94 (T=0.94) and thus not allowing any patient to drive when predicted below this threshold. This would increase the true negative rate (TNR) to 0.99 (TNR = 1-FPR), see figure 6.7, however that would result in many patients getting a negative

advice while they could still drive according to the data, which is shown by the high false negative rate (FNR) of 0.38 (FNR = 1-TPR).

The freedom of movement can be prioritized by setting the classification threshold to 0.11 (T=0.11) and thus allowing all patients to drive when predicted above this threshold. This would increase the true positive rate (TPR) to 0.99, see figure 6.7, however that would result in many patients getting a positive advice while they do not drive according to the data, which is shown by the high false positive rate (FPR) of 0.34.

Here we do not make the decision for the patient and clinician. For now the decision is left to the patient and clinicians, since the intention is to use the model as an advisory system. Therefore, the threshold is set to a default of 0.5, which results in a TNR of 0.86 (TNR = 1-FPR) and a TPR of 0.91, see figure 6.7. In general the GRU can also almost perfectly separate the two classes, since the AUC is close to 1 and the ROC line is close to the top left corner.

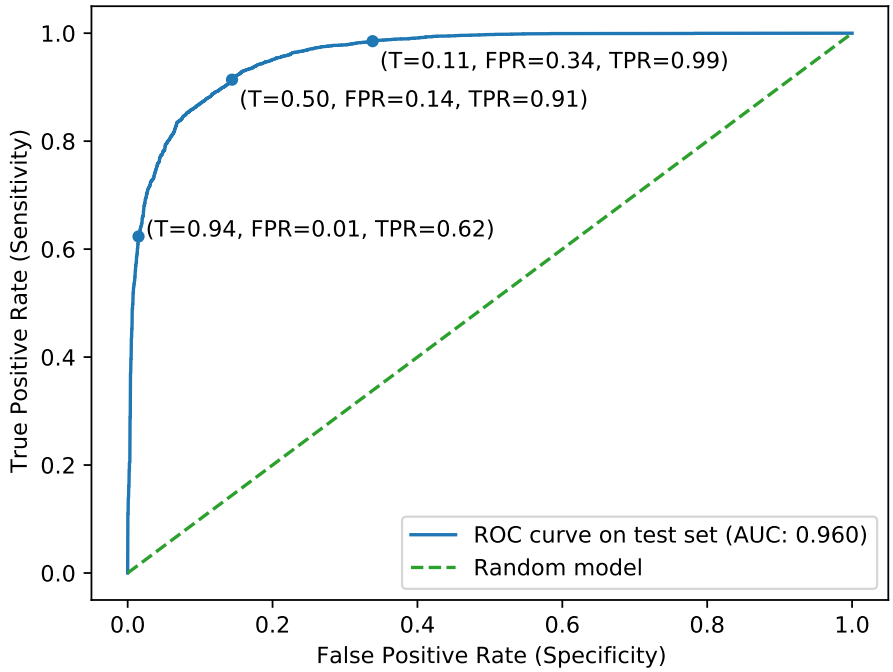


Figure 6.7: *The ROC curve calculated on the test set, resulting in an AUC of 0.96.*

In figure 6.8 the top 20 highest mean absolute SHAP value of all patients in the test set for each variable is shown. The SHAP values represent the importance of each variable to predict the output variable. This figure shows that the indepscl (subject's independence in %) and supchild (could subject supervise children without help) variables have the highest mean absolute SHAP value to predict the driving capability of a patient.

This figure also shows that many components of the functional assessment score (fascore), which includes the drive variable, are present in the top 20 SHAP values. The other fascore components are indepscl, supchild, grocery (could subject shop for groceries without help?), housewrk (could subject do his/her own housework without help?), volunt (could subject engage in any kind of volunteer or non-gainful work?), toilet (could subject use toilet/commode without help?), walknbr (could subject walk to places in his/her neighbourhood without help?), ownmeds (could subject take his/her own medications without help), pubtrans (could subject use public transport to get to places without help?), fafinan (could subject manage his/her finances (monthly) without any help?), and bathe (could subject bathe himself/herself without help?). This shows that the driving capability is very related to these other functional components.

What is odd, is that no components of the TMS are in the top 20 SHAP values. It is quite odd that motoric issues are less important in predicting driving capability than all the other variables in the top 20 SHAP values. On the other hand it is not that odd, since the fascore components should be related to each other, since they form a single score.

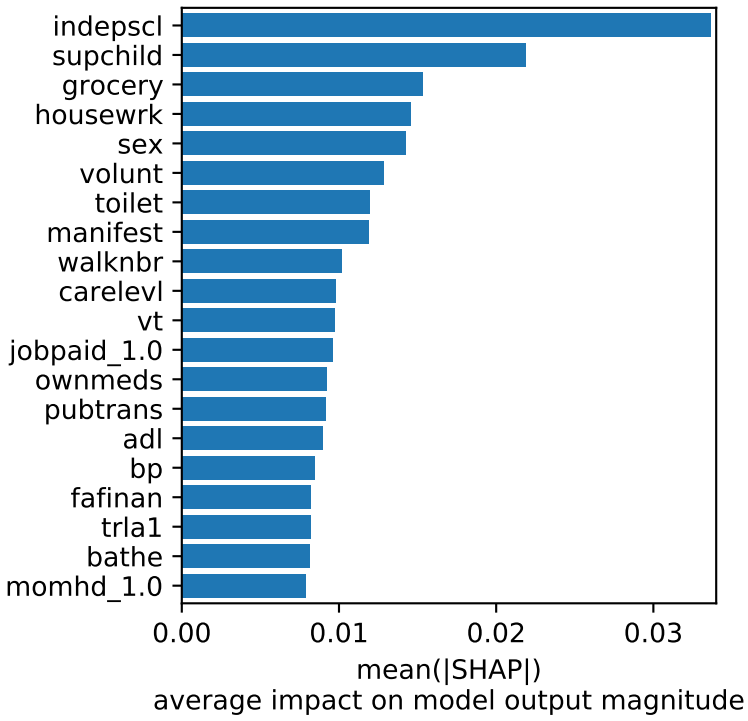


Figure 6.8: *The mean absolute SHAP value over all patients to predict the driving capability.*

6.2 Composite Unified Huntington Disease Rating Scale (cUHDRS) Progression

6.2.1 Model Performance

Next the cUHDRS progression was modelled, here the cUHDRS was predicted for each visit 1 year ahead in time. The distribution of the labels in both the training and test set are shown in figure 6.9. In this figure only four of five time steps are shown, since the predicted time step is one year/time step ahead from the current visit, i.e. only a maximum of five visits. Therefore, there is no label for visit five ($t = 4$). In total around 60% of the training and test data is labeled, since not all participants visited the clinic five years in a row, which is shown in figure 2.4b. The number of unmasked labels, i.e. the labels that are actually trained on is around 49%, since the models are not trained on imputed labels, which is discussed in Assigning Labels and Sample Weights. The shape of the train and test sets are shown in table 6.2. Here the datasets for the RNN models is smaller, since for the other models the samples had to be duplicated for each time step, which is described in Model Input. The total number of samples/patients in this experiment are less than the previously modeled driving capability, see table 6.1, since samples that had only one visit had to be discarded, which is described in Train and Test Split.

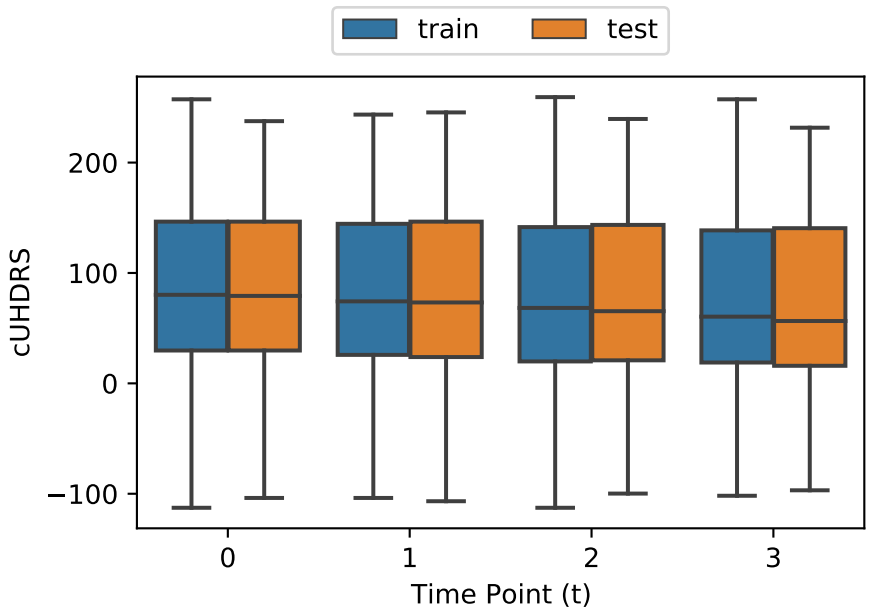


Figure 6.9: *Labels of the train and test set for the cUHDRS.*

Table 6.2: *All the train and test stats of for modelling the cUHDRS, TMS, TFC, SDMT, and SWRT.*

Model	Subset	Shape x	Shape y & sample weights
RNN	Train	(9861, 4, 191)	(9861, 4, 1)
RNN	Test	(2415, 4, 191)	(2415, 4, 1)
Other	Train	(39444, 4, 191)	(39444, 1, 1)
Other	Test	(9660, 4, 191)	(9660, 1, 1)

All trained models were evaluated on four performance measures, namely the MAE, RMSE, Maximum AE, and the R^2 score. All the models and their evaluation metrics are shown in appendix G. To define the best model the average metrics, the metrics per time step, and the metrics per status group are shown. Here the best model can be deducted by evaluating which model performed best at each evaluation.

In figure 6.10 the average over all visits of all the metrics are shown. This figure shows that the LSTM, GRU, and CNN have similar and the best performance in terms of MAE, RMSE, and R^2, however the LSTM has the lowest Max AE.

When looking at the temporal results of the models, see figure 6.11, all the neural network models generally increase in performance over time and the machine learning models decrease in performance. Even the baseline/constant model outperforms the Random Forest and SVM model in terms of MAE, RMSE, and R^2 at the last time point. Something else to note is that the CNN outperforms both the LSTM and GRU at 2 and 4 years into the future in all metrics except the max MAE.

Lastly, the performance per progression group is shown in figure 6.12. Here the groups are defined by the change in the cUHDRS score between the first visit and last visit. The different groups are defined as follows, the stable group includes patients which had no change in the cUHDRS between the first and last visit, the recovered group includes patients which had an increase in the cUHDRS between the first and last visit, and lastly the progressive group includes patients which had a decrease in the cUHDRS between the first and last visit. Figure 6.12 shows that the stable group (n=32) and recovered group (n=506) are generally best predicted by the constant model, however it is the worst model for the progressive group (n=1406).

Considering that the CNN, LSTM, GRU generally had a similar performance, it was decided to look deeper into the performance of the GRU model, since it is more suitable for temporal data than the CNN in this case and it performed similar or better than the LSTM.

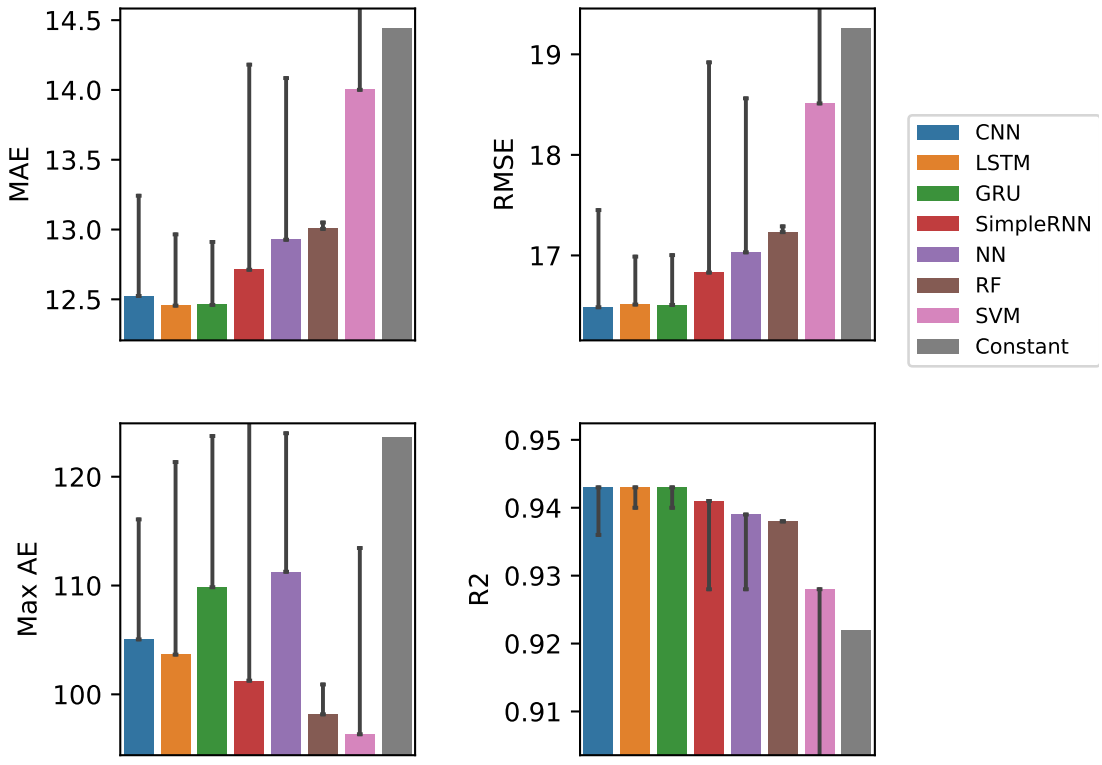


Figure 6.10: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for cUHDRS. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

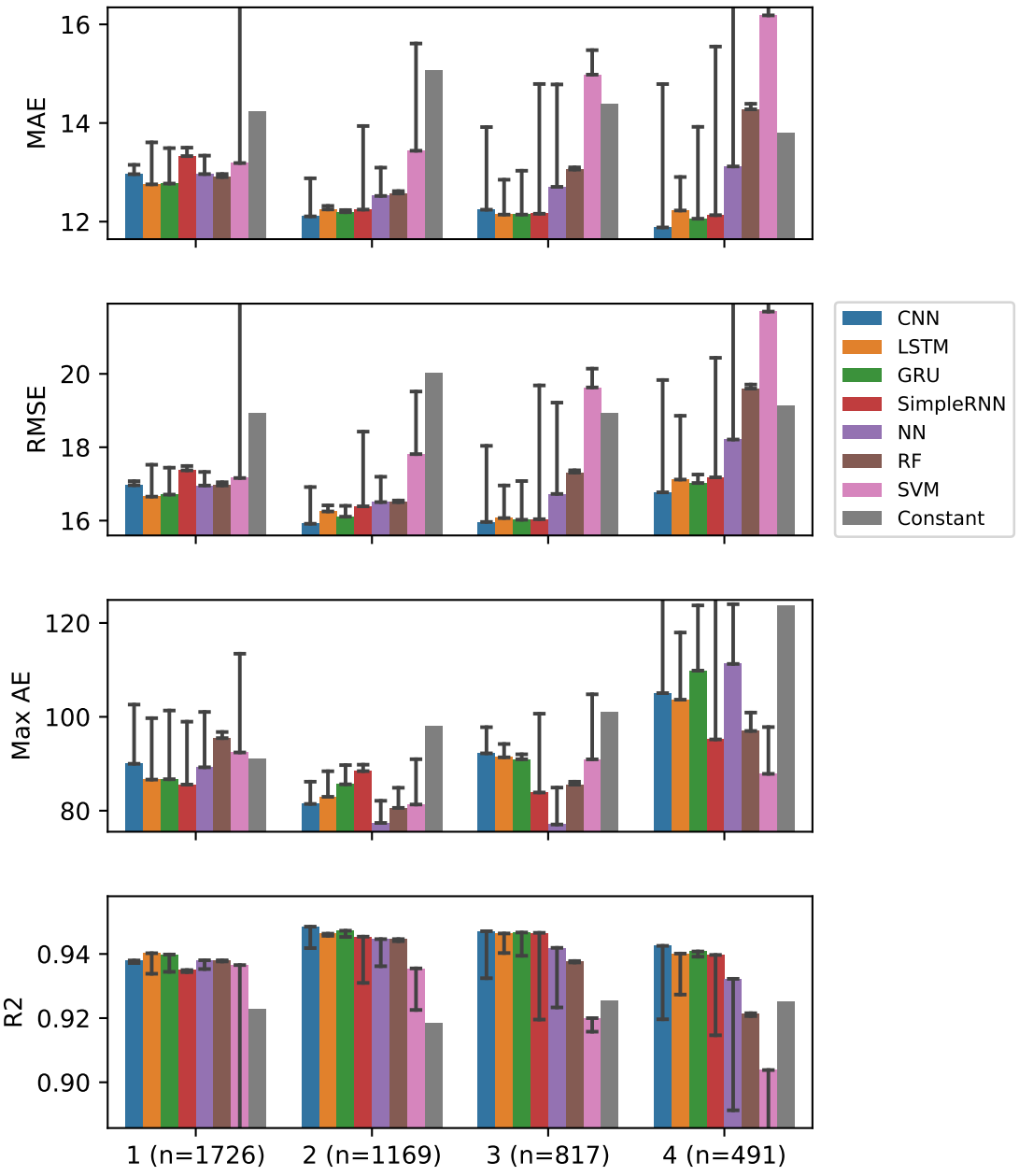


Figure 6.11: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for cUHDRS over all the time points. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

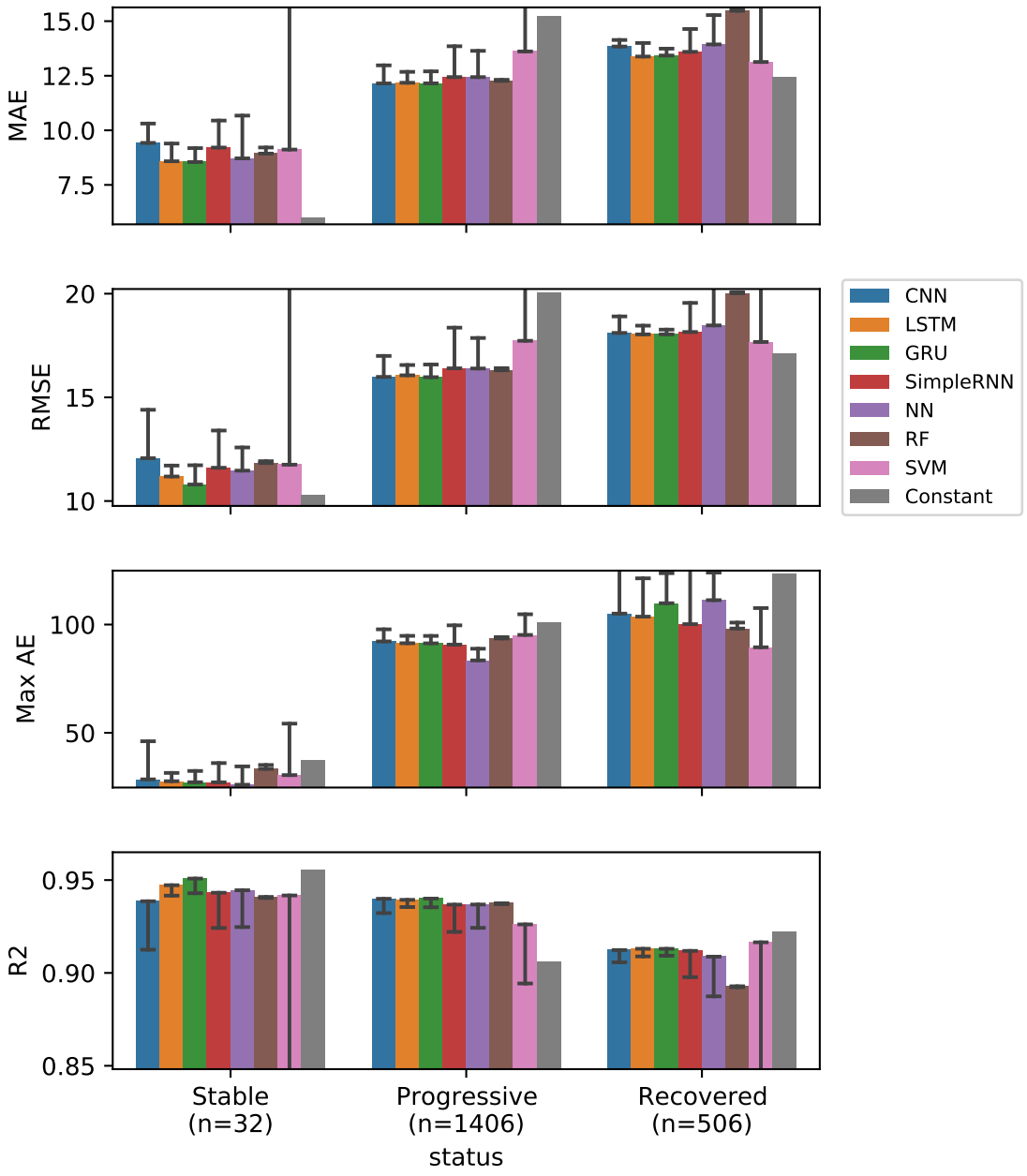


Figure 6.12: *The MAE, RMSE, Max AE, and the R^2 score of the tuned models for cUHDRS over all the status groups consisting of stable (no change in cUHDRS between first and last visit), progressive (decrease in cUHDRS between first and last visit), and recovered (increase in cUHDRS between first and last visit). Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

6.2.2 Model Interpretation

In this section we will further analyze the GRU. First the performance of the GRU is further analyzed and the SHAP algorithm is again applied to find the most important variable in predicting the cUHDRS.

In figure 6.13 the distribution of the absolute error per time step/year ahead on the test set is shown. This figure shows that a lot of predictions have a low absolute error (<5) and that only some predictions have a high absolute error (>35). This suggests that the model has a low error margins for a lot of samples and a higher error margin for some samples. For now it is unknown why some samples have a higher error margin. This can be due to noise in the data or there are not enough training samples for some types of patients or some essential data variables are missing. The figure also shows that the error in predictions is very similar for each year ahead. Suggesting that the model could also be used on cross-sectional data. However, the MAE is slightly higher on the first year than the following years, see the GRU in figure 6.11.

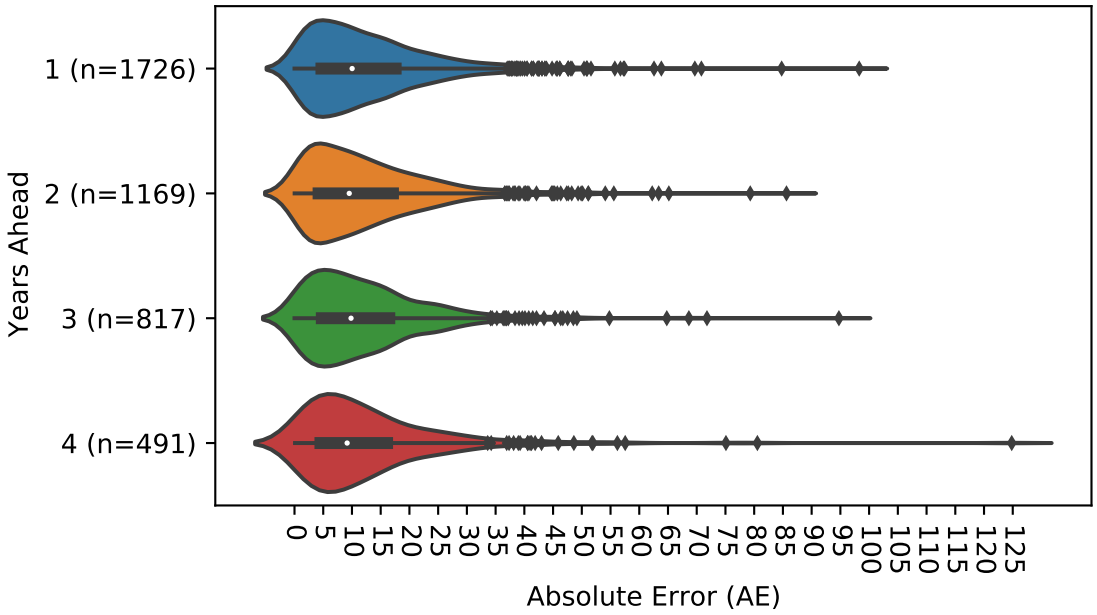


Figure 6.13: *The absolute error distribution of the test set for all predictions/years ahead. This figure shows that a lot of predictions have a very low error margin (<10) and that some predictions have a very high error margin (>35).*

In figure 6.14 the top 20 highest mean absolute SHAP value of all patients in the test set for each variable is shown. The SHAP values represent the importance of each variable to predict the output variable. This figure shows that the SWRT has a slightly higher mean absolute SHAP value than the cUHDRS to predict the cUHDRS in the next

year. Also, the sdmt1 (SDMT) and cognitive score (cogscore) variables have a high SHAP value. The cUHDRs and cogscore variables might have a high SHAP value, since they include the SWRT.

Not all components of the cUHDRS (TMS, TFC, SDMT, SWRT) are in the top 20 highest SHAP values, namely the TMS is missing. Only one component of the TMS is present, the tandem walking (tandem) variable. This might suggest that the progression of cUHDRS is not so much influenced by the TMS and that it is more influenced by cognitive abilities, measured by SWRT, SDMT, cognitive score, and Stroop Colour Naming Test (SCNT).

In figure 6.16 the SHAP values of the diagnosic confidence for motor abnormalities (diagconf) and the TMS score (motscore) variables are shown. This figure also includes all components that make up the TMS score (motscore), which are shown in appendix A. This figure shows that some components making up the TMS have a higher SHAP value than the TMS itself, e.g. gait, tandem walking (tandem), and dysarthria (dysarth). Also, some components have a lower SHAP value than the TMS, e.g. retropulsion pull (retropls), rigidity-arms right (rigarmr), and maximal chorea face (chorface). Suggesting that some TMS components are more important in predicting the progression of the cUHDRS than other TMS components or the sum of those components, i.e. the TMS.

In figure 6.15 the SHAP values of the TFC (tfcscore) and all the TFC components to predict the cUHDRS are shown. This figure shows that the TFC itself has the highest SHAP value than any TFC component. However, some TFC components have a higher SHAP value than others, e.g. finances has the highest SHAP value (0.4) and the Care Level (carelevl) has the lowest SHAP value (0.1). Again, this might suggest that some TFC components are more important in predicting the progression of the cUHDRS than other TFC components. However the TFC score (the sum of the TFC components) seems to be more important than any other component.

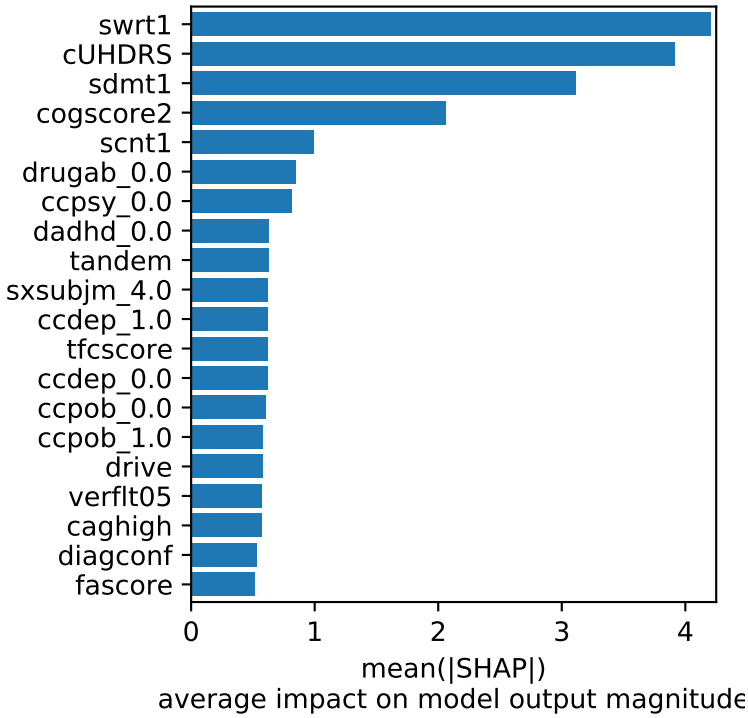


Figure 6.14: *The mean absolute SHAP value over all patients to predict the cUHDRS.*

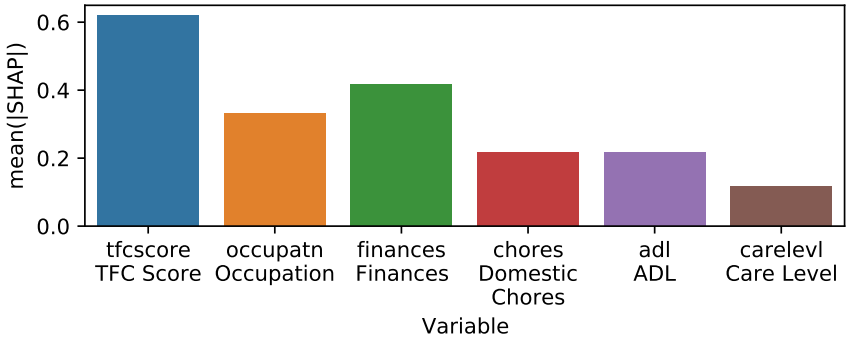


Figure 6.15: *The mean absolute SHAP value of all the components of the TFC. Here it is shown that some components have a higher SHAP value than other components to predict the cUHDRS in the next year.*

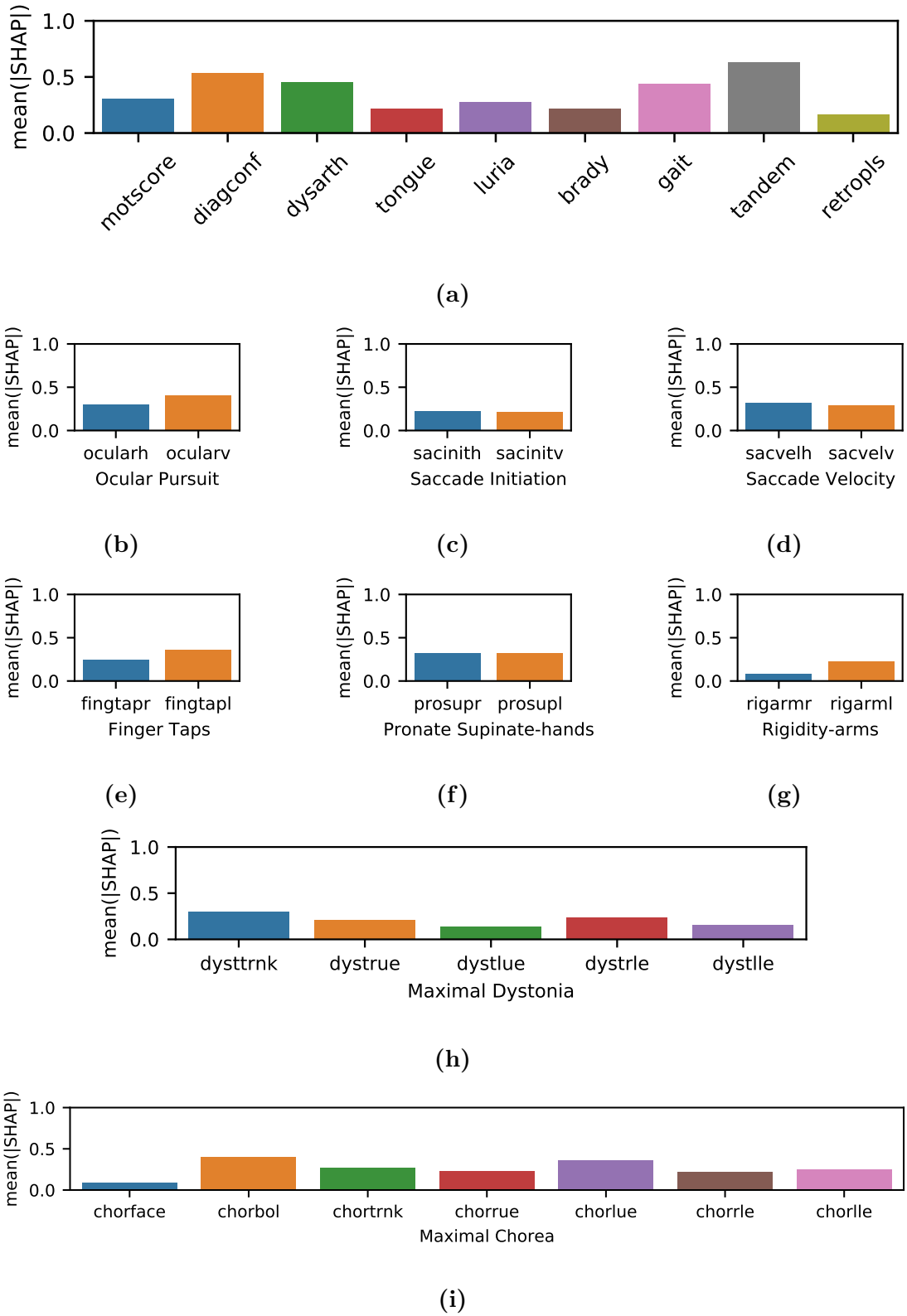


Figure 6.16: *The mean absolute SHAP values of all the TMS components.*

6.3 Total Motor Score (TMS) Progression

6.3.1 Model Performance

In this section the results of the TMS model is shown, which is a component of the cUHDRS. Here we want to find the best model to predict the TMS in the next years and to deduct which variables affect TMS progression and relate that to cUHDRS. Here the same train and test subsets are used, see table 6.2. The distribution of the TMS labels in the train and test set are shown in figure 6.17. In total around 60% of the training and test data is labelled. The number of unmasked labels, i.e. the labels that are actually trained on is around 49%.

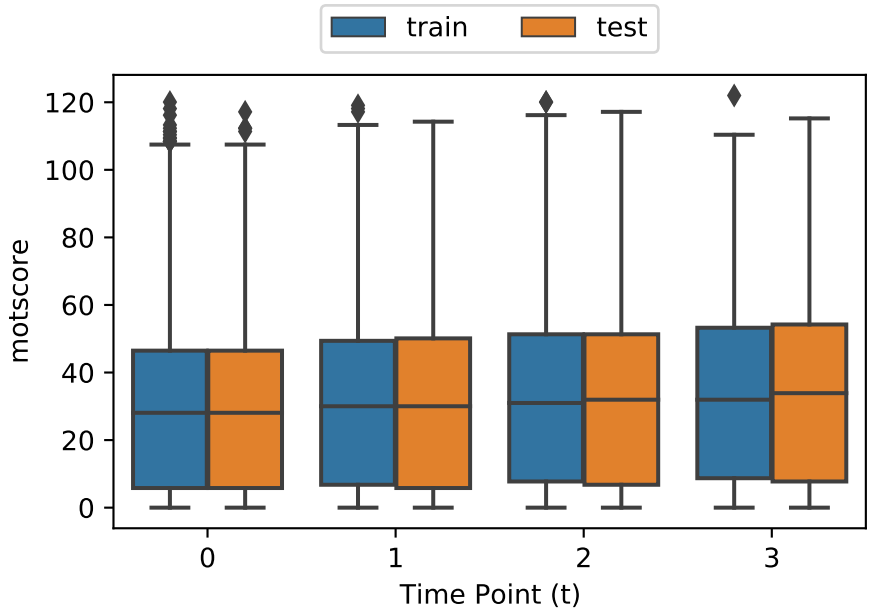


Figure 6.17: *Labels of the train and test set for the TMS.*

All models were evaluated on four performance measures, namely the MAE, RMSE, Maximum AE, and the R^2score. All the models and their evaluation metrics are shown in appendix H. All the models are also evaluated to define which model works best for the TMS. Below the average over all visits of all the metrics are shown in figure 6.18. This figure shows that the LSTM, GRU, and CNN perform similarly, however the CNN has the best performance overall.

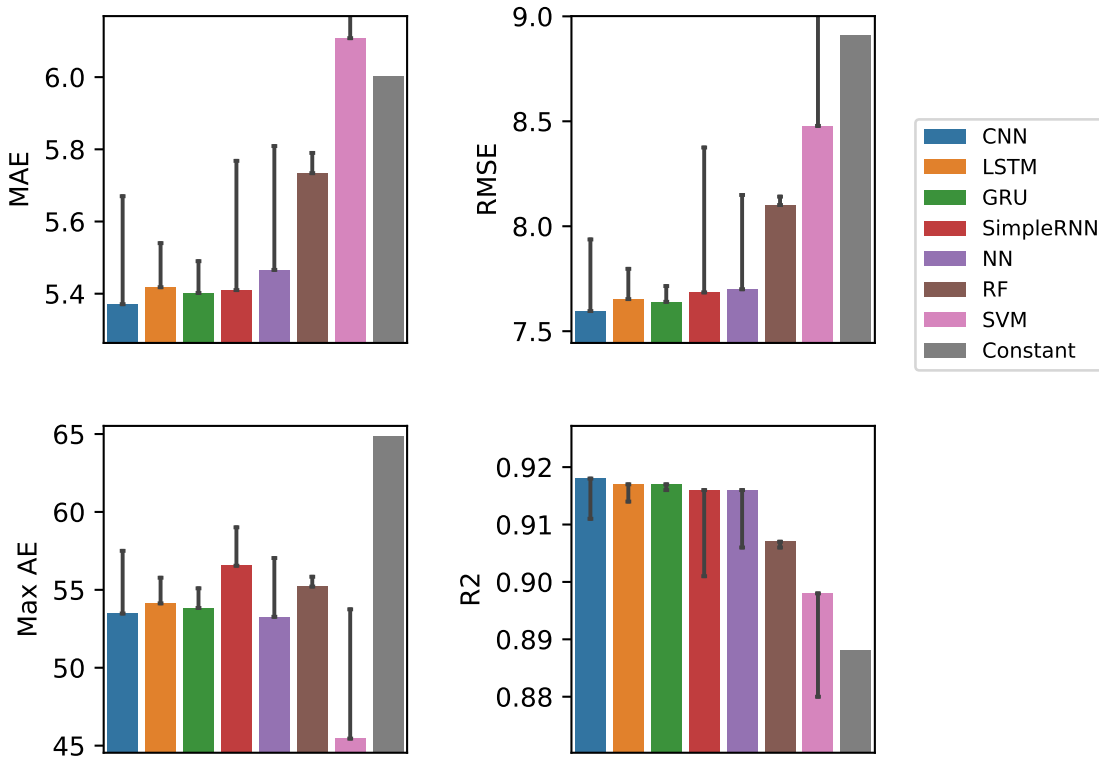


Figure 6.18: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for TMS. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

When looking at the performance for predicting each year ahead in time, see figure 6.19, it is shown that the CNN has the best or second best performance of all models, except for the fourth year. It is also shown that the RF and SVM decrease in performance for each year further ahead.

In figure 6.20 the performance per status group is shown. This figure shows that the constant model is the best for the recovered (n=470) and stable (n=272) group, however it is the worst model for the progressive (n=1354) group. In the progressive group it is not clear which model is the best, since all neural network models and the Random Forest model have a similar performance.

Considering the CNN was overall slightly better overall, see figure 6.18, it was decided to further investigate the model.

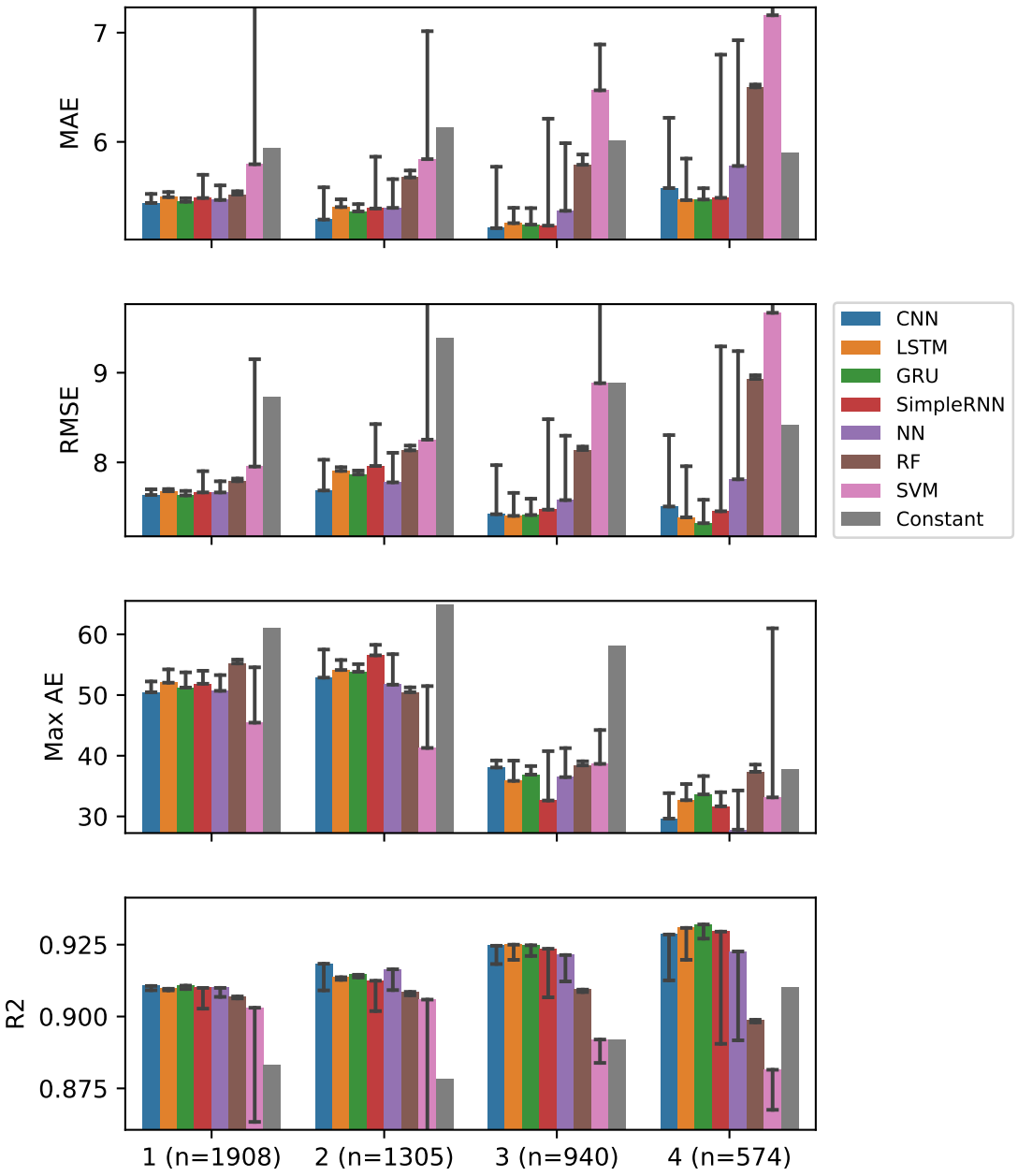


Figure 6.19: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for TMS over all the time points. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

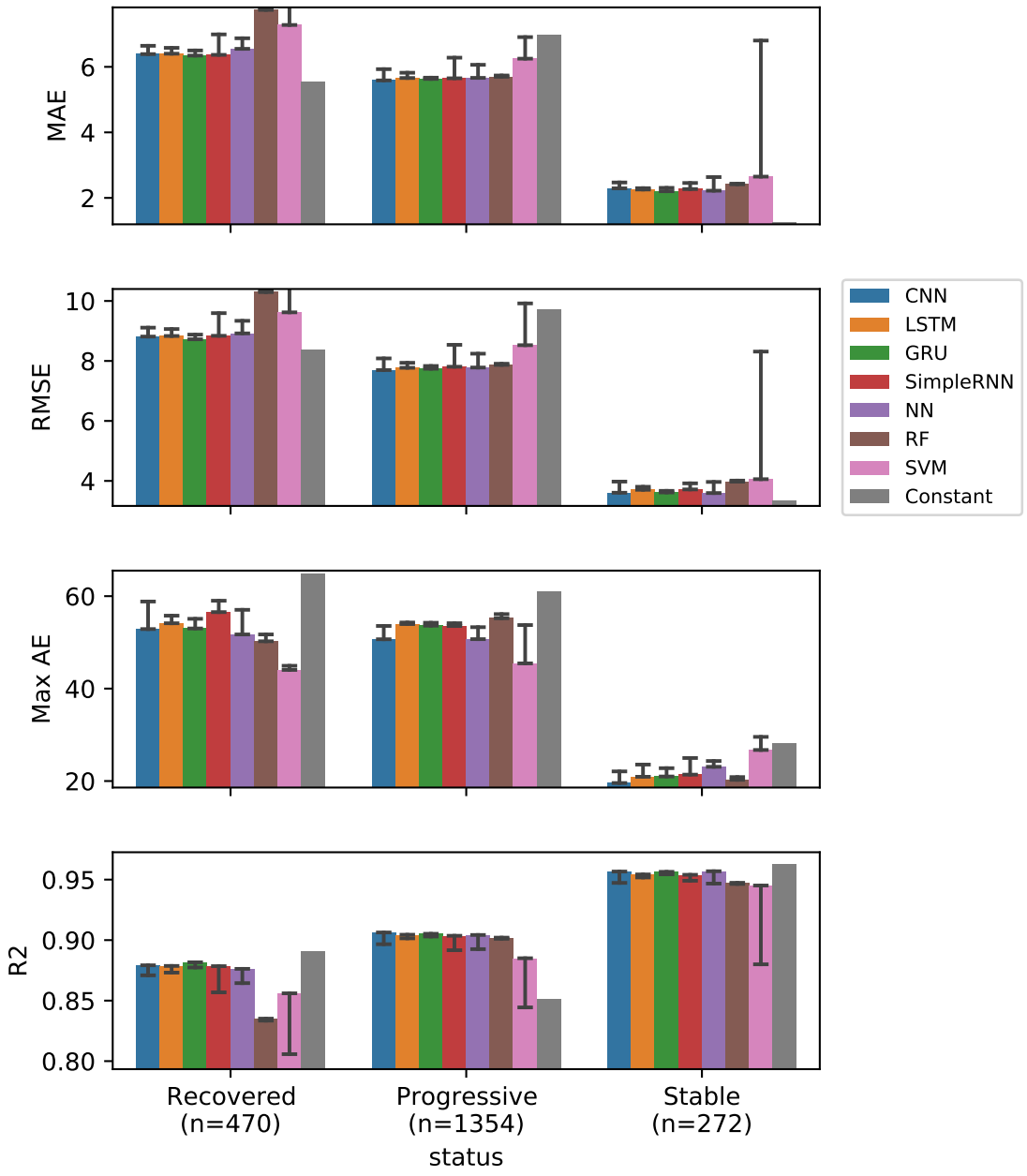


Figure 6.20: *The MAE, RMSE, Max AE, and the R^2 score of the tuned models for TMS over all the status groups consisting of stable (no change in cUHDRS between first and last visit), progressive (increase in TMS between first and last visit), and recovered (decrease in TMS between first and last visit). Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

6.3.2 Model Interpretation

For the TMS the CNN was further analyzed by using the SHAP algorithm. In figure 6.21 the top 20 highest mean absolute SHAP value of all patients in the test set for each variable is shown. This figure shows that the diagnostic confidence of motor abnormalities (diagconf) and tandem walking (tandem) variable have the highest mean absolute SHAP value to predict the TMS in the next year. Many of the components of the TMS are found in the top 20 highest mean absolute SHAP values, which is to be expected. However the figure also shows that some components have a higher SHAP value than others e.g. tandem has the highest SHAP in the TMS.

In figure 6.22 the SHAP values of the TMS (motscore), diagnostic confidence of motor abnormalities (diagconf), and all the TMS components are shown. Here some TMS components are grouped by a specific group component, e.g. the finger taps includes the variables finger tap right hand (fingtapr) and the variable finger tap left hand (fingtapl), see figure 6.22e. The saccade initiation, saccade velocity, and rigidity-arms groups have similar SHAP values between the components.

When looking at all the components it is clear that the tandem walking (tandem) variable is the most important TMS component that makes up the TMS score (motscore). This does not include the diagnostic confidence of motor abnormalities (diagconf), because it is not used in creating the TMS score, see appendix A. All the other TMS components are near a SHAP value of 0.2 or below. Here the lowest SHAP values is achieved by the retropulsion pull (retropls) variable. This suggest that some components have a bigger impact on the progression of the TMS than other components.

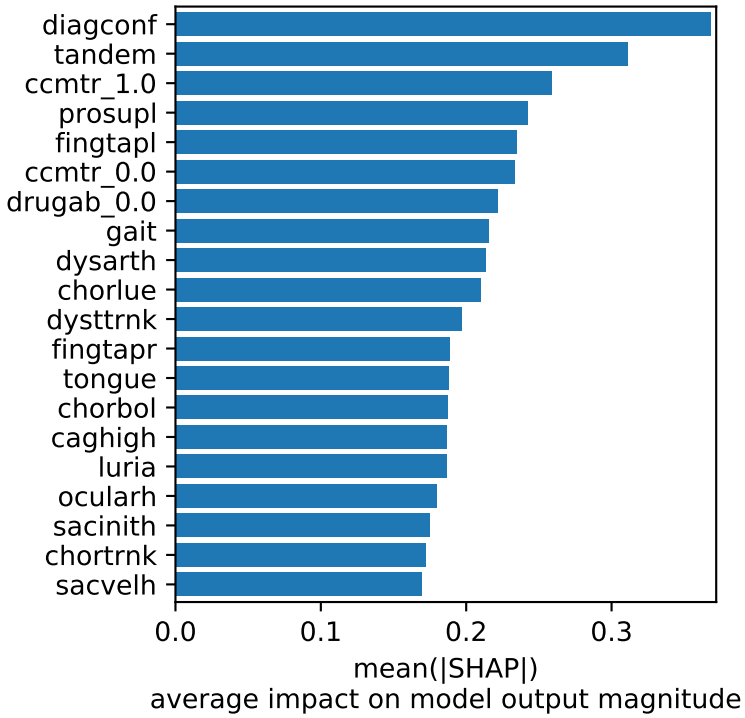


Figure 6.21: *The mean absolute SHAP value over all patients to predict the TMS.*

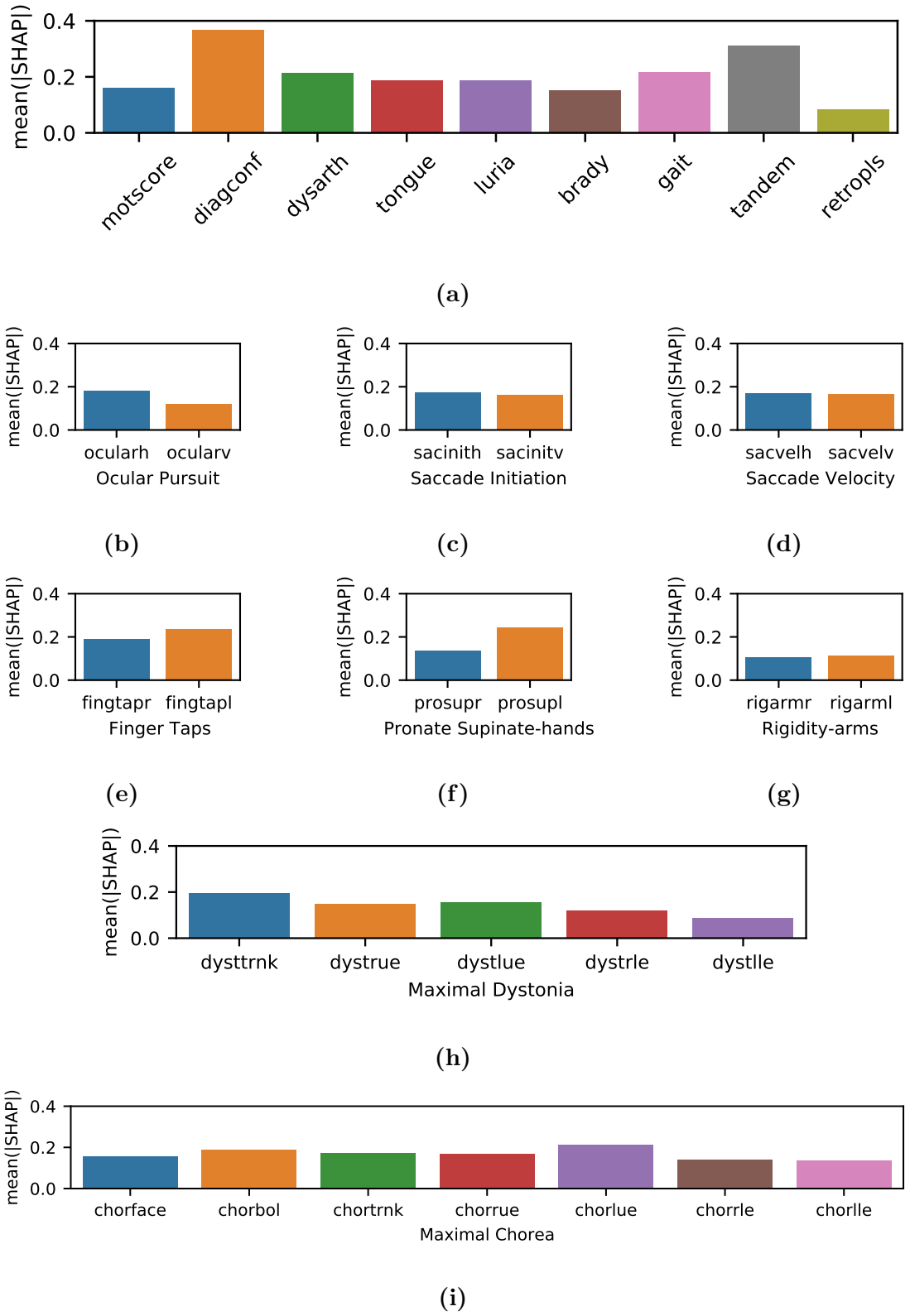


Figure 6.22: *The mean absolute SHAP values of all the TMS components.*

6.4 Total Functional Capacity (TFC) Progression

6.4.1 Model Performance

To show what affects the components of the cUHDRS score the TFC also needs to be modelled similarly as the cUHDRS. Here the same train and test subsets are used, see table 6.2. The distribution of the TFC labels in the train and test set are shown in figure 6.23. In total around 60% of the training and test data is labelled. The number of unmasked labels, i.e. the labels that are actually trained on is around 49%.

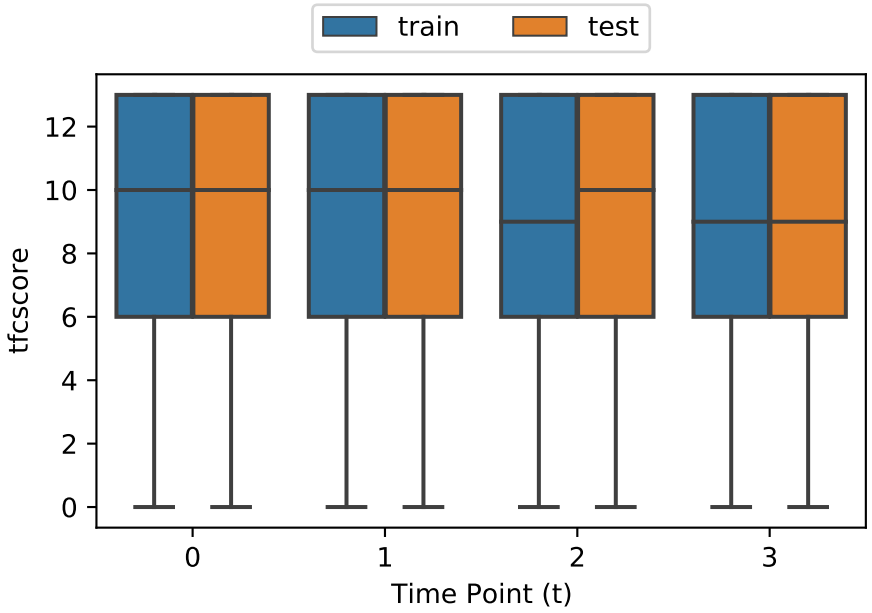


Figure 6.23: *Labels of the train and test set for the TFC.*

All models were evaluated on four performance measures, namely the MAE, RMSE, Maximum AE, and the R^2 score. All the models and their evaluation metrics are shown in appendix I. All the models are also evaluated to define which model works best for the TFC. Below the average over all visits of all the metrics are shown in figure 6.24. This figure shows that the GRU and CNN perform similarly, with CNN having the lowest MAE, GRU the lowest RMSE and highest R^2, however the difference is negligible.

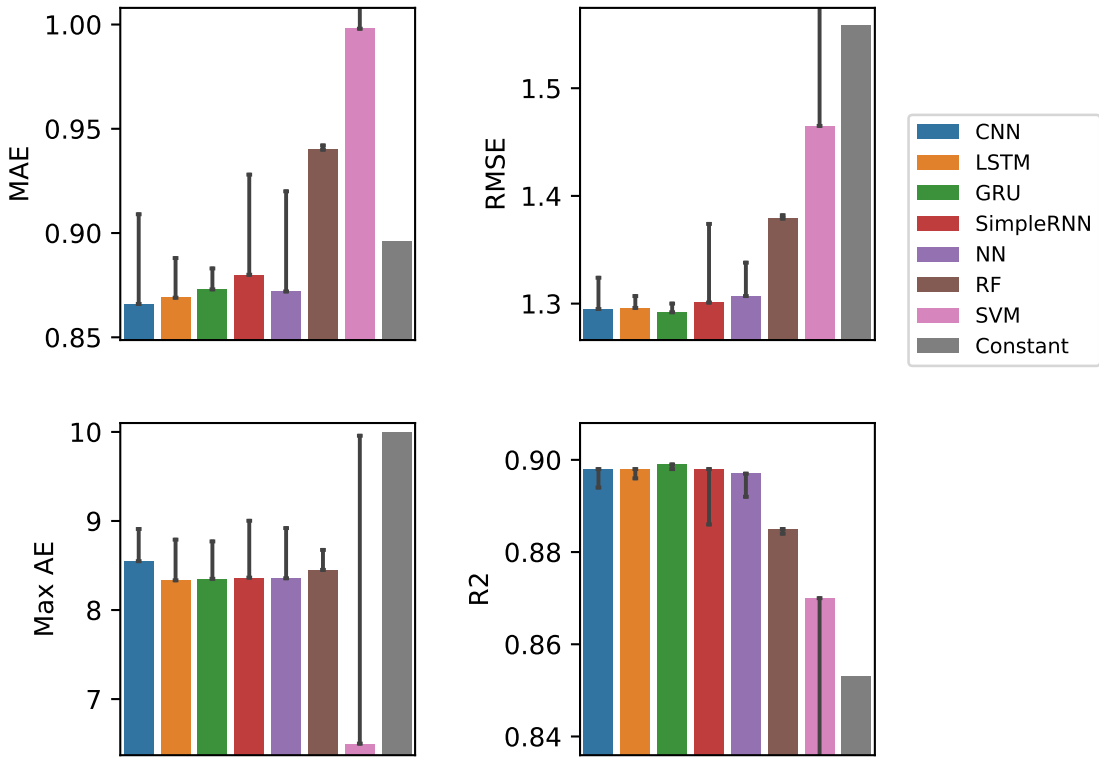


Figure 6.24: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for TFC. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

When looking at the performance for predicting each year ahead in time, see figure 6.25, it is shown that again CNN and GRU have similar performance and that the differences between them are very small. It is also shown that the RF and SVM model again perform worse when predicting further ahead in time. It is also shown that the baseline/constant model has a similar and sometimes the best performance in terms of MAE, however it has one of the worst performance in terms of the other metrics.

In figure 6.26 the performance per status group is shown. This figure shows that the constant model is the best for the recovered and stable group in terms of MAE and RMSE, however it is the worst model for the progressive group. In the progressive group it is not clear which model is the best, since all neural network models model have a similar performance.

It was decided to define the best model from figure 6.24. Considering it is not completely clear from the temporal and status figures which model is the best. It was shown that the CNN had the lowest MAE and the GRU had the lowest RMSE. Since the differences between the metrics of the CNN and GRU were not big and GRU works better for time series in this case. Therefore. it was decided to further analyze the GRU model for TFC.

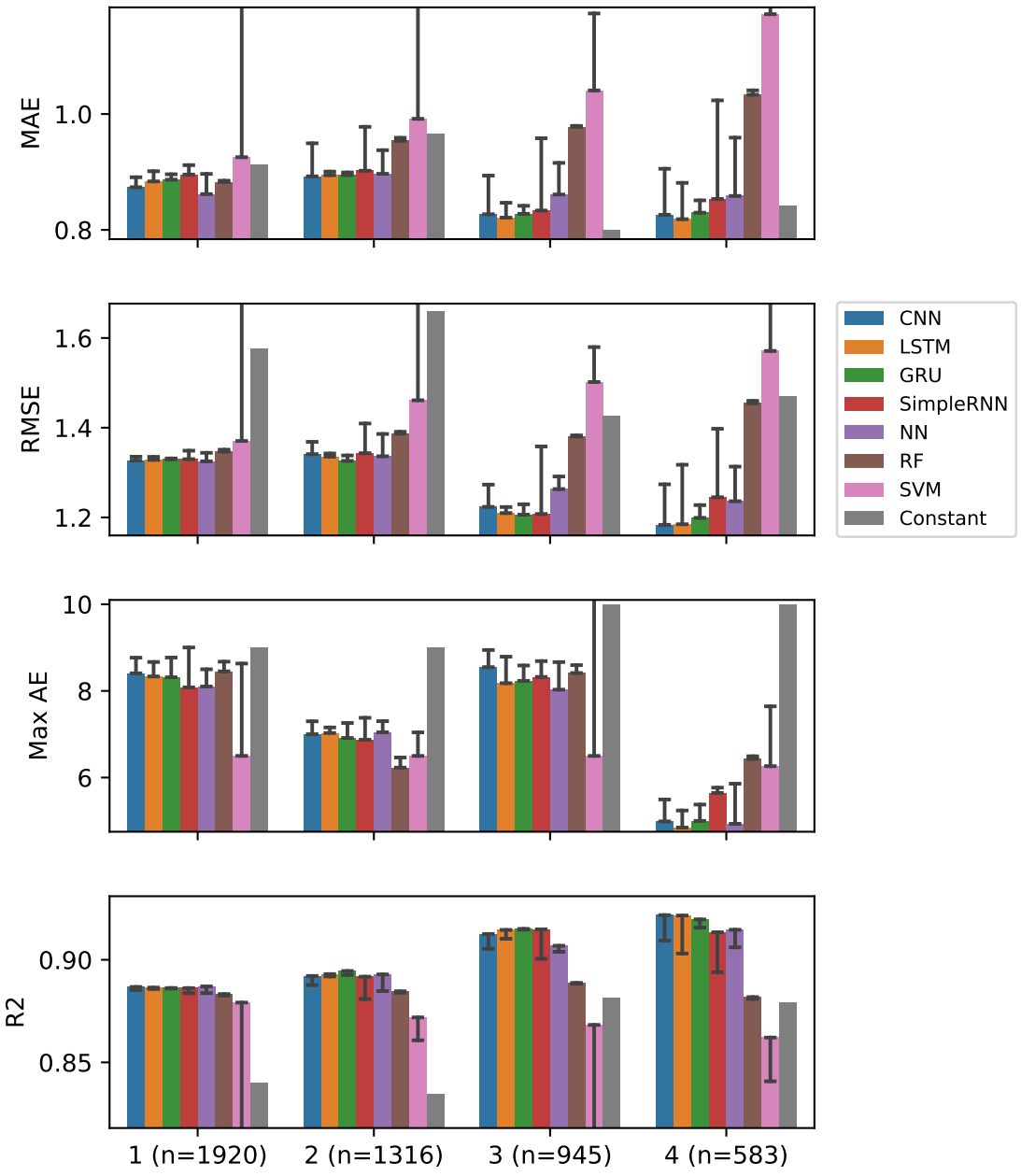


Figure 6.25: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for TFC over all the time points. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

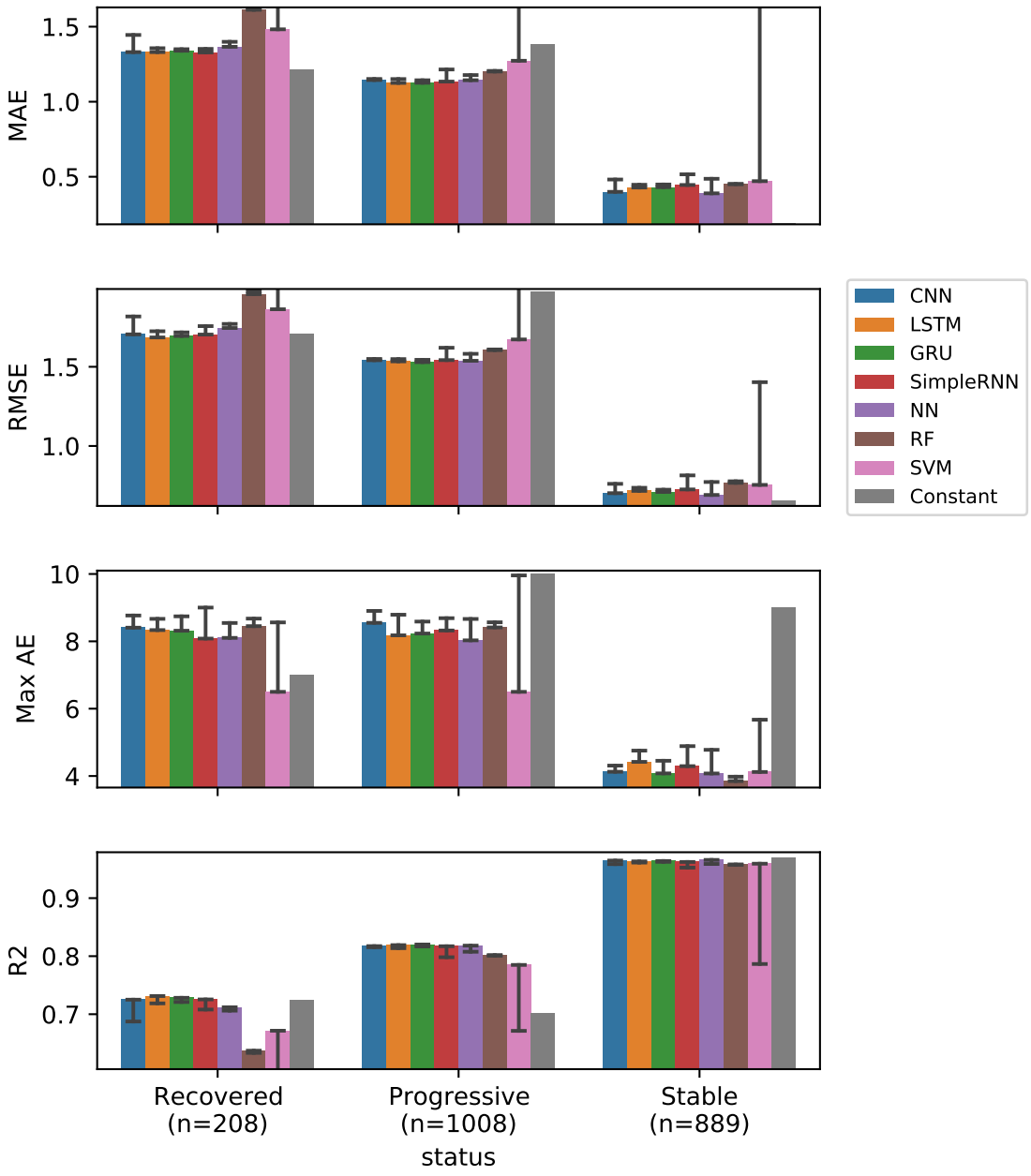


Figure 6.26: *The MAE, RMSE, Max AE, and the R^2 score of the tuned models for TFC over all the status groups consisting of stable (no change in cUHDRS between first and last visit), progressive (increase in TFC between first and last visit), and recovered (decrease in TFC between first and last visit). Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

6.4.2 Model Interpretation

For the TFC the GRU was further analyzed by using the SHAP algorithm. In figure 6.27 the top 20 highest mean absolute SHAP value of all patients in the test set for each variable is shown. In this figure it is shown that the finances variable has the highest mean absolute SHAP value to predict the TFC in the next year. Here the finances variable indicates if the patient is unable, needs major or slight assistance or can independently manage their finances. Also, the tfcscore (TFC) and all the components of the TFC (adl, occupatn, carelevl, chores) have a high SHAP value, which is to be expected. In addition to the TFC components, some components of the Fascore and the Fascore itself also have high SHAP values, e.g. bathe, carehome, pubtrans, drive.

The SHAP values of the TFC components are shown in figure 6.28. This figure shows that some components of TFC have a lower impact than other components, e.g. domestic chores (chores) has the lowest SHAP value of all components. This suggests that some TFC components affect the progression of the TFC score more than other components, e.g. the TFC score will change more if a patient needs more assistance in managing their finances than when a patient needs help with domestic chores (chores). This might also explain why the finances variable is slightly more important than the sum of all the TFC components, i.e. the TFC score.

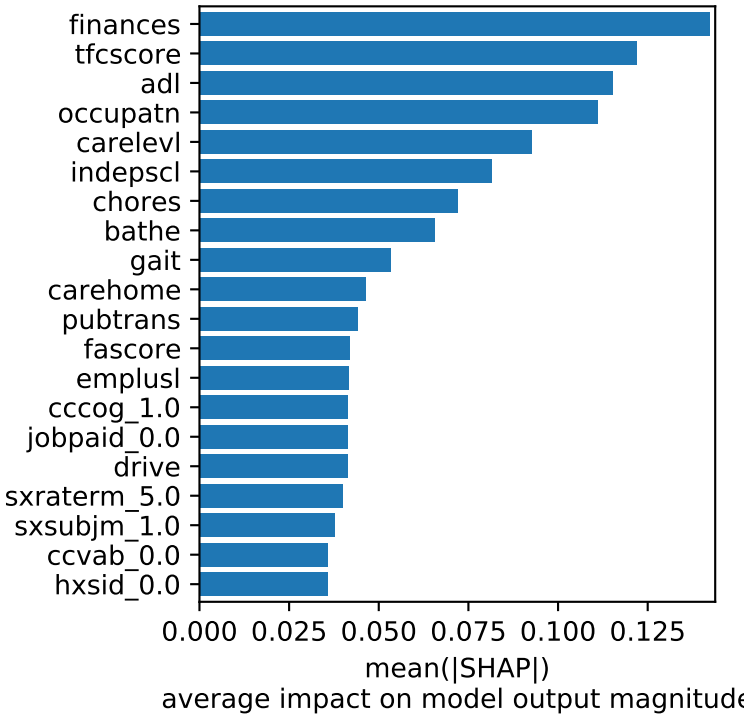


Figure 6.27: *The mean absolute SHAP value over all patients to predict the TFC.*

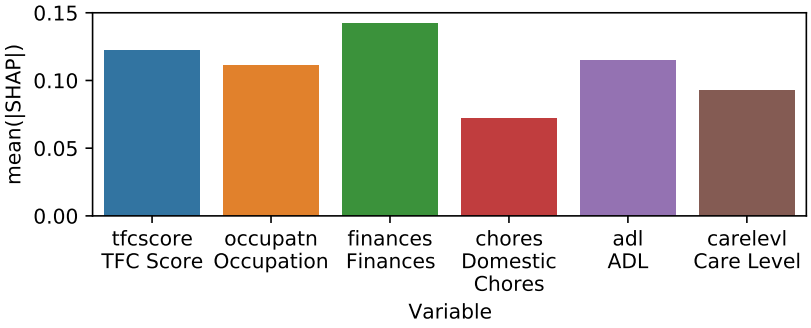


Figure 6.28: *The mean absolute SHAP value of all the components of the TFC. Here it is shown that some components have a higher SHAP value than other components to predict the TFC in the next year.*

6.5 Symbol Digit Modality Test (SDMT) Progression

6.5.1 Model Performance

To show what affects the components of the cUHDRS score the SDMT also needs to be modelled similarly as the cUHDRS. Here the same train and test subsets are used, see table 6.2. The distribution of the SDMT labels in the train and test set are shown in figure 6.29. In total around 60% of the training and test data is labelled. The number of unmasked labels, i.e. the labels that are actually trained on is around 44%.

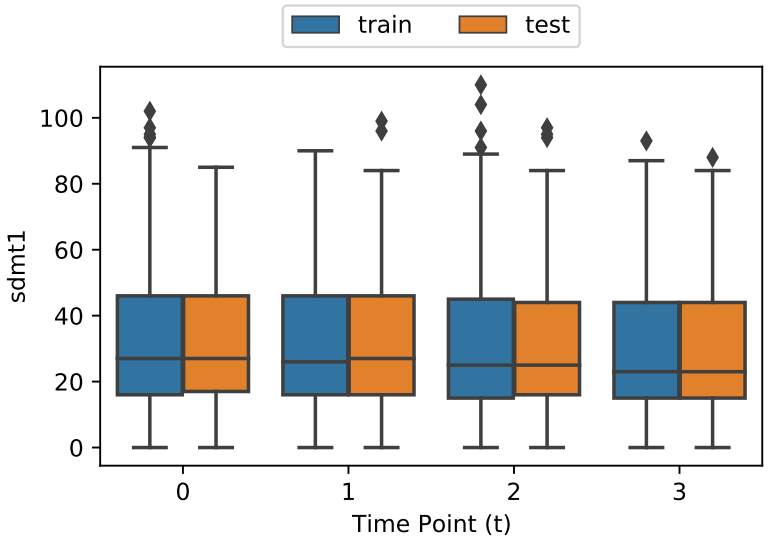


Figure 6.29: *Labels of the train and test set for the SDMT.*

All models were evaluated on four performance measures, namely the MAE, RMSE, Maximum AE, and the R^2 score. All the models and their evaluation metrics are shown in appendix J. All the models are also evaluated to define which model works best for the SDMT. Below the average over all visits of all the metrics are shown in figure 6.30. This figure shows that the GRU has the lowest MAE, GRU the lowest RMSE and highest R^2.

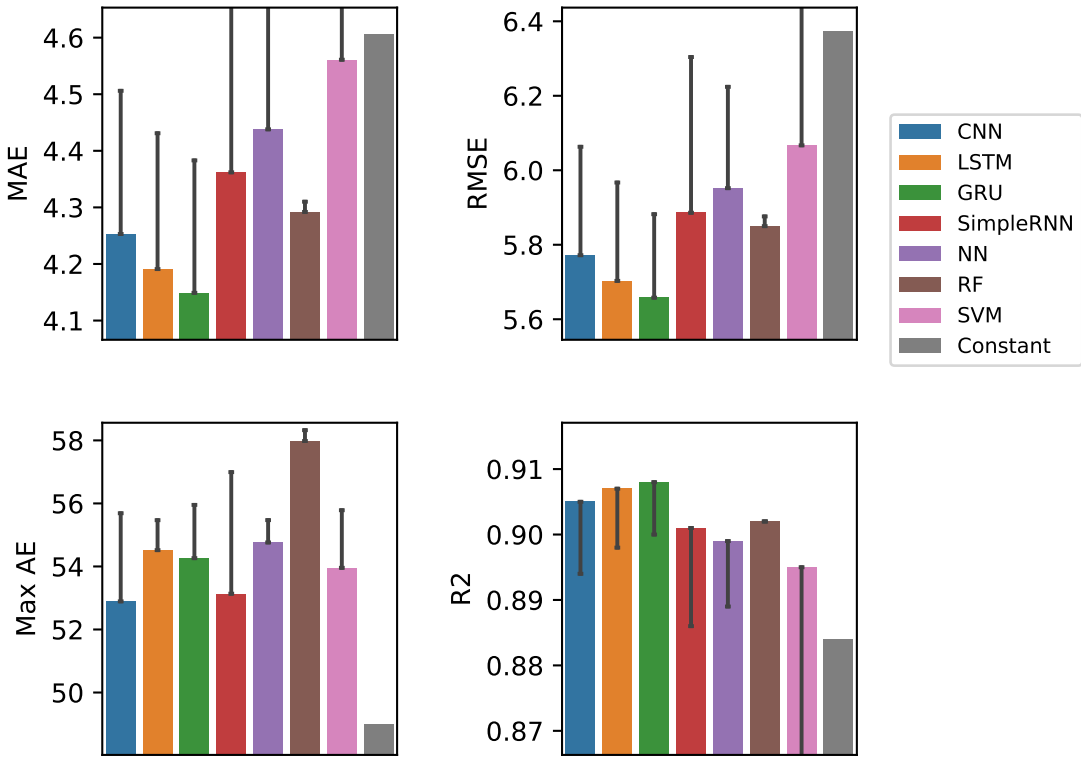


Figure 6.30: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for SDMT. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

When looking at the performance for predicting each year ahead in time, see figure 6.31, it is shown that the GRU is overall performing the best at each time step and metric, excluding the fourth year in terms of RMSE and Max AE.

In figure 6.32 the performance per status group is shown. This figure shows that the constant model is the best for the stable group in terms of MAE and RMSE. The recovered group is best predicted by the GRU in terms of MAE, RMSE, and R^2. It is not clear which model is best for the progressive group, except for the constant model.

It was decided to further analyze the GRU model, considering that the GRU is generally the best model overall, see figure 6.30 and also considering that the GRU is generally the best when looking at the temporal results, see figure 6.31.

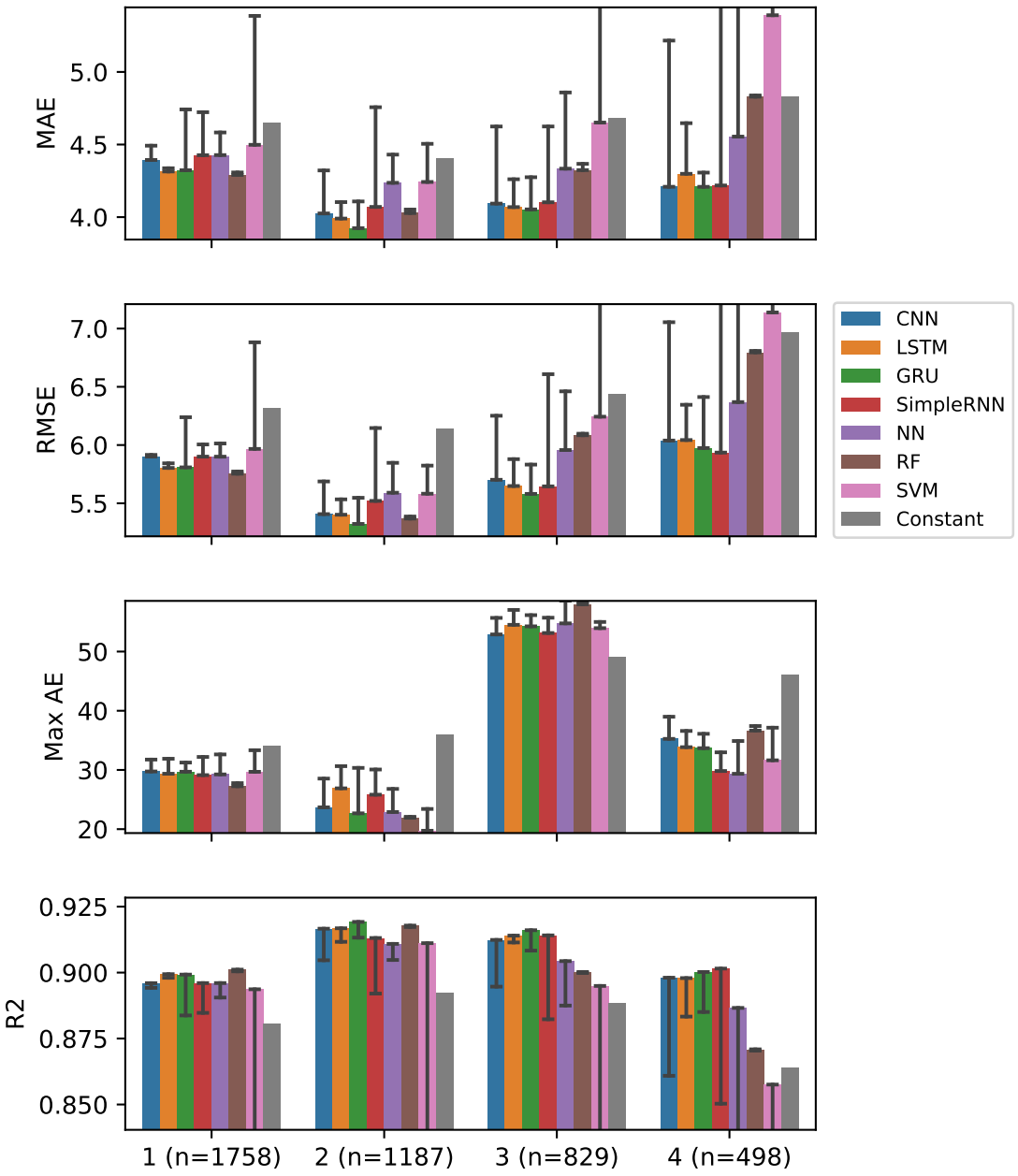


Figure 6.31: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for SDMT over all the time points. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

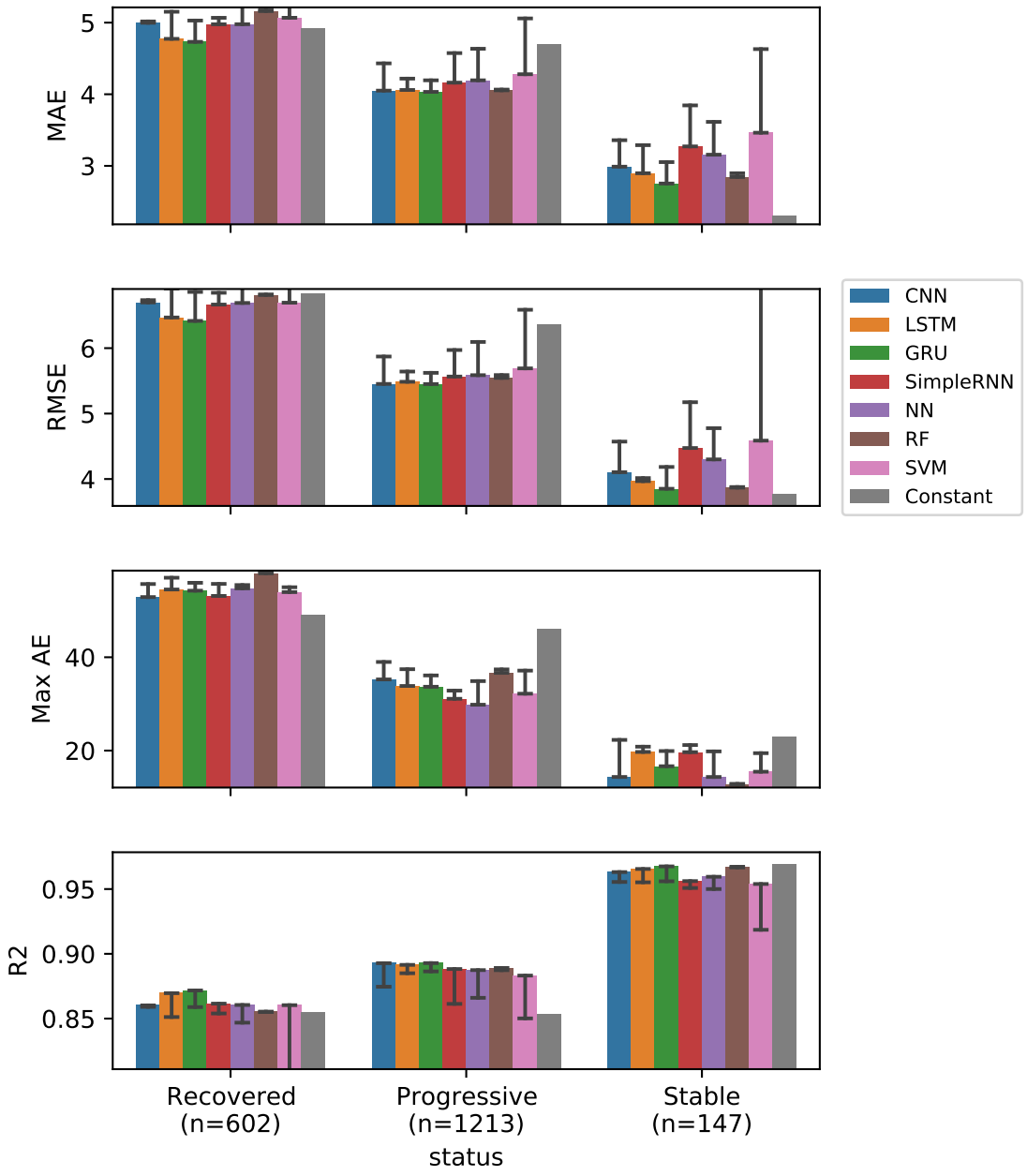


Figure 6.32: *The MAE, RMSE, Max AE, and the R^2 score of the tuned models for SDMT over all the status groups consisting of stable (no change in cUHDRS between first and last visit), progressive (increase in SDMT between first and last visit), and recovered (decrease in SDMT between first and last visit). Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

6.5.2 Model Interpretation

For the SDMT the GRU was further analyzed by using the SHAP algorithm. In figure 6.33 the top 20 highest mean absolute SHAP value of all patients in the test set for each variable is shown. In this figure it is shown that the sdmt1 (SDMT) is the highest mean absolute SHAP value to predict the SDMT in the next year. Also, the cUHDRS, cognitive score (cogscore2), and swrt1 (SWRT) have a high SHAP value. Here the cUHDRS might be important because SDMT is a component of the cUHDRS.

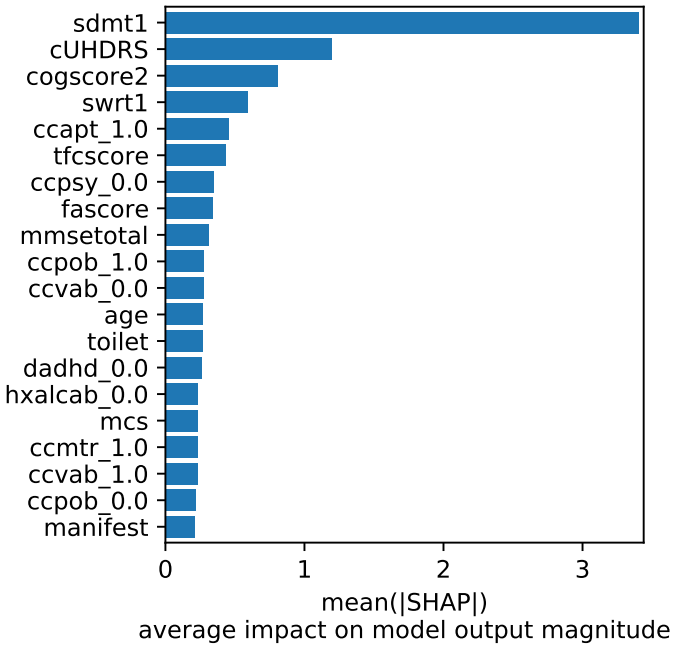


Figure 6.33: *The mean absolute SHAP value over all patients to predict the SDMT.*

6.6 Stroop Word Reading Test (SWRT) progression

6.6.1 Model Performance

To show what affects the components of the cUHDRS score the SWRT also needs to be modelled similarly as the cUHDRS. Here the same train and test subsets are used, see table 6.2. The distribution of the SWRT labels in the train and test set are shown in figure 6.34. In total around 60% of the training and test data is labelled. The number of unmasked labels, i.e. the labels that are actually trained on is around 46%.

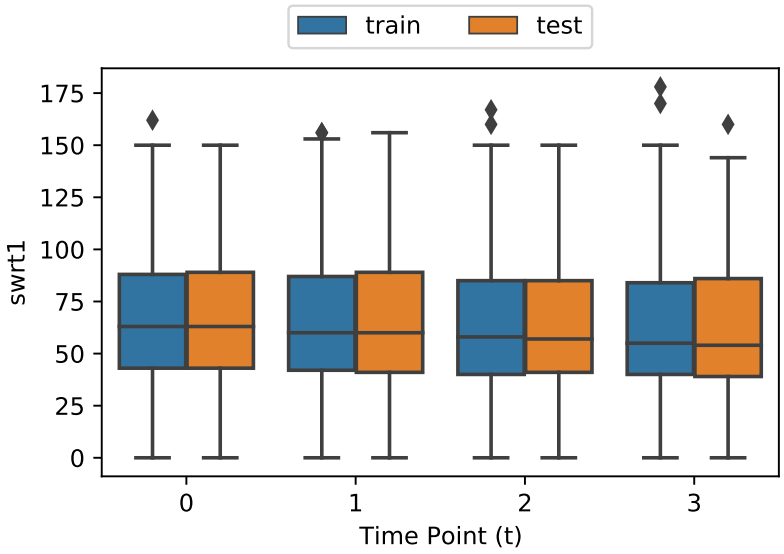


Figure 6.34: *Labels of the train and test set for the SWRT.*

After training all models were evaluated on four performance measures, namely the MAE, RMSE, Maximum AE, and the R^2 score. All the models and their evaluation metrics are shown in appendix K. All the models are also evaluated to define which model works best for the SWRT. Below the average over all visits of all the metrics are shown in figure 6.35. This figure shows that the LSTM, GRU, and CNN perform overall the best and that the GRU has the lowest MAE and that the CNN has the highest R^2, however the differences between the scores of the LSTM, GRU, and CNN are very small.

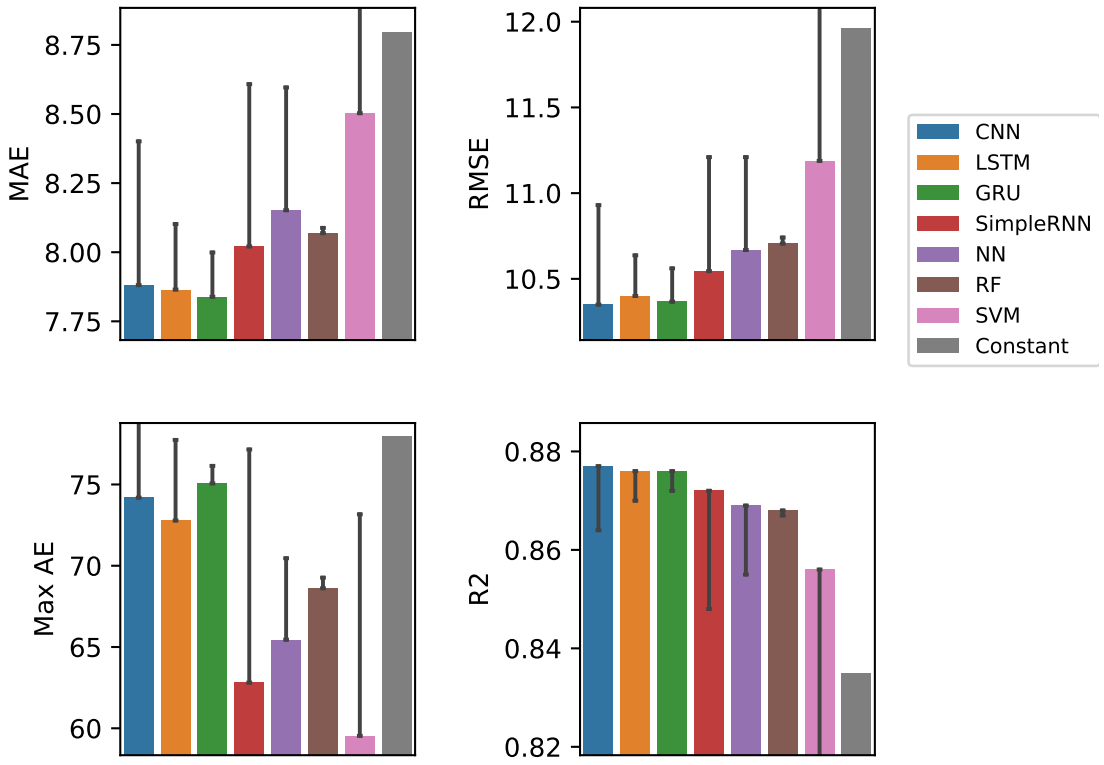


Figure 6.35: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for SWRT. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

When looking at the performance for predicting each year ahead in time, see figure 6.36, it is shown again that the LSTM, GRU, and CNN perform similarly.

In figure 6.37 the performance per status group is shown. This figure shows that the constant model is the best for the stable group in terms of MAE and RMSE. The recovered group is best predicted by the GRU in terms of MAE, RMSE, and R^2, but only slightly. It is not clear which model is best for the progressive group, since all models perform similarly.

Considering it is not completely clear whether the CNN or GRU performed the best according to the previous figures, it was decided to further analyze the GRU, since it is more suited for modelling the data in our case, i.e. it needs less samples to predict the same output.

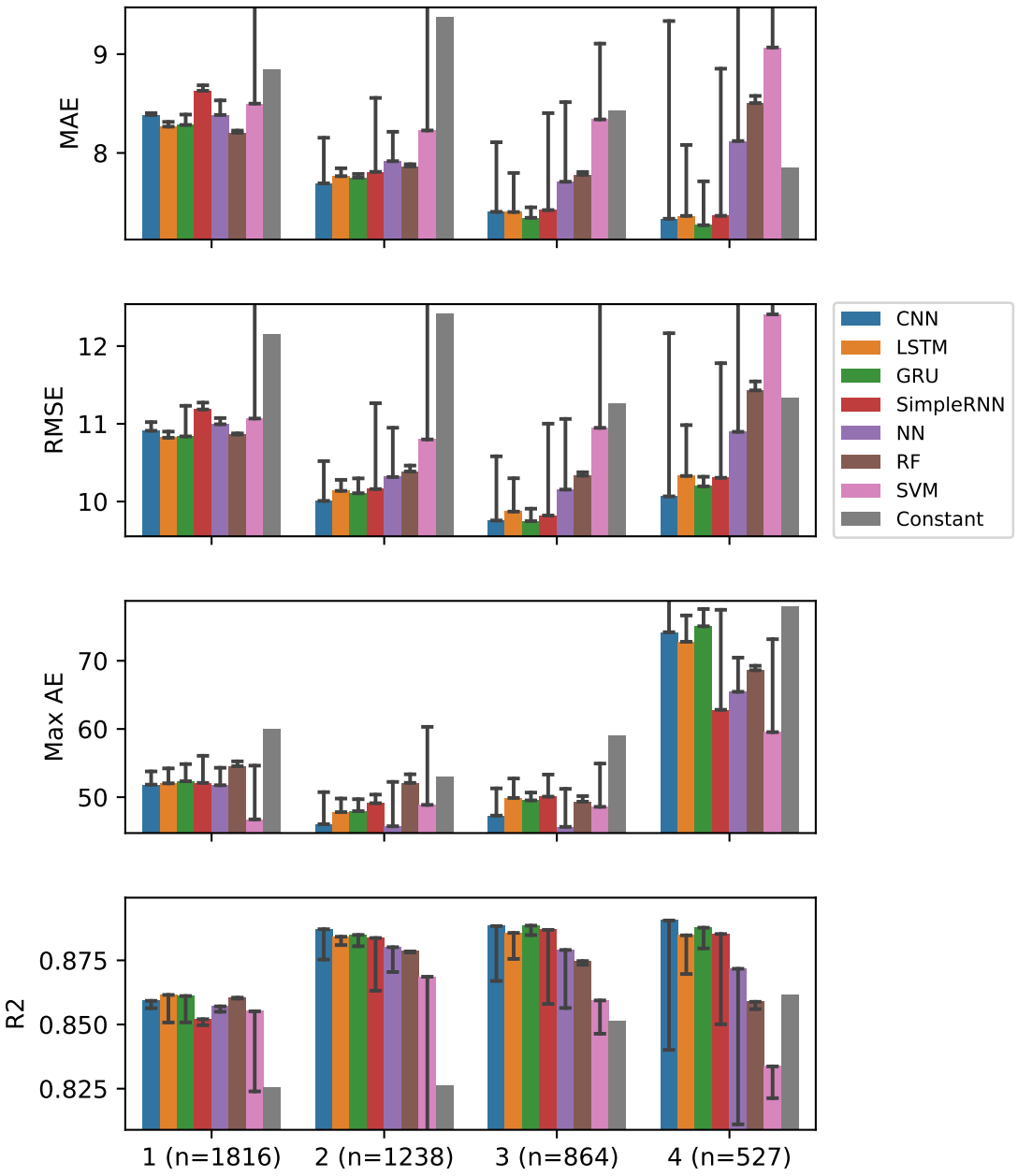


Figure 6.36: *The MAE, RMSE, Max AE, and the R^2 score of all the tuned models for SWRT over all the time points. Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

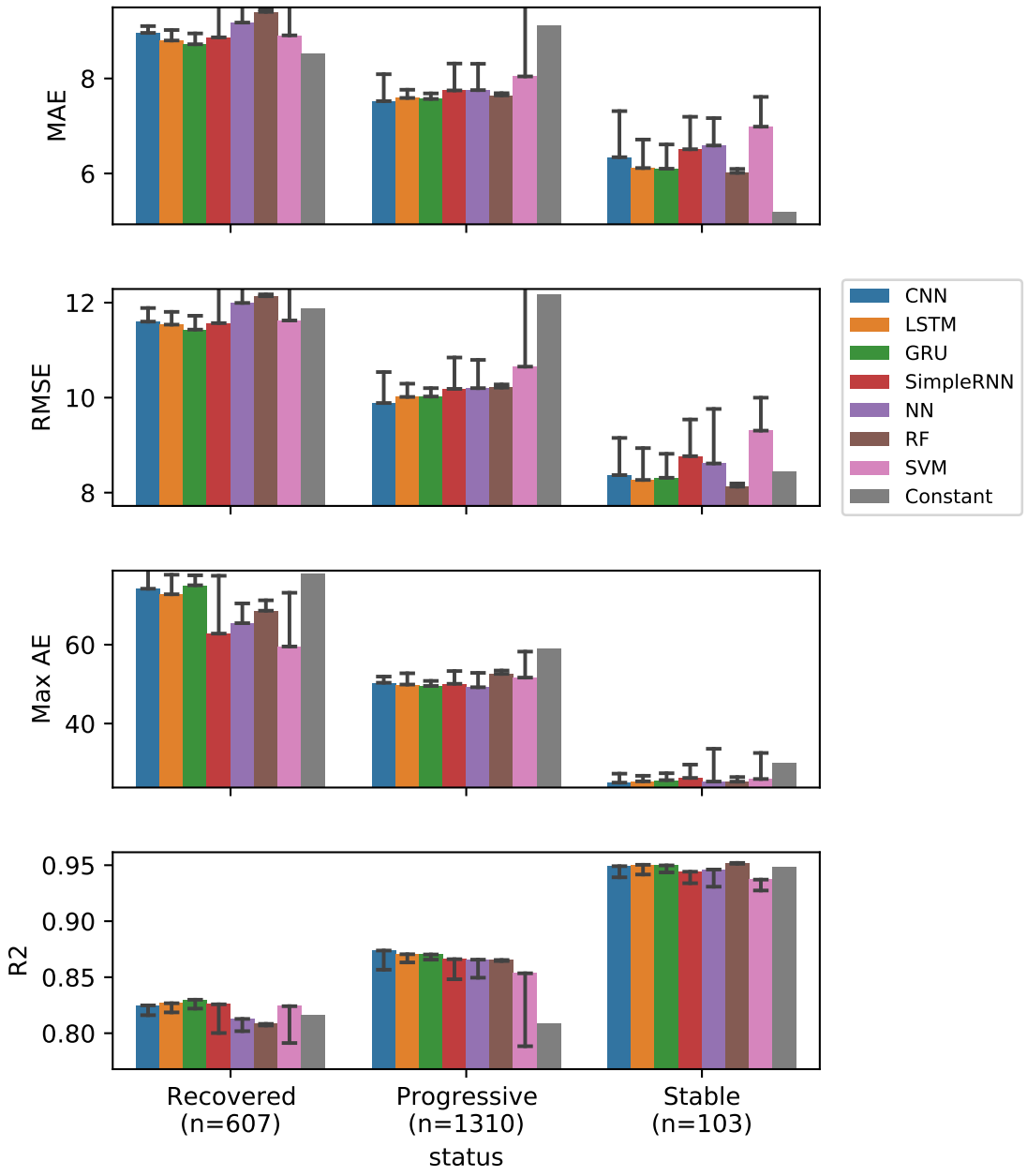


Figure 6.37: *The MAE, RMSE, Max AE, and the R^2 score of the tuned models for SWRT over all the status groups consisting of stable (no change in cUHDRS between first and last visit), progressive (increase in SWRT between first and last visit), and recovered (decrease in SWRT between first and last visit). Here the bars correspond to the best score and the error bars represent the scores of the other tuned models.*

6.6.2 Model Interpretation

For the SWRT the GRU was further analyzed by using the SHAP algorithm. In figure 6.38 the top 20 highest mean absolute SHAP value of all patients in the test set for each variable is shown. In this figure it is shown that the swrt1 (SWRT) has the highest mean absolute value to predict the SWRT in the next year, which is to be expected. Also the cUHDRS, cogscore, and scnt1 have a high SHAP value. Here the cUHDRS and cogscore might be important because the SWRT is a component of the cUHDRS.

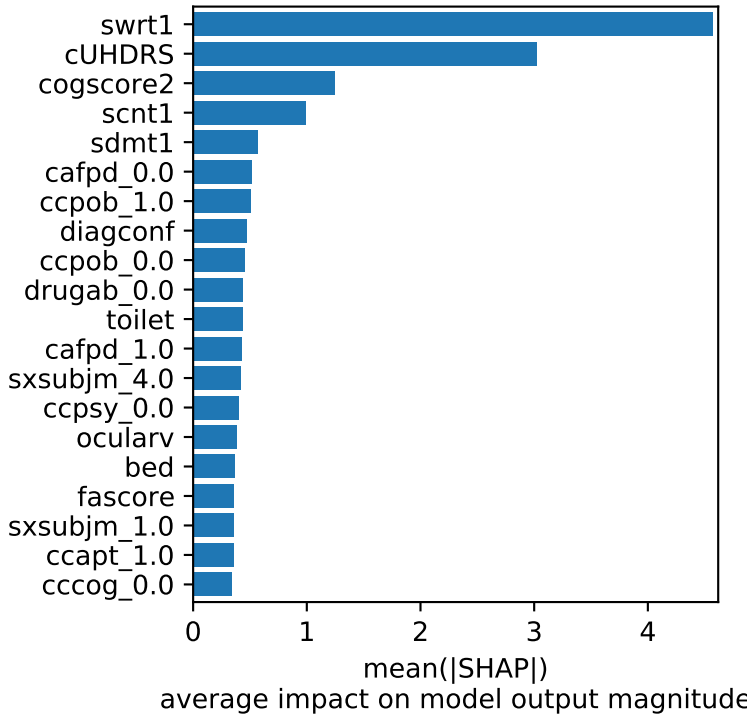


Figure 6.38: *The mean absolute SHAP value over all patients to predict the SWRT.*

6.7 Summary

To summarize, many models and hyperparameters were tuned which resulted in one best model for each variable, namely Drive, cUHDRS, TMS, TFC, SDMT, and SWRT. For each model the SHAP algorithm was applied to find the most important feature for each predicted label. In table 6.3 these results are shown for each predicted label. Here the expectation was that the most important variable would be equal to the predicted variable, except for the drive variable. However this was only the case for the SDMT and SWRT.

Table 6.3: *Summary of all the best models and most important variable for each predicted variable.*

Predicted Variable	**Best Model**	**Most Important Variable**	**Description Most Important Variable**
Drive	GRU	Indepscl	Subject's independence in %
cUHDRS	GRU	SWRT	Stroop Word Reading Test
TMS	CNN	Diagconf	Diagnostic confidence of motor abnormalities
TFC	GRU	Finances	Finances
SDMT	GRU	SDMT	Symbol Digit Modality Test
SWRT	GRU	SWRT	Stroop Word Reading Test

Chapter 7

Discussion & Conclusion

In this research multiple machine learning models were trained using the Enroll-HD dataset. This is one of the first research approaches where machine and deep learning models are used to analyze HD patients. With this research we want to introduce the HD research field to data-driven techniques in finding new or confirm known information. Using these kinds of models an advisory system for driving capability is created for HD patients. This system should show the feasibility of these kinds of models. In addition, another model was established to model the progression of the disease. This was achieved by modeling the longitudinal changes in cUHDRS, TMS, TFC, SWRT, and SDMT. Here the main goal was to give an accurate prognosis on the disease progression, defined by cUHDRS, to ensure that patients are treated in time. Here we also wanted find what variables affect the progression of these variables the most. With this information we want to confirm known or discover variables that affect the progression of the disease.

7.1 Research Questions

In total four research questions were tackled in this project. Here the research questions are given an answer and the results related to the questions are discussed.

7.1.1 Can ML models provide an accurate advice on driving capability to HD patients and clinicians?

To start with the advisory system of the driving capability is discussed to see if an accurate advice can be provided to HD patients and clinicians. Based on figure 6.2, 6.3, and 6.4 it was found that the GRU performed the best out of all models and was therefore further analyzed.

On average the GRU had an AUC of 0.962, an accuracy of 0.893, and a F1-score of 0.906, see figure 6.2 and appendix F. The results per positive, i.e. able to drive and negative, i.e. not able to drive, samples are shown in 6.5. This figure shows that the

driving capability can be accurately predicted in 91.41% of the positive examples and in 85.47% of the negative samples. This shows that the model is slightly more accurate for positive samples than negative examples. This is likely caused by the higher counts of positive training samples, see figure 6.1. The accuracy for the negative samples could be improved by adding more negative training samples.

For the miss-classified samples a deeper look can be provided by looking at figure 6.6. Here it was shown that a patient might be miss-classified in the clinic at time point 1, since the model predicted a value around 0.6, i.e. the patient was still able to drive according to the model. This value is close to the classification threshold (0.5) and is associated with a low accuracy around 0.5 in the test set. This shows that the model is very useful in identifying uncertain cases of driving capability. Here the model can provide the clinician with the information that the patient might still be able to drive. Their exact capability could for example be determined during a test, giving the patient the assurance to keep their freedom of movement, while still acknowledging the safety of themselves and others.

The model can also be used to make it very accurate for positive and negative examples by shifting the classification threshold. In figure 6.7 this is shown. When shifting the threshold to 0.11 the true positive rate (TPR) is 0.99. In that case almost all patients are allowed to drive prioritizing the freedom of movement. This also shows that if a patient's driving capability is predicted below 0.11 that it is almost 100% certain that the actual driving capability is indeed negative.

On the other hand the threshold can also be shifted to 0.94 to make the true negative rate (TNR) 0.99 (TNR=1-FPR). In that case the safety of the patient and others is prioritized. It also shows that if a patient's driving capability is predicted above 0.94 that it is almost 100% certain that the actual driving capability is indeed positive.

However, we do not make that decision for the patient and clinician, since this would raise many ethical questions. Therefore, the threshold is set to a default of 0.5 resulting in a TPR of 0.91 and a TNR of 0.86.

This threshold can also be shifted to make the TPR equal to the TNR, which might be more fair. However, it was already shown in the confusion matrix, see figure 6.5, that the model is already very accurate for both positive and negative samples.

When looking at the most important variables affecting the driving model, see figure 6.8, it was shown that the independence scale had the highest impact. In addition, many variables from the functional assessment score (fascore) were also important, which driving capability is a part of. Now that it is known which variables mostly impact the driving capability it is possible to use only a limited number of variables. This can be achieved by using the SHAP values and measuring how much the error increases when only using a limited number of variables. This is possible since the expected value (calculated by the SHAP algorithm) and the SHAP values add up to the predicted values. With this in mind we can control how many variables should be added to minimize the increased error compared to the error when using all the variables. This allows us to reduce input size which reduces computational intensity and increases model interpretability. This

allows for more hyperparameter tuning to make the model more accurate. This makes it also possible to create a more simpler model which can be used in the clinic.

In short, an accuracy of 91% was achieved for positive samples and an accuracy of 85% for the negative samples. When the model predicts a value above 0.94 it will almost be 100% accurate that the patient is able to drive. When the model predicts a value below 0.11 it will almost be 100% accurate that the patient is unable to drive. In uncertain cases, e.g. a lower/higher predicted outcome than the label or a predicted outcome with a low associated accuracy, a deeper look can be provided to the HD patient to determine their driving capability. This all illustrates that indeed an accurate advice can be given on driving capability to both HD patients and clinicians. In the future a more accurate model can be created by filtering out input variables with the help of the SHAP values.

7.1.2 Can ML models provide a personalized prognosis on HD progression, defined by the cUHDRS, to HD patients and clinicians?

A model that can give a prognosis of cUHDRS for each patient was successfully created. This was achieved by creating a model that can predict the cUHDRS one year ahead in time when given one or multiple visits. To our knowledge no models have been created yet which could provide such a prognosis on cUHDRS. Based on figure 6.10, 6.11, and 6.12 it was found that the GRU was the best performing model and it was therefore further analyzed. On average the GRU had a MAE of 12.460, a RMSE of 16.508, a Max AE of 124.821, and a R^2 score of 0.943, see figure 6.10 and appendix G.

This model allows the clinic to foresee changes in disease progression a year ahead in time. Which can be used to ensure a patient is treated in time and to identify optimal moments for intervention. It could also be used in selecting patients for a clinical trial, since for such trials patients that have a change in their symptoms are needed.

However this does require the model to be accurate and reliable. To have a deeper look into the error in the predictions of the model the distribution of the absolute error over all predictions in the test set was shown in figure 6.13. This shows that the model had a very low error margin (<5) for a lot of visits/time steps and for quite some other visits/time steps the error margin is a lot higher (>35). However, it was shown by Trundell et al.[35] (2019) that a worsening of 1.2 in cUHDRS can already be considered clinically meaningful.[35] Therefore the model could be used for some samples with a low error and the model should improve for the higher error samples.

What causes these high errors is still unknown. Possible solutions to these high errors include: identifying and filtering noise or error in the data, increase the training data for certain test samples, testing more hyperparameters, or adding essential variables to the data, e.g. medicine and nutritional supplement intake and imaging or bio markers.

To identify if more training data is needed or whether the data is noisy a new model

is proposed, namely the Bayesian neural network (BNN). Using a BNN the uncertainty can be modelled. There are two main types of uncertainty, namely aleatoric uncertainty and epistemic uncertainty. The aleatoric uncertainty measures the noise in the data and the epistemic uncertainty measures the uncertainty in the model, which is caused by insufficient amount of training data. Using a BNN would allow us to determine whether inaccurate predictions are made, because of noise in the data or because there is simply insufficient amount of training data for a specific test sample. In addition, it can also aid in explaining the model predictions, e.g. if the uncertainty is low for a prediction it would indicate to the clinician that the prediction is accurate and vice versa.[36]

Another solution to make the model more accurate is to let experts detect errors in the data. Such experts can decide whether certain changes in the cUHDRS over time are actually possible. By filtering out these outliers/errors the model might be able to converge better during training.

Another way of increasing the accuracy of the model is to add medicine and nutritional supplement intake or bio and imaging markers. Currently, the Enroll-HD dataset does include information about medicine and nutritional supplement intake. However, this data was not included in this study. Adding such information might increase the accuracy of the model.

In addition, more visits can also be added to the data. There is still unused data from the Registry and Ad Hoc study, which were mentioned in chapter 2. More data would increase the number of training samples, which could possibly increase the accuracy of the model.

Additional imaging and biological markers were not available in the dataset. However it is possible to request additional bio-samples and imaging data. This could provide relevant information with regards to our models. Many papers have already shown that there are promising bio and imaging markers which could help with further understanding HD.[37,38,39,40,41,42]

In conclusion, accurate predictions can be made with the GRU model. These predictions could make early interventions possible before symptom changes and ensures that patient can be treated at the right time. It could also be used in clinical trials. However, high errors are still possible in predictions, which could be attributed to noise in the data or insufficient training data. To identify which predictions are accurate a BNN could be used. Another solution is to increase the accuracy of the GRU by filtering outliers in the labels, filtering out noise in the data, adding more training data, train the model more on hyperparameters, or adding more bio and imaging markers.

7.1.3 Do the variables making up the cUHDRS also affect the progression of cUHDRS the most?

For this research question it was the goal to determine what variables affect disease progression. Here disease progression is defined by the cUHDRS, which is a combination of

the main symptoms of HD. These symptoms include motor (TMS), functional (TFC), and cognitive (SDMT and SWRT) symptoms. To show which variables affect the cUHDRS the most the mean absolute SHAP values were calculated. These SHAP values show which variables were most important in predicting the cUHDRS. The top 20 of the SHAP values are shown in figure 6.14. It was expected that the variables making up the cUHDRS would also affect the progression of the cUHDRS the most. However, this was not the case. It was shown in figure 6.14 that the SWRT, cUHDRS, and SDMT had the largest impact. The TFC was the twelfth most important variable and the TMS was not within the top 20. Only one component of the TMS was present in the top 20, namely the tandem walking (tandem) variable, which was the seventh most impactful variable. This suggests that the TMS and TFC do not affect the progression of the cUHDRS as much as the SWRT and SDMT, i.e. the cognitive variables. This shows that with the proposed data-driven approach new hypothesis can be generated by discovering unknown patterns within the data.

The components of the TFC and TMS were also further analyzed, to see whether they had an effect on the progression of the cUHDRS. In figure 6.15 the SHAP values of the TFC components to predict the cUHDRS are shown. This figure showed that the finances variables was the most important TFC component to predict the cUHDRS. This variable was followed by the occupation variable, then domestic chores (chores) and ADL, and ending on the Care Level (carelevl) variables which was least important. This seems to indicate that some components of the TFC are more important in predicting the disease progression than other components. This might be due to the different impacts these variables can have on HD patients. Here the Care Level (carelevl) variable might have a lesser impact than the ability to manage your own finances. Note that the care level (carelvl) and domestic chores (chores) variables range form 0-2 and the other TFC components range from 0-3. This already shows that they have a lower in impact in the TFC score and therefore in the cUHDRS. However, it was still shown that variables with similar value ranges had a different impact, e.g. finances had a greater impact than occupation (occupatn).

Figure 6.16 also showed that some TMS components are more important in predicting the progression of the cUHDRS. The most important variable was tandem walking (tandem). This again indicates that some components of the TMS are more important in predicting the disease progression than other components. This could be due to many factors, according to clinicians. For instance a variable might be more consistent, e.g. some variables might increase or decrease in value each visit, while other variables always increase or stay the same with each visit. Another factor could be that some measurements of variables are more affected by external factors, e.g. the stress of the patient or how much pain the patient is experiencing during the measurement. Varying impact values can also be caused by the handedness of the patient, e.g. for the rigidity-arms, pronate supinate-hands, and finger taps.

These varying impact values found in the TMS, TFC, and in the top 20 most important variables should give hints at underlying patterns in the data. This can help

clinicians with deciding what to research or help them confirm their intuition about certain variables or components of the TMS and TFC that they want to research.

In short, the progression of the cUHDRS is better predicted by the cUHDRS itself and the SDMT and SWRT components. The progression of the cUHDRS is not so much affected by the TFC and TMS, even though the cUHDRS is a measure of disease progression. It was also found that the variables making up the TMS and TFC have varying impacts on the progression of the cUHDRS. Which shows that some components of the TMS and TFC are more important than others, with regards to predicting the cUHDRS progression. This could suggest that the cUHDRS, can be made more reliable when taking into account the impact of each variable on the disease/patient. This all might indicate that the cUHDRS is mainly a measure of cognitive progression, even though it is used as a measure of disease progression.

7.1.4 Which variables affect the components of the cUHDRS the most?

For this research question each individual score of the cUHDRS (TMS, TFC, SWRT, and SDMT) is also modelled. Here we want to deduct what impacts the progression of motor (TMS), functional (TFC), and cognitive (SWRT, SDMT) symptoms using a data-driven approach. With this approach the aim was to confirm known information and to find variables that might impact disease progression, which were previously unknown. Different models were tuned and the best performing model was deducted using the MAE, RMSE, Max AE, and the R^2 score. With the best model of each variable, the mean absolute SHAP values were calculated to deduct which variables affect the outcome of the model the most.

Firstly, the TMS was modelled. It was found that the CNN was the best performing model for the TMS. On average the CNN had a MAE of 5.371, RMSE of 7.6, Max AE of 53.756, and a R^2 score of 0.918 on the test set, see figure 6.18 and appendix H. Based on figure 6.19 it was found that the CNN only slightly improved over time, except for the last time step. This might suggest that the model can also be used on cross-sectional data. It was also shown that the model can predict the recovered (n=470) and progressive (n=1354) group with a similar error, see figure 6.20. Here the progressive group had a change in the label (TMS) between the first and last visit and vice versa for the recovered group. It was also found that the error between the CNN and the baseline model is different on average, temporal, and per status group. This might suggest that the model is not biased towards a stable, recovered or progressive trajectory and that is does not simply predict the value at the previous time step. Which shows that the model actually learned useful patterns within the data.

Using this model the mean absolute SHAP values were calculated. The 20 variables with the highest SHAP values are shown in figure 6.21. This figure showed that the TMS itself is not the most important in predicting the TMS in the next year. The figure showed

that the diagnostic confidence of motor abnormalities (diagconf) was the most important variable. It also showed varying values for the different components of the TMS, which are described in appendix A. The component with the highest SHAP value, see figure 6.22, was the tandem walking (tandem) variable and the least important variable was the retropulsion pull (retropls). This suggests that these different components vary in impact on the progression of the TMS. This could be attributed to many factors, which were also mentioned for the TMS components when modelling the cUHDRS. These factors include: the consistency of a variable, how much each variable is affected by external factors, e.g. pain, stress or handedness.

Secondly, the TFC was modelled. It was found that the GRU was the best performing model for the TFC. On average the GRU had a MAE of 0.873, RMSE of 1.292, Max AE of 8.714, and a R^2 score of 0.899 on the test set, see figure 6.24 and appendix I. It was found that the model improves over time however the performance is very similar for the first and last visit. This suggests that the model could be used on cross-sectional data. It was also found that the model has a similar error for the progressive and recovered patients. It was also found that the model performs differently than the baseline model. This suggests that the model is learning useful patterns within the data.

Using this model the mean absolute SHAP values were calculated. The 20 variables with the mean absolute highest SHAP values are shown in figure 6.27. This figure showed that the finances variable, a component of the TFC score, was the most important variable in predicting the TFC score in the next year. However, the TFC score itself (tfcscore) was the second most important variable. This was followed by the subject's independence in % (indepscl) and the other components making up the TFC score, ADL, occupation (occupatn), care level (carelvl), domestic chores (chores). It was to be expected that the components of the TFC score and the TFC score itself would be important in predicting the TFC score in the next year. However, it is interesting that the variables do not equally contribute to the output of the model, which is shown in figure 6.28. Here it is shown that domestic chores (chores) and the care level (carelevl) variables contribute the least out of all TFC components. This might be due to the smaller distribution of values in these variables, ranging from 0-2, compared to the other variables, which range from 0 to 3. However, there is also a difference in SHAP value between the other TFC components. The finances variable is more important than the occupation (occupatn) and ADL variable, even though the range of the values is the same. This suggests that some TFC components affect the progression of the TFC score more than other components. Here the clinicians did argue that in their opinion the components of the TFC do not have an equal impact on the life of the patient. Here we present that indeed there is a different impact in terms of TFC progression.

Thirdly, the SDMT was modelled. It was found that the GRU was the best performing model for the SDMT. On average the GRU had a MAE of 4.149, a RMSE of 5.658, a Max AE of 54.264, and a R^2 score of 0.908, see figure 6.30 and appendix J. It was found that the GRU performed worse at predicting the first year than predicting the subsequent years. However the difference in MAE was only between 0.25 or 0.5, see figure 6.31.

This shows that the model could also be used on cross-sectional data. It was also found that the GRU performed reasonably similar for the progressive (n=1213) and recovered (n=602) group, see figure 6.37. Again it was found that the GRU performed differently than the baseline. This suggests that the model actually learned useful patterns within the data.

The SHAP algorithm was used to calculate the mean absolute SHAP values for each variable. The top 20 highest SHAP variables are shown in figure 6.33. The figure showed that the SDMT (sdmt1) had the biggest impact in predicting the SDMT in the next year, which was to be expected. Here the cUHDRS, cognitive score (cogscore2), and SWRT (swrt1) also had a small impact on the output of the model. The cUHDRS and cognitive score (cogscore2) might be important here because they include the SDMT. This indicates that these cognitive measures (SDMT and SWRT) are generally useful in predicting the progression of the SDMT. It also shows that the SDMT is a good standalone variable.

Finally, the SWRT was modelled. It was found that the GRU was the best performing model for the SWRT. On average the GRU had a MAE of 7.839, RMSE of 10.367, Max AE of 75.963, and a R^2 score of 0.876, see figure 6.35 and appendix K. In figure 6.36 it was shown that the MAE of the GRU is reducing with each subsequent year predicted. This might indicate that this model is better to use in longitudinal data than cross-sectional data. It was also shown that the GRU performed similar for the progressive (n=1310) and recovered (n=607) group, see figure 6.37. Again it was shown that the GRU had a different performance than the baseline/constant model, in terms of the average, temporal, and status group metrics. This suggest that the model did learn useful patterns within the data to predict the progression of SWRT.

Again the SHAP algorithm was used to calculate the mean absolute SHAP values for each variable. In figure 6.38 the top 20 highest SHAP values are shown. This figure showed that, as expected, the SWRT had the biggest impact in predicting the SWRT in the next year. This was followed by the cUHDRS, cognitive score (cogscore2), and the stroop colour naming test (SCNT). Similar to the SDMT, the cUHDRS and cognitive score (cogscore2) might be important, because they include the SWRT variable. This indicates that generally the SWRT is useful in predicting the SWRT progression and that it is a good standalone variable.

In short, the progression of TMS and TFC are mainly affected by the scores themselves and the components of those scores. Here it was found that the components do vary in importance to predict the progression of the TMS and TFC. This indicates that some components have a bigger impact on the progression of the disease, in terms of motor (TMS) and function (TFC). For the SDMT it was mainly found that only the variable itself had an effect in predicting the progression of the SMDT. The same was true for the SWRT.

7.1.5 Summary

With all the research questions answered it is now clear to what extend our approach can answer these questions. It was shown that an accurate advisory system on driving capability was created. However, the model might need to be down scaled in terms of input variables before using it in the clinic. In addition, it would be beneficial for the clinicians and patients to have a dashboard/application which explains the model and the predictions. This would make the model more interpretable and easy to use.

It was also shown that an accurate prognosis can be given on disease progression (cUHDRS). However, there are some predictions with a very high error. These errors need to be reduced or it should be clarified on what kind of input data the prediction error becomes to high. A good target is to reduce the MAE of the model below 1.2, since it was shown by Trundell et al.[35] (2019) that a worsening of 1.2 in cUHDRS can be considered clinical meaningful. To detect high error predictions Bayesian neural networks could be used. In addition, prediction error can be reduced by adding more training data, reducing noise in the data, or removing variables that had a very low impact on model output. With these further improvements it might be possible to use the model to treat patients in time for symptom changes or to select patients for clinical trials.

Lastly, it was found that new hypothesis could be generated based on the data-driven approach. The benefit of this approach was that no patients or variables are filtered out based on some inclusion criteria. This reduces the bias that was introduced and allowed us to reassess what variables are important for certain symptoms. This resulted in very straightforward results confirming known information, but it also resulted in clues for new hypotheses. An example is that the cUHDRS progression was mainly influenced by cognitive measures and not so much by functional (TFC) or motoric (TMS) variables. This shows that with this approach it is possible to generate new hypotheses by providing new directions to further research.

7.2 Future work

Previously many improvements were already mentioned to improve the model performance and interpretability. These included identifying and filtering noise or error in the data, increase the training data for certain test samples, testing more hyperparameters, or adding essential variables to the data, e.g. medicine and nutritional supplement intake and imaging or bio markers. Here even more improvements are proposed and some elaboration is given on already suggested improvements.

7.2.1 Data

In chapter 2 it was stated that only the enroll data was used from the Enroll-HD study, however two more datasets were available from this study, namely registry and adhoc. Adding these studies to the dataset would add another 15,000 visits, however it would

have also increased the number of missing values, due to new variables being added which did not exist in the enroll data or vice versa. However, we have established the whole process of pre-processing and training machine learning models. Therefore, this workflow could be expanded to include the additional data to our main dataset and analysis. This could lead to better performing models, since more data can be used to train the models.

7.2.2 Temporal Resolution

In the methods in section 5.3.4, it was described how the data was reshaped to make longitudinal predictions possible. The data was transformed to a 3-dimensional matrix, like so: (participants, time step, features). Here each time step would be around 1 years apart to mimic the regular yearly visits of the enroll dataset, see figure 2.5. However, it might be more beneficial to reduce the months between the time steps, also known as temporal resolution, since it would increase the number of visits per patients used and it might increase the accuracy of the exact time step of a specific visit. Then again this would also increase the dimensionality of the data, increasing the training time of the models and possibly result in even worse results. In the future a smaller temporal resolution could be tested to show if the models perform better.

7.2.3 Feature Selection

In this research the goal was to model HD progression to deduct which variables might influence the progression. Here a data-driven technique was used, i.e. the models themselves should learn what variables are important and which are not. Therefore, the feature selection step was less intensive, which was described in 5.3.2. Here it was explained that only some variables were discarded. This method was beneficial for our specific goal, since we wanted to reduce the possibility of filtering out relevant data and introducing bias. However, multiple sources have shown that selecting features increases performance of models, especially for machine learning models.[43,44,45] Also, filtering out irrelevant data also reduces the input size, which reduces computational intensity. This allows for faster hyperparameter tuning of models, which might results in better performing models. It also makes it feasible to use more computational intensive models, e.g. Radial Basis Function (RBF) kernel SVM.

In this study the most important variables were deducted using the SHAP algorithm. This gives us information on what data is relevant in predicting the output of the model. Therefore, these irrelevant variables can be filtered out and our proposed models can be trained on the filtered dataset. This could increase the performance of the current models and allow for faster and therefore more hyperparameter tuning.

7.3 Conclusion

In conclusion, many machine learning models were successfully applied on the Enroll-HD dataset. We showcased that machine learning is a feasible method to further analyze the Enroll-HD dataset. It was shown that a GRU is generally the best model to give an accurate advice on driving capability and modeling disease progression, in terms of cUHDRS, TMS, TFC, SDMT, and SWRT, in pre-manifest and manifest HD patients. Using a data-driven approach it was shown that the progression of cUHDRS is mainly predicted by cognitive measures (SWRT and SDMT) and that the components of the TMS and TFC vary in importance with regards to predicting the progression of cUHDRS. It was also found that the TMS components have a varying impact on TMS progression, which was also the case for the TFC components on the TFC progression. Finally, it was found that the SDMT and SWRT progression is mainly influenced by the variables themselves, which shows that these variables are good standalone measures. With this approach a prognosis on the disease progression could also be given. However, there is a risk that some predictions have a very high error. What causes these high errors is still unknown. However, many solutions were proposed to tackle this problem, namely using Bayesian neural networks, filtering out noise in the data, and adding more training data in terms of visits and variables. With this study the HD research field is introduced to using ML on the Enroll-HD dataset and to use a data-driven approach to predict and derive variable importance. With this information the research field is also introduced to new patterns in the data which can lead to new hypothesis, e.g. that TMS components have a varying impact on cUHDRS and TMS progression. This might indicate that some components are more impactful than others, which suggests that these components also have to be treated differently in analysis. For now this is not the case, however here we present that this might be helpful in improving HD research.

Bibliography

[1] Praveen Dayalu and Roger L. Albin. "Huntington Disease: Pathogenesis and Treatment". In: *Neurologic Clinics* 33.1 (2015). Movement Disorders, pp. 101–114. ISSN: 0733-8619. DOI: https://doi.org/10.1016/j.ncl.2014.09.003. URL: http://www.sciencedirect.com/science/article/pii/S0733861914000711.

[2] Samuel Frank. "Treatment of Huntington's Disease". In: *Neurotherapeutics* 11.1 (Dec. 2013), pp. 153–160. DOI: 10.1007/s13311-013-0244-z. URL: https://doi.org/10.1007/s13311-013-0244-z.

[3] Caron NS;Wright GEB;Hayden MR; "Huntington Disease". In: *National Center for Biotechnology Information* (). URL: https://pubmed.ncbi.nlm.nih.gov/20301482/.

[4] Christian Landles and Gillian P Bates. "Huntingtin and the molecular pathogenesis of Huntington's disease". In: *EMBO reports* 5.10 (Oct. 2004), pp. 958–963. DOI: 10.1038/sj.embor.7400250. URL: https://doi.org/10.1038/sj.embor.7400250.

[5] N. Ahmad Aziz et al. "Overlap between age-at-onset and disease-progression determinants in Huntington disease". In: *Neurology* 90.24 (2018), e2099–e2106. ISSN: 0028-3878. DOI: 10.1212/WNL.0000000000005690. eprint: https://n.neurology.org/content/90/24/e2099.full.pdf. URL: https://n.neurology.org/content/90/24/e2099.

[6] Jae Whan Keum et al. "The HTT CAG-Expansion Mutation Determines Age at Death but Not Disease Duration in Huntington Disease". In: *The American Journal of Human Genetics* 98.2 (Feb. 2016), pp. 287–298. DOI: 10.1016/j.ajhg.2015.12.018. URL: https://doi.org/10.1016/j.ajhg.2015.12.018.

[7] Jane Paulsen et al. "Clinical and biomarker changes in premanifest Huntington disease show trial feasibility: a decade of the PREDICT-HD study". In: *Frontiers in Aging Neuroscience* 6 (2014), p. 78. ISSN: 1663-4365. DOI: 10.3389/fnagi.2014.00078. URL: https://www.frontiersin.org/article/10.3389/fnagi.2014.00078.

[8] Alvin Rajkomar, Jeffrey Dean, and Isaac Kohane. "Machine Learning in Medicine". In: *New England Journal of Medicine* 380.14 (Apr. 2019), pp. 1347–1358. DOI: 10.1056/nejmra1814259. URL: https://doi.org/10.1056/nejmra1814259.

[9] Eric Topol. *Deep medicine. how artificial intelligence can make healthcare human again*. Basic Books, 2019.

[10] Simon Ronicke et al. "Can a decision support system accelerate rare disease diagnosis? Evaluating the potential impact of Ada DX in a retrospective study". In: *Orphanet Journal of Rare Diseases* 14.1 (Mar. 2019). DOI: 10.1186/s13023-019-1040-6. URL: https://doi.org/10.1186/s13023-019-1040-6.

[11] Scott Mayer McKinney et al. "International evaluation of an AI system for breast cancer screening". In: *Nature* 577.7788 (Jan. 2020), pp. 89–94. DOI: 10.1038/s41586-019-1799-6. URL: https://doi.org/10.1038/s41586-019-1799-6.

[12] Natalia P. Rocha et al. "The Clinical Picture of Psychosis in Manifest Huntington's Disease: A Comprehensive Analysis of the Enroll-HD Database". In: *Frontiers in Neurology* 9 (2018), p. 930. ISSN: 1664-2295. DOI: 10.3389/fneur.2018.00930. URL: https://www.frontiersin.org/article/10.3389/fneur.2018.00930.

[13] Yury Seliverstov et al. "F49 Machine learning approach in analysis of enroll-hd data for suicidality prediction in huntington disease". In: *Journal of Neurology, Neurosurgery & Psychiatry* 89.Suppl 1 (2018), A57–A57. ISSN: 0022-3050. DOI: 10.1136/jnnp-2018-EHDN.152. eprint: https://jnnp.bmj.com/content/89/Suppl_1/A57.2.full.pdf. URL: https://jnnp.bmj.com/content/89/Suppl_1/A57.2.

[14] Yury Seliverstov et al. "I9 The size of the CAG-expansion mutation can be predicted in hd based on phenotypic data using a machine learning approach". In: *Journal of Neurology, Neurosurgery & Psychiatry* 87.Suppl 1 (2016), A62–A62. ISSN: 0022-3050. DOI: 10.1136/jnnp-2016-314597.174. eprint: https://jnnp.bmj.com/content/87/Suppl_1/A62.1.full.pdf. URL: https://jnnp.bmj.com/content/87/Suppl_1/A62.1.

[15] Enroll-HD. *Enroll-HD PDS5: PDS5 Overview*. 2021. URL: https://enroll-hd.org/enrollhd_documents/2020-10-R1/Enroll-HD-PDS5-Overview-2020-10-R1.pdf (visited on 06/07/2021).

[16] Scott A. Schobel et al. "Motor, cognitive, and functional declines contribute to a single progressive factor in early HD". In: *Neurology* 89.24 (2017), pp. 2495–2502. ISSN: 0028-3878. DOI: 10.1212/WNL.0000000000004743. eprint: https://n.neurology.org/content/89/24/2495.full.pdf. URL: https://n.neurology.org/content/89/24/2495.

[17] Carlos Estevez-Fraga et al. "Composite UHDRS Correlates With Progression of Imaging Biomarkers in Huntington's Disease". In: *Movement Disorders* 36.5 (2021), pp. 1259–1264. DOI: https://doi.org/10.1002/mds.28489. eprint: https://movementdisorders.onlinelibrary.wiley.com/doi/pdf/10.1002/mds.28489. URL: https://movementdisorders.onlinelibrary.wiley.com/doi/abs/10.1002/mds.28489.

[18] Enroll-HD. *Enroll-HD protocol*. 2021. URL: https://enroll-hd.org/enrollhd_documents/Enroll-HD-Protocol-1.0.pdf (visited on 06/07/2021).

[19] Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow*. O'Reilly Media, Inc, 2019.

[20] S. Wang et al. "Training deep neural networks on imbalanced data sets". In: *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016, pp. 4368–4374.

[21] Gary King and Langche Zeng. "Logistic Regression in Rare Events Data". In: *Political Analysis* 9.2 (2001), pp. 137–163. DOI: 10.1093/oxfordjournals.pan.a004868.

[22] Sebastian Raschka. *Confusion Matrix*. URL: http://rasbt.github.io/mlxtend/user_guide/evaluate/confusion_matrix/.

[23] Roman M. Balabin, Ravilya Z. Safieva, and Ekaterina I. Lomakina. "Comparison of linear and nonlinear calibration models based on near infrared (NIR) spectroscopy data for gasoline properties prediction". In: *Chemometrics and Intelligent Laboratory Systems* 88.2 (2007), pp. 183–188. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2007.04.006. URL: http://www.sciencedirect.com/science/article/pii/S0169743907000925.

[24] Pejman Tahmasebi and Ardeshir Hezarkhani. "Application of a Modular Feedforward Neural Network for Grade Estimation". In: *Natural Resources Research* 20.1 (Jan. 2011), pp. 25–32. DOI: 10.1007/s11053-011-9135-3. URL: https://doi.org/10.1007/s11053-011-9135-3.

[25] Vikas Gupta. *Understanding Feedforward Neural Networks*. 2017. URL: https://www.learnopencv.com/understanding-feedforward-neural-networks/ (visited on 09/10/2020).

[26] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[27] Yann A. LeCun et al. "Efficient BackProp". In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 9–48. DOI: 10.1007/978-3-642-35289-8_3. URL: https://doi.org/10.1007/978-3-642-35289-8_3.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[29] Daniel Lopez. *RNN, LSTM & GRU*. Apr. 2019. URL: dprogrammer.org/rnn-lstm-gru (visited on 01/27/2021).

[30] Kyunghyun Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[31] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[32] L. Jiang et al. "Prediction of SNP Sequences via Gini Impurity Based Gradient Boosting Method". In: *IEEE Access* 7 (2019), pp. 12647–12657. DOI: 10.1109/ACCESS.2019.2893269.

[33] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[34] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 4765–4774. URL: http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

[35] Dylan Trundell et al. "Defining Clinically Meaningful Change on the Composite Unified Huntington's Disease Rating Scale (cUHDRS) (P1.8-043)". In: *Neurology* 92.15 Supplement (2019). ISSN: 0028-3878. eprint: https://n.neurology.org/content. URL: https://n.neurology.org/content/92/15_Supplement/P1.8-043.

[36] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *CoRR* abs/1703.04977 (2017). arXiv: 1703.04977. URL: http://arxiv.org/abs/1703.04977.

[37] Fanny Mochel et al. "Early Energy Deficit in Huntington Disease: Identification of a Plasma Biomarker Traceable during Disease Progression". In: *PLOS ONE* 2.7 (July 2007), pp. 1–8. DOI: 10.1371/journal.pone.0000647. URL: https://doi.org/10.1371/journal.pone.0000647.

[38] Eric R. Reed et al. "MicroRNAs in CSF as prodromal biomarkers for Huntington disease in the PREDICT-HD study". In: *Neurology* 90.4 (2018). Ed. by et al., e264–e272. ISSN: 0028-3878. DOI: 10.1212/WNL.0000000000004844. eprint: https://n.neurology.org/content/90/4/e264.full.pdf. URL: https://n.neurology.org/content/90/4/e264.

[39] Peter A. Wijeratne et al. "Robust Markers and Sample Sizes for Multicenter Trials of Huntington Disease". In: *Annals of Neurology* 87.5 (2020), pp. 751–762. DOI: https://doi.org/10.1002/ana.25709. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/ana.25709. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/ana.25709.

[40] Jeffrey D. Long et al. "8OHdG as a marker for Huntington disease progression". In: *Neurobiology of Disease* 46.3 (2012). Non-motor Aspects of PD, pp. 625–634. ISSN: 0969-9961. DOI: https://doi.org/10.1016/j.nbd.2012.02.012. URL: https://www.sciencedirect.com/science/article/pii/S0969996112000678.

[41] Cristina Sánchez-Castañeda et al. "Seeking Huntington disease biomarkers by multimodal, cross-sectional basal ganglia imaging". In: *Human brain mapping* 34.7 (2013), pp. 1625–1635.

[42] Valerio Leoni et al. "Plasma 24S-hydroxycholesterol correlation with markers of Huntington disease progression". In: *Neurobiology of Disease* 55 (2013),

pp. 37–43. ISSN: 0969-9961. DOI: https://doi.org/10.1016/j.nbd.2013.03.013. URL: https://www.sciencedirect.com/science/article/pii/S0969996113001010.

[43] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[44] Avrim L. Blum and Pat Langley. "Selection of relevant features and examples in machine learning". In: *Artificial Intelligence* 97.1 (1997). Relevance, pp. 245–271. ISSN: 0004-3702. DOI: https://doi.org/10.1016/S0004-3702(97)00063-5. URL: https://www.sciencedirect.com/science/article/pii/S0004370297000635.

[45] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*. Vol. 454. Springer Science & Business Media, 2012.

Appendices

Appendix A

TMS Components

Table A.1: *All components of the TMS (the sum of all components).*

Group	Names	Variables
Ocular Pursuit	Horizontal	ocularh
	Vertical	ocularv
Saccade initiation	Horizontal	sacinith
	Vertical	sacinitv
Saccade velocity	Horizontal	sacvelh
	Vertical	sacvelv
Dysarthria	Dysarthria	dysarth
Tongue protrusion	Tongue protrusion	tongue
Finger taps	Right	fingtapr
	Left	fingtapl
Pronate supinate-hands	Right	prosupr
	Left	prosupl
Luria	Luria	luria
Rigidity-arms	Right	rigarmr
	Left	rigarml
Bradykinesiabody	Bradykinesiabody	brady
Maximal dystonia	Trunk	dysttrnk
	RUE	dystrue
	LUE	dystlue
	RLE	dystrle
	LLE	dystlle
Maximal chorea	Face	chorface
	BOL	chorbol
	Trunk	chortrnk
	RUE	chorrue

	LUE	chorlue
	RLE	chorrle
	LLE	chorlle
Gait	Gait	gait
Tandem walking	Tandem walking	tandem
Retropulsion pull	Retropulsion pull	retropls

Appendix B

Variable Statistics

Table B.1: *A list of all variables in the imputed dataset and their: mean, standard deviation (std), minimum value (min), 25^{th} percentile, 50^{th} percentile (median), 75^{th} percentile, and maximum value (max).*

variable	mean	std	min	25%	50%	75%	max
sex	0.542	0.498	0.000	0.000	1.000	1.000	1.000
caghigh	43.267	3.170	36.000	41.000	43.000	45.000	59.000
caglow	18.252	2.931	8.000	17.000	17.000	19.000	28.000
fhx	0.880	0.325	0.000	1.000	1.000	1.000	1.000
hxtobcpd	7.911	11.086	0.000	0.000	0.000	15.000	100.000
hxtobyos	9.732	13.549	0.000	0.000	0.000	19.000	74.000
hxpacky	8.942	15.775	0.000	0.000	0.000	12.500	195.000
age	50.369	13.614	18.000	40.000	51.000	60.000	94.000
hddiagn_est	0.287	0.452	0.000	0.000	0.000	1.000	1.000
parentagesx	0.977	12.391	-44.000	-7.000	1.000	9.000	72.000
ccmtrage	6.349	6.028	-7.000	0.000	6.000	10.000	49.000
sxsubj	6.104	6.288	-7.000	0.000	5.000	10.000	58.000
sxfam	6.376	6.586	-7.000	0.000	5.000	10.000	53.000
ccdepage	7.609	9.645	-6.000	0.000	5.000	12.000	71.000
ccirbage	5.405	7.830	-7.000	0.000	3.000	8.000	70.000
ccvabage	2.656	6.284	-7.000	0.000	0.000	3.000	74.000
ccaptage	3.575	5.613	-7.000	0.000	1.000	6.000	71.000
ccpobage	3.054	6.143	-7.000	0.000	0.000	4.000	62.000
ccpsyage	0.707	3.257	-7.000	0.000	0.000	0.000	62.000
cccogage	3.161	4.819	-6.000	0.000	0.000	6.000	53.000
rtrddur	3.195	6.006	-6.000	0.000	0.000	5.000	67.000
bmi	25.439	5.317	8.900	21.800	24.600	28.100	66.000
alcunits	2.781	6.985	0.000	0.000	0.000	2.000	170.000
tobcpd	3.322	7.734	0.000	0.000	0.000	0.000	120.000
tobyos	5.140	11.647	0.000	0.000	0.000	0.000	70.000

packy	4.323	11.744	0.000	0.000	0.000	0.000	270.000
manifest	0.702	0.457	0.000	0.000	1.000	1.000	1.000
capscore	103.106	24.439	21.212	87.081	107.177	120.255	245.614
motscore	31.323	25.914	0.000	6.000	29.000	49.000	126.000
diagconf	2.955	1.594	0.000	1.000	4.000	4.000	4.000
ocularh	0.903	1.016	0.000	0.000	1.000	2.000	4.000
ocularv	1.013	1.072	0.000	0.000	1.000	2.000	4.000
sacinith	1.164	1.150	0.000	0.000	1.000	2.000	4.000
sacinitv	1.191	1.163	0.000	0.000	1.000	2.000	4.000
sacvelh	1.079	1.135	0.000	0.000	1.000	2.000	4.000
sacvelv	1.148	1.194	0.000	0.000	1.000	2.000	4.000
dysarth	0.671	0.872	0.000	0.000	0.000	1.000	4.000
tongue	0.801	1.074	0.000	0.000	0.000	1.000	4.000
fingtapr	1.222	1.147	0.000	0.000	1.000	2.000	4.000
fingtapl	1.348	1.179	0.000	0.000	1.000	2.000	4.000
prosupr	1.068	1.132	0.000	0.000	1.000	2.000	4.000
prosupl	1.208	1.163	0.000	0.000	1.000	2.000	4.000
luria	1.327	1.374	0.000	0.000	1.000	2.000	4.000
rigarmr	0.621	0.797	0.000	0.000	0.000	1.000	4.000
rigarml	0.601	0.802	0.000	0.000	0.000	1.000	4.000
brady	1.069	1.180	0.000	0.000	1.000	2.000	4.000
dysttrnk	0.582	0.937	0.000	0.000	0.000	1.000	4.000
dystrue	0.530	0.875	0.000	0.000	0.000	1.000	4.000
dystlue	0.516	0.870	0.000	0.000	0.000	1.000	4.000
dystrle	0.409	0.781	0.000	0.000	0.000	1.000	4.000
dystlle	0.400	0.773	0.000	0.000	0.000	1.000	4.000
chorface	0.894	0.930	0.000	0.000	1.000	1.000	4.000
chorbol	0.873	0.958	0.000	0.000	1.000	1.000	4.000
chortrnk	0.924	0.990	0.000	0.000	1.000	2.000	4.000
chorrue	1.001	0.969	0.000	0.000	1.000	2.000	4.000
chorlue	0.998	0.968	0.000	0.000	1.000	2.000	4.000
chorrle	0.889	0.927	0.000	0.000	1.000	2.000	4.000
chorlle	0.888	0.928	0.000	0.000	1.000	2.000	4.000
gait	0.864	0.979	0.000	0.000	1.000	1.000	4.000
tandem	1.360	1.357	0.000	0.000	1.000	2.000	4.000
retropls	0.805	1.019	0.000	0.000	0.000	1.000	4.000
fascore	19.635	6.620	0.000	17.000	22.000	25.000	25.000
indepscl	82.254	19.098	5.000	70.000	85.000	100.000	100.000
drive	0.536	0.499	0.000	0.000	1.000	1.000	1.000
emplusl	0.387	0.487	0.000	0.000	0.000	1.000	1.000
emplany	0.478	0.500	0.000	0.000	0.000	1.000	1.000
volunt	0.624	0.484	0.000	0.000	1.000	1.000	1.000

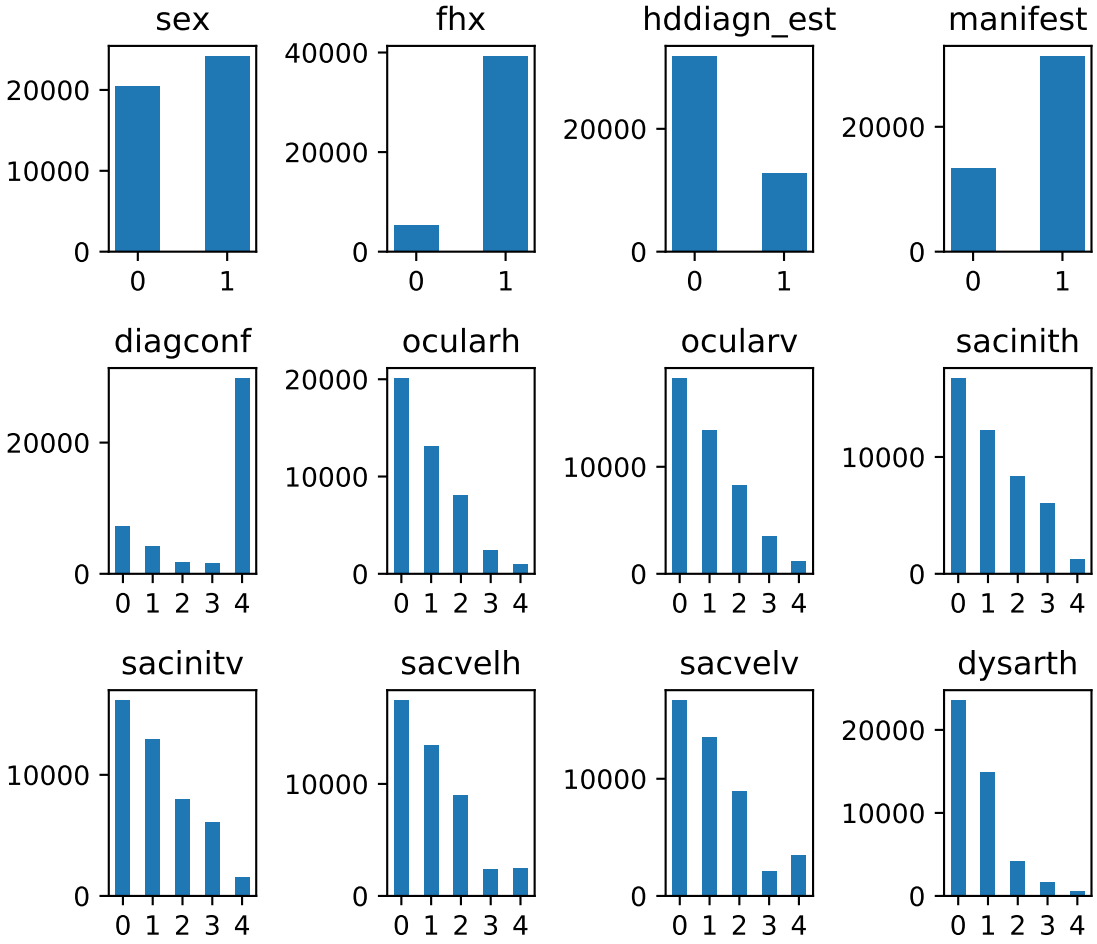
fafinan	0.571	0.495	0.000	0.000	1.000	1.000	1.000
grocery	0.753	0.431	0.000	1.000	1.000	1.000	1.000
cash	0.845	0.362	0.000	1.000	1.000	1.000	1.000
supchild	0.612	0.487	0.000	0.000	1.000	1.000	1.000
housewrk	0.676	0.468	0.000	0.000	1.000	1.000	1.000
laundry	0.769	0.422	0.000	1.000	1.000	1.000	1.000
prepmeal	0.800	0.400	0.000	1.000	1.000	1.000	1.000
telephon	0.891	0.312	0.000	1.000	1.000	1.000	1.000
ownmeds	0.838	0.369	0.000	1.000	1.000	1.000	1.000
feedself	0.923	0.266	0.000	1.000	1.000	1.000	1.000
dress	0.895	0.306	0.000	1.000	1.000	1.000	1.000
bathe	0.869	0.338	0.000	1.000	1.000	1.000	1.000
pubtrans	0.753	0.431	0.000	1.000	1.000	1.000	1.000
walknbr	0.851	0.356	0.000	1.000	1.000	1.000	1.000
walkfall	0.910	0.287	0.000	1.000	1.000	1.000	1.000
walkhelp	0.922	0.268	0.000	1.000	1.000	1.000	1.000
comb	0.928	0.259	0.000	1.000	1.000	1.000	1.000
trnchair	0.954	0.209	0.000	1.000	1.000	1.000	1.000
bed	0.956	0.206	0.000	1.000	1.000	1.000	1.000
toilet	0.946	0.226	0.000	1.000	1.000	1.000	1.000
carehome	0.952	0.215	0.000	1.000	1.000	1.000	1.000
tfcscore	9.278	3.826	0.000	6.000	10.000	13.000	13.000
occupatn	1.443	1.309	0.000	0.000	1.000	3.000	3.000
finances	2.062	1.150	0.000	1.000	3.000	3.000	3.000
chores	1.441	0.711	0.000	1.000	2.000	2.000	2.000
adl	2.483	0.840	0.000	2.000	3.000	3.000	3.000
carelevl	1.849	0.449	0.000	2.000	2.000	2.000	2.000
cogscore2	5.733	2.612	0.000	3.744	5.683	7.927	13.603
sdmt1	30.294	18.543	0.000	16.000	27.000	45.000	110.000
verfct5	14.645	7.349	0.000	9.000	14.000	20.000	75.000
scnt1	49.937	22.852	0.000	33.000	48.000	68.000	217.000
swrt1	64.503	28.826	0.000	43.000	63.000	88.000	220.000
sit1	28.333	15.408	0.000	17.000	27.000	40.000	180.000
trla1	67.885	55.285	0.000	29.000	48.000	84.000	240.000
trlb1	132.026	74.675	0.000	60.000	120.000	207.000	240.000
verflt05	26.683	15.700	0.000	14.000	24.000	38.000	104.000
tug1	11.957	9.339	-95.000	8.000	10.000	14.000	897.000
scst1	11.882	4.070	0.000	9.000	12.000	14.000	80.000
mmsetotal	25.506	4.726	0.000	24.000	27.000	29.000	30.000
depscore	4.531	5.758	0.000	0.000	2.000	7.000	48.000
irascore	2.881	4.441	0.000	0.000	1.000	4.000	32.000
psyscore	0.355	1.495	0.000	0.000	0.000	0.000	32.000

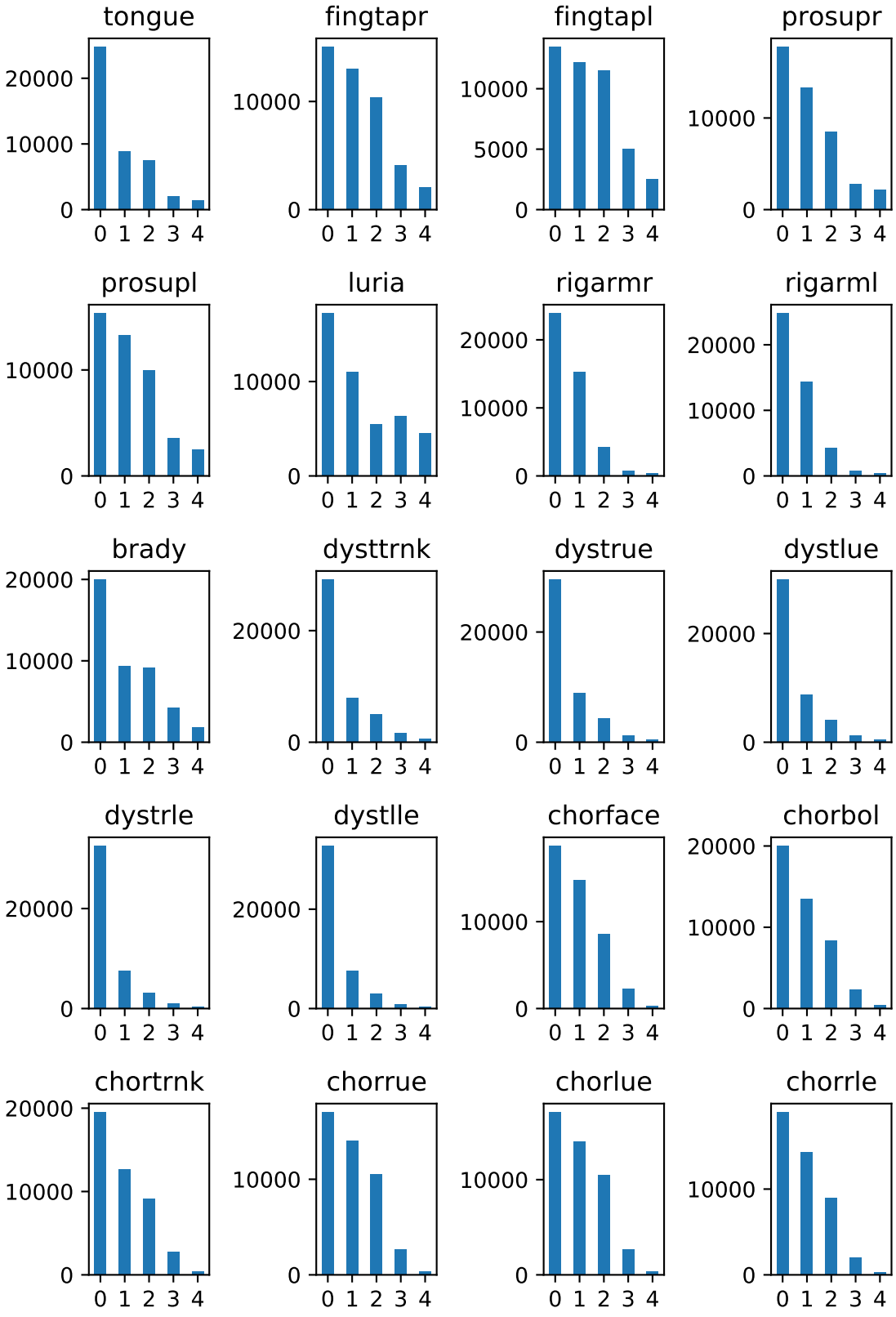
aptscore	2.722	4.072	0.000	0.000	0.000	4.000	16.000
exfscore	2.753	4.741	0.000	0.000	0.000	4.000	32.000
dbscore	0.888	2.396	0.000	0.000	0.000	0.000	16.000
pf	46.393	10.435	25.663	40.194	49.354	57.251	57.251
rp	44.489	10.004	23.335	38.309	44.426	53.579	57.080
bp	50.860	8.527	10.908	47.451	54.777	56.948	80.291
gh	48.767	9.078	24.729	45.145	48.062	57.784	63.617
vt	51.407	8.846	22.175	46.907	50.215	58.689	79.879
sf	46.139	9.844	19.138	37.657	46.916	56.175	56.175
re	42.411	12.415	7.749	32.551	44.015	55.480	67.057
mh	48.735	8.399	18.888	42.916	49.421	54.175	68.283
pcs	48.371	8.817	9.908	42.736	49.299	56.259	74.144
mcs	46.946	9.752	2.315	40.925	48.210	53.889	76.342
anxscore	5.756	3.419	-2.000	3.000	5.000	8.000	21.000
hads_depscore	5.539	3.606	0.000	3.000	5.000	8.000	21.000
irrscore	5.793	3.705	0.000	3.000	5.000	8.000	24.000
outscore	3.608	2.261	-2.000	2.000	3.000	5.000	14.000
inwscore	2.186	1.922	0.000	1.000	2.000	3.000	12.000
cUHDRS	80.047	72.548	-116.705	26.295	77.295	144.295	303.295
handed	1.119	0.380	1.000	1.000	1.000	1.000	3.000
hxsid	0.303	0.460	0.000	0.000	0.000	1.000	1.000
momhd	0.505	0.500	0.000	0.000	1.000	1.000	1.000
dadhd	0.448	0.497	0.000	0.000	0.000	1.000	1.000
sxraterm	2.452	1.881	1.000	1.000	1.000	3.000	6.000
ccmtr	0.757	0.429	0.000	1.000	1.000	1.000	1.000
sxsubjm	2.043	1.568	1.000	1.000	1.000	3.000	6.000
sxfamm	2.186	1.666	1.000	1.000	1.000	3.000	6.000
ccdep	0.692	0.462	0.000	0.000	1.000	1.000	1.000
ccirb	0.639	0.480	0.000	0.000	1.000	1.000	1.000
ccvab	0.365	0.481	0.000	0.000	0.000	1.000	1.000
ccapt	0.566	0.496	0.000	0.000	1.000	1.000	1.000
ccpob	0.499	0.500	0.000	0.000	0.000	1.000	1.000
ccpsy	0.110	0.313	0.000	0.000	0.000	0.000	1.000
cccog	0.455	0.498	0.000	0.000	0.000	1.000	1.000
hxalcab	0.087	0.282	0.000	0.000	0.000	0.000	1.000
hxtobab	0.477	0.499	0.000	0.000	0.000	1.000	1.000
hxdrugab	0.119	0.324	0.000	0.000	0.000	0.000	1.000
cafab	0.806	0.395	0.000	1.000	1.000	1.000	1.000
cafpd	0.365	0.481	0.000	0.000	0.000	1.000	1.000
drugab	0.035	0.184	0.000	0.000	0.000	0.000	1.000
jobpaid	0.018	0.134	0.000	0.000	0.000	0.000	1.000
emplnrd	0.304	0.460	0.000	0.000	0.000	1.000	1.000

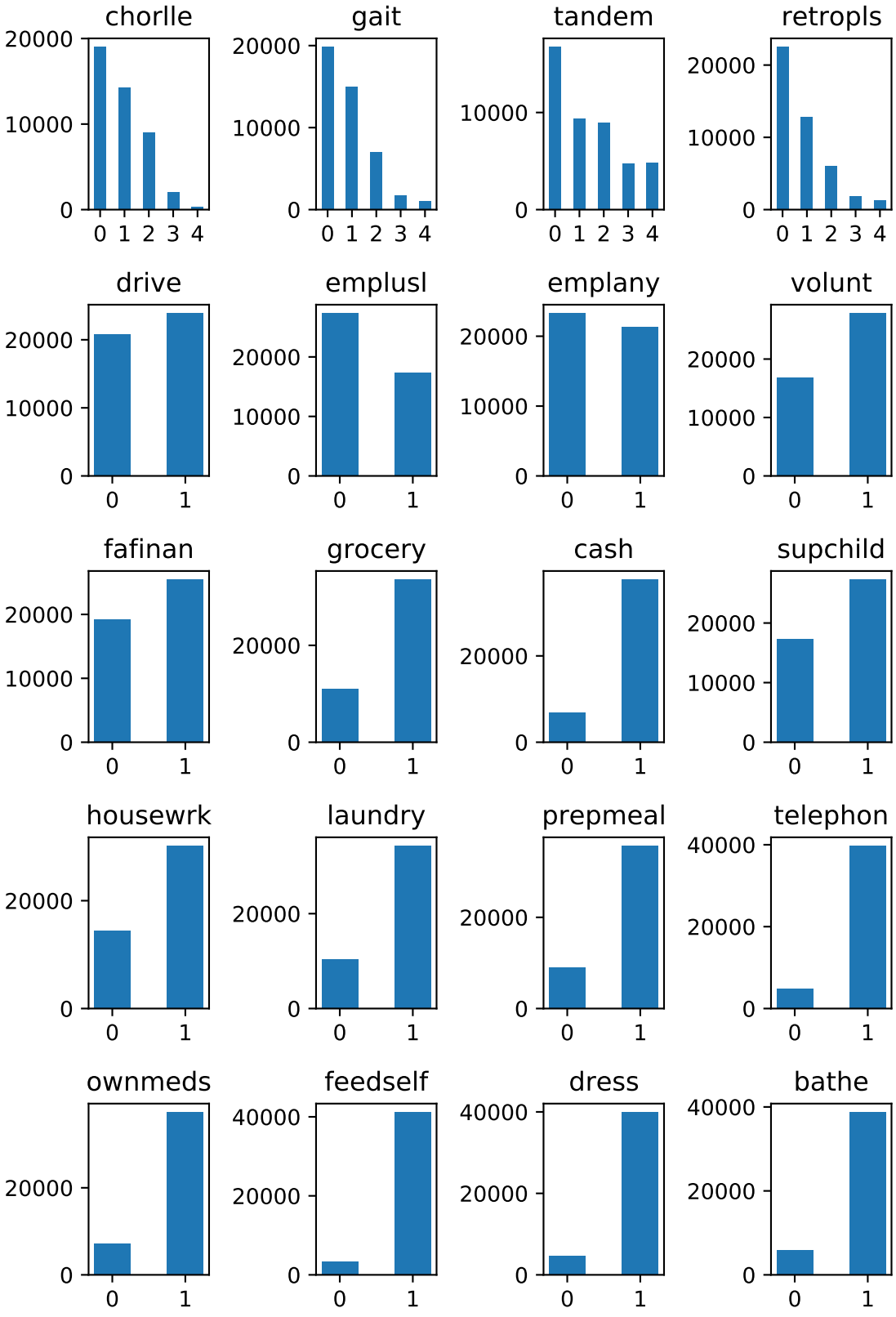
ssdb	0.669	0.471	0.000	0.000	1.000	1.000	1.000
rtrnwk	0.091	0.288	0.000	0.000	0.000	0.000	1.000

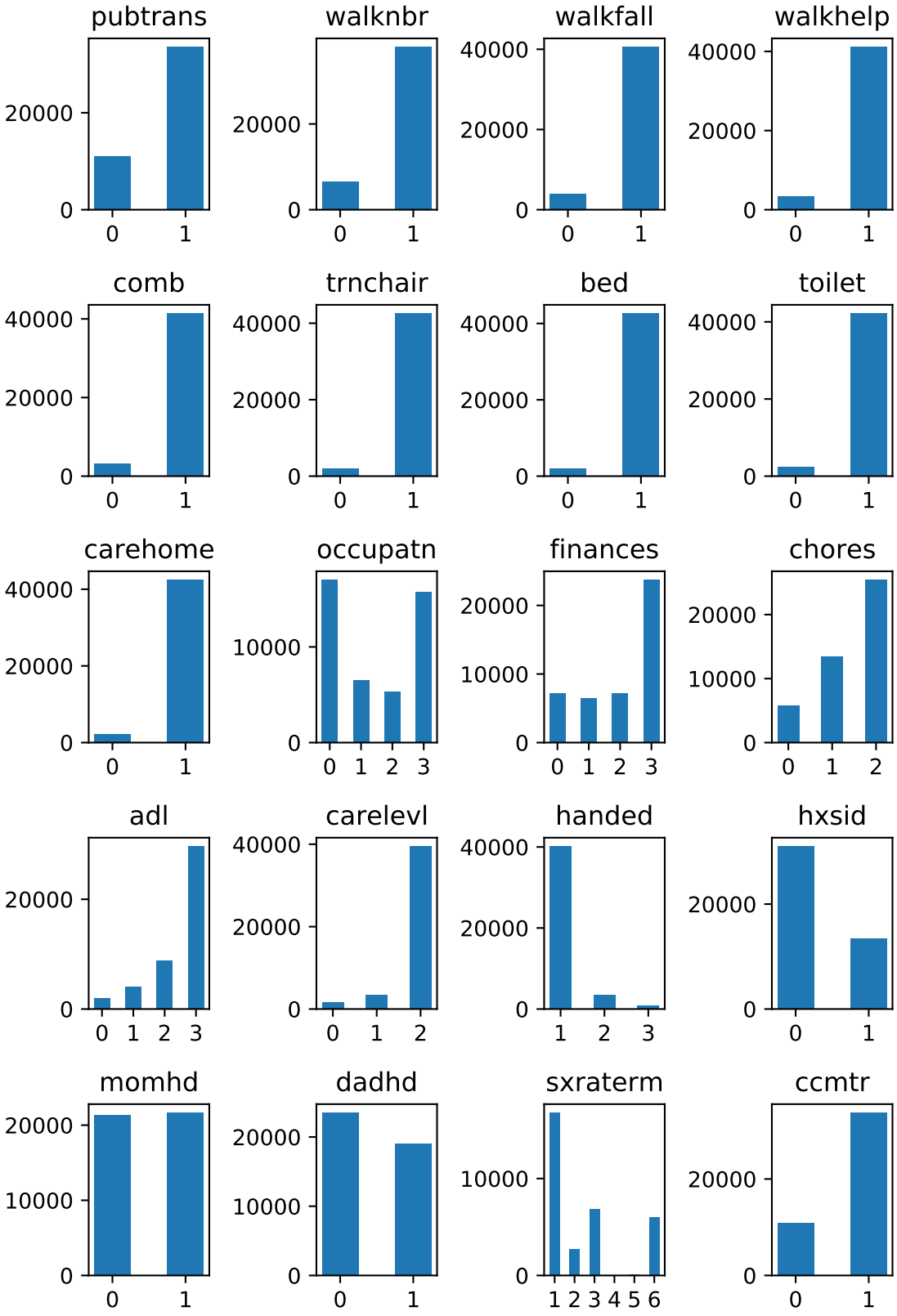
Appendix C

Categorical Variable Distribution




tongue
fingtapr
fingtapl
prosupr
prosupl
luria
rigarmr
rigarml
brady
dysttrnk
dystrue
dystlue
dystrle
dystlle
chorface
chorbol
chortrnk
chorrue
chorlue
chorrle
0 1 2 3 4


chorlle
gait
tandem
retropls
drive
emplusl
emplany
volunt
fafinan
grocery
cash
supchild
housewrk
laundry
prepmeal
telephon
ownmeds
feedself
dress
bathe


pubtrans
walknbr
walkfall
walkhelp
comb
trnchair
bed
toilet
carehome
occupatn
finances
chores
adl
carelevl
handed
hxsid
momhd
dadhd
sxraterm
ccmtr

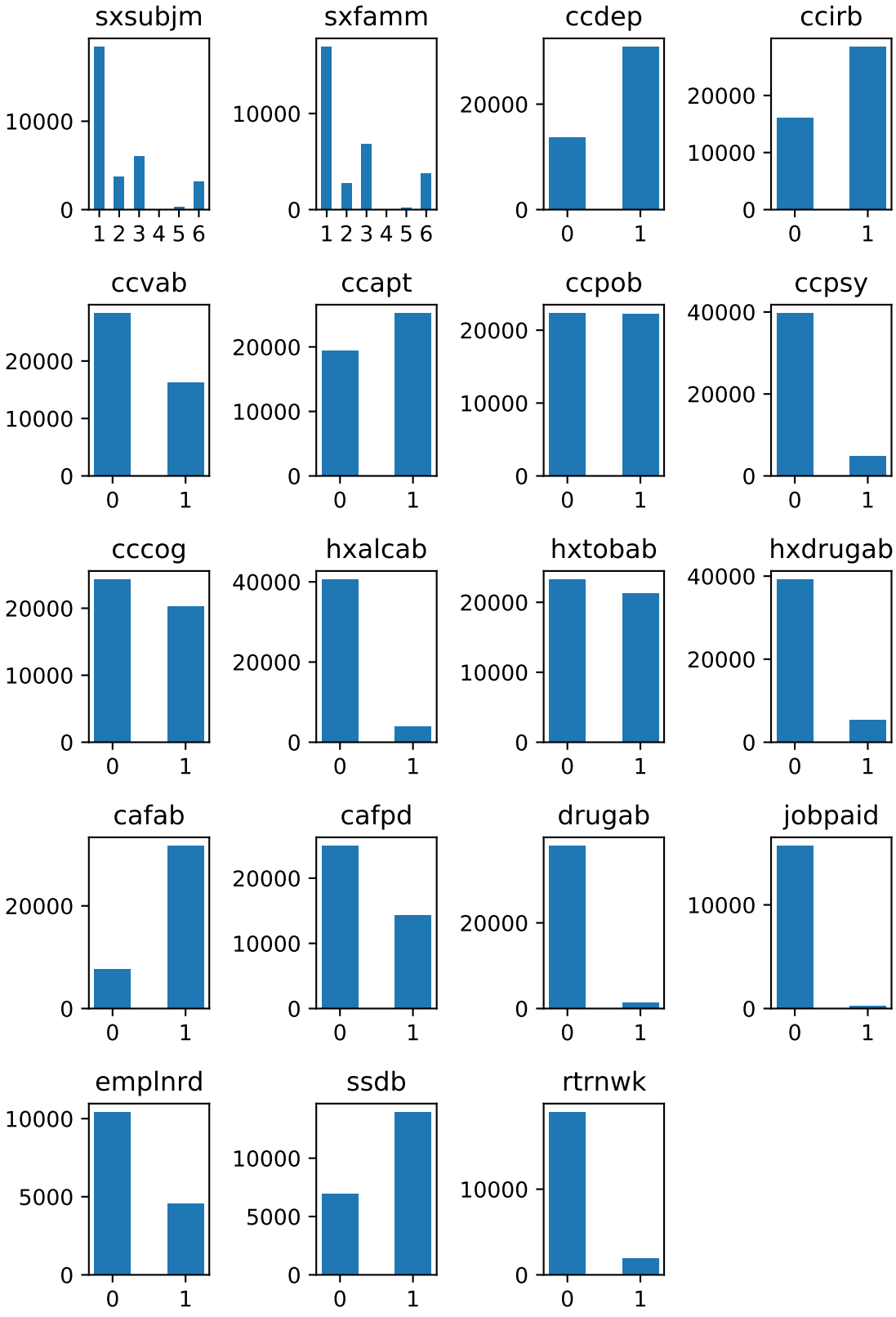
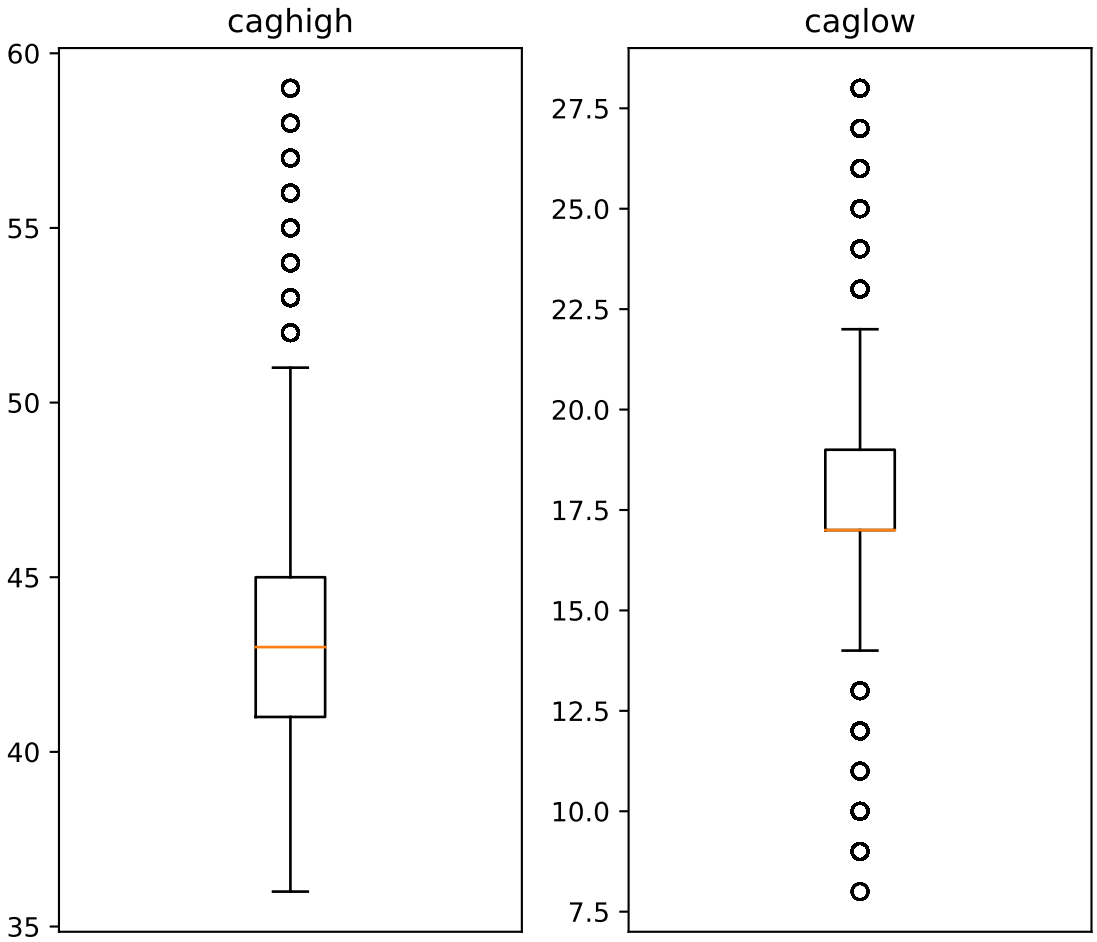
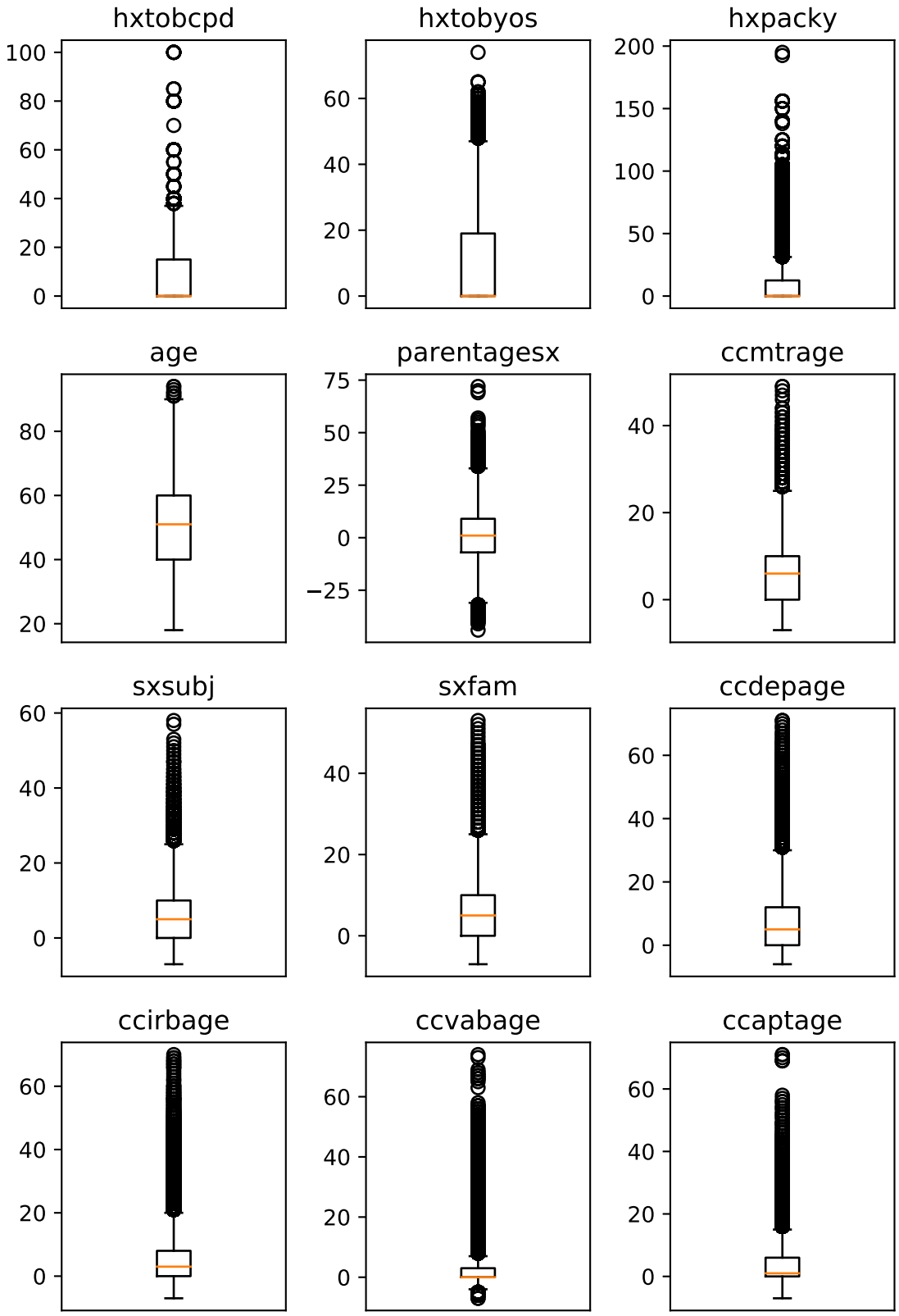


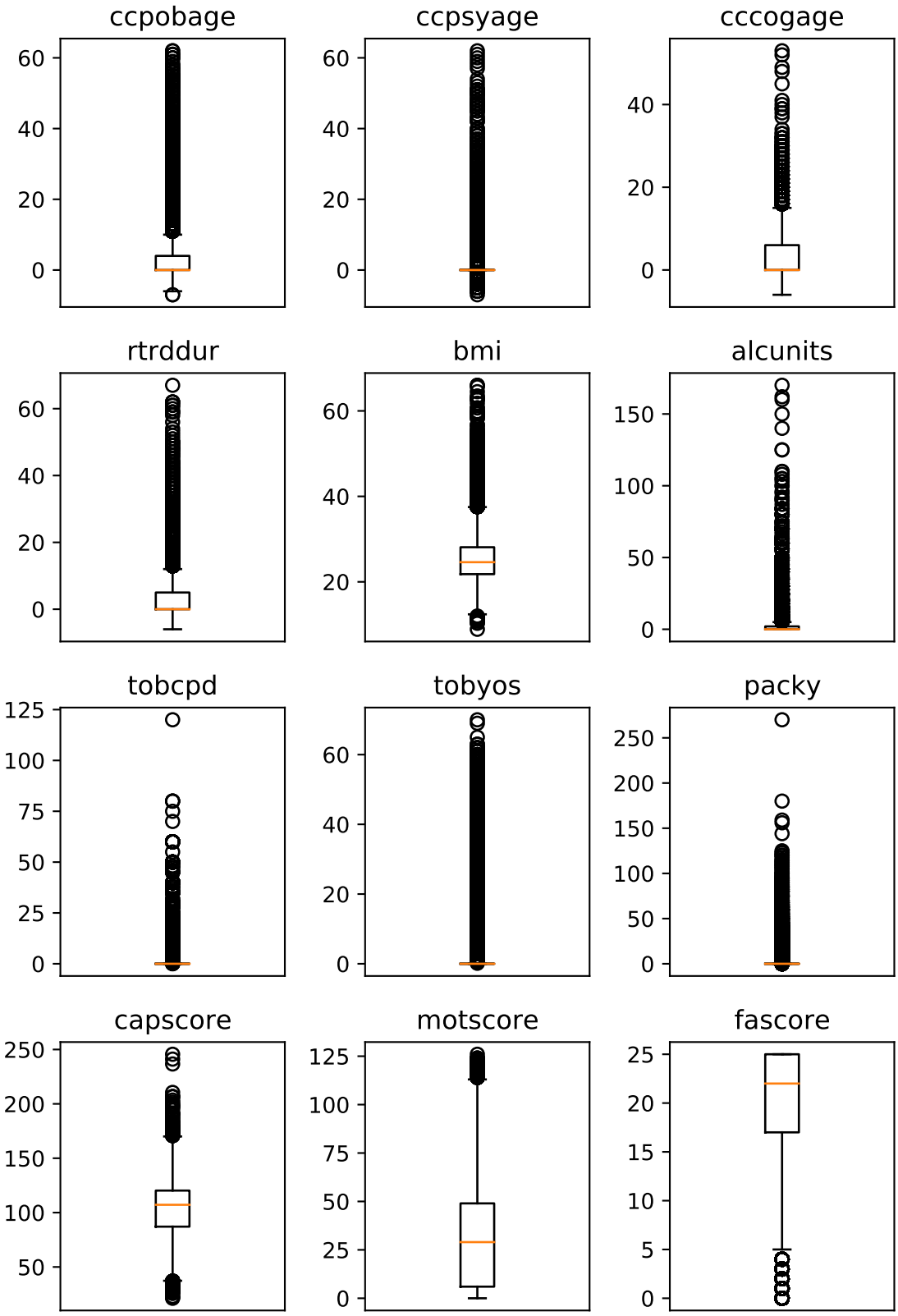
Figure C.1: *Distribution of all categorical variables in the imputed dataset.*

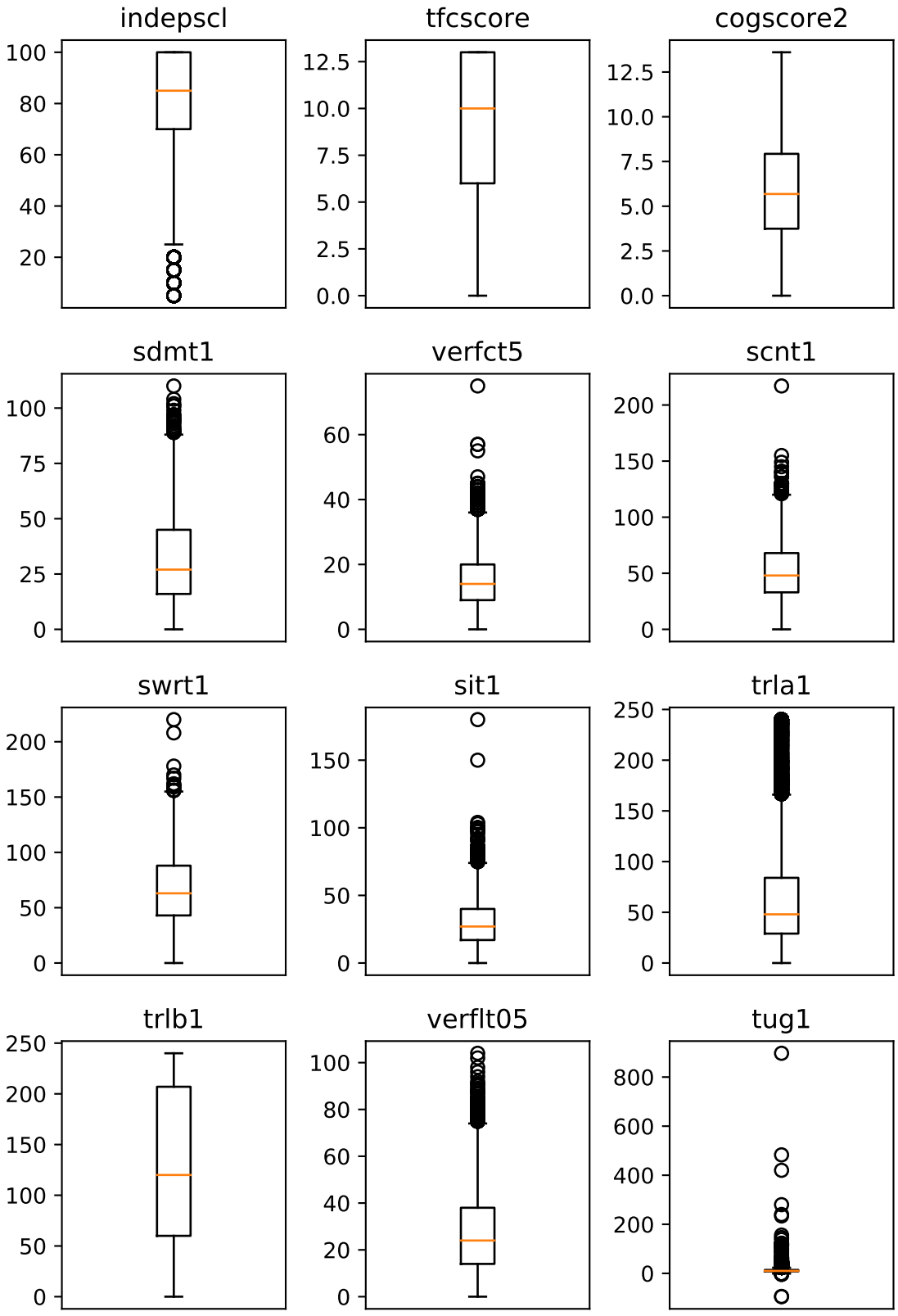
Appendix D

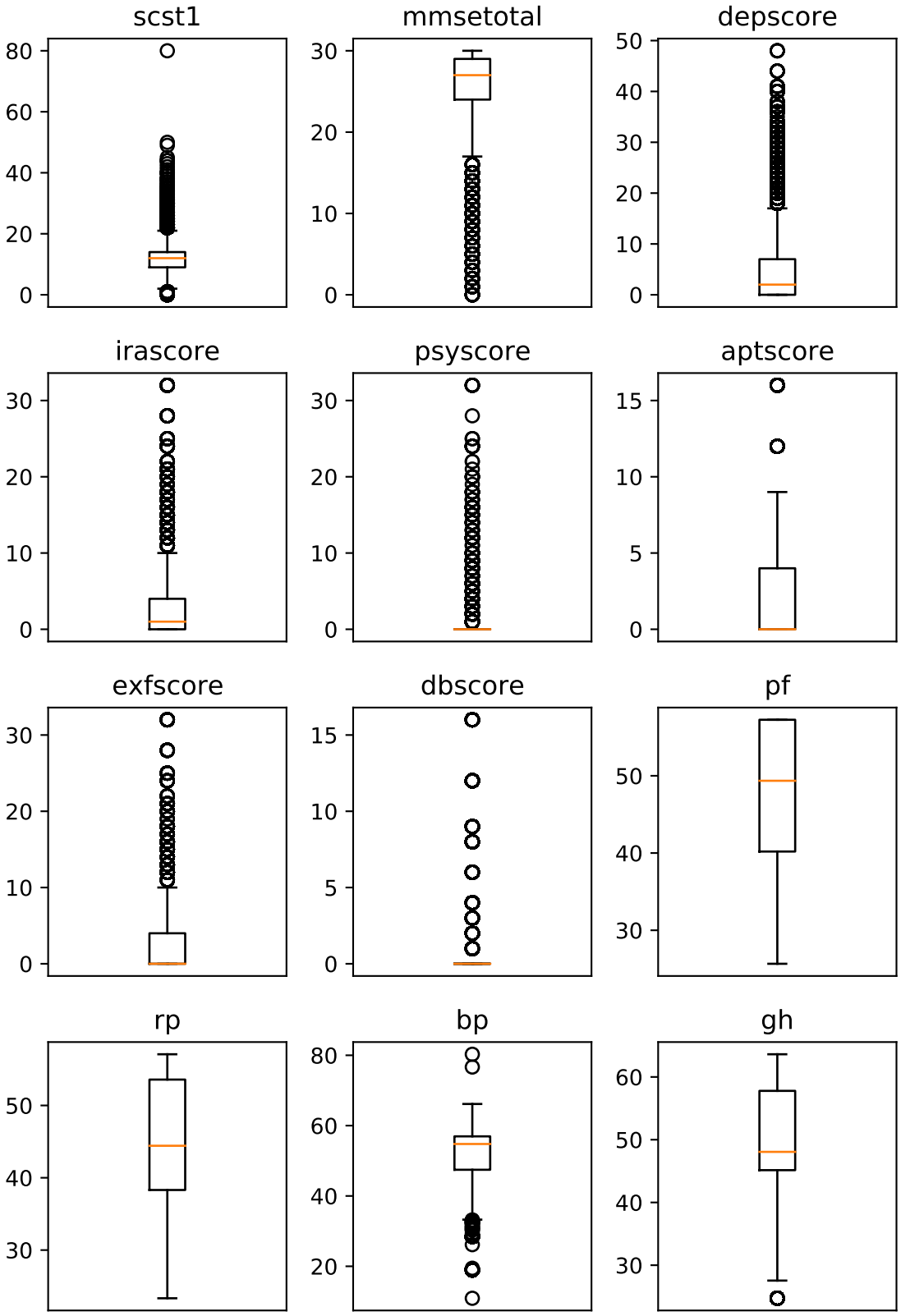
Numerical Variable Distribution




hxtobcpd
hxtobyos
hxpacky
age
parentagesx
ccmtrage
sxsubj
sxfam
ccdepage
ccirbage
ccvabage
ccaptage


ccpobage
ccpsyage
cccogage
rtrddur
bmi
alcunits
tobcpd
tobyos
packy
capscore
motscore
fascore


indepscl
tfcscore
cogscore2
sdmt1
verfct5
scnt1
swrt1
sit1
trla1
trlb1
verflt05
tug1


scst1
mmsetotal
depscore
irascore
psyscore
aptscore
exfscore
dbscore
pf
rp
bp
gh

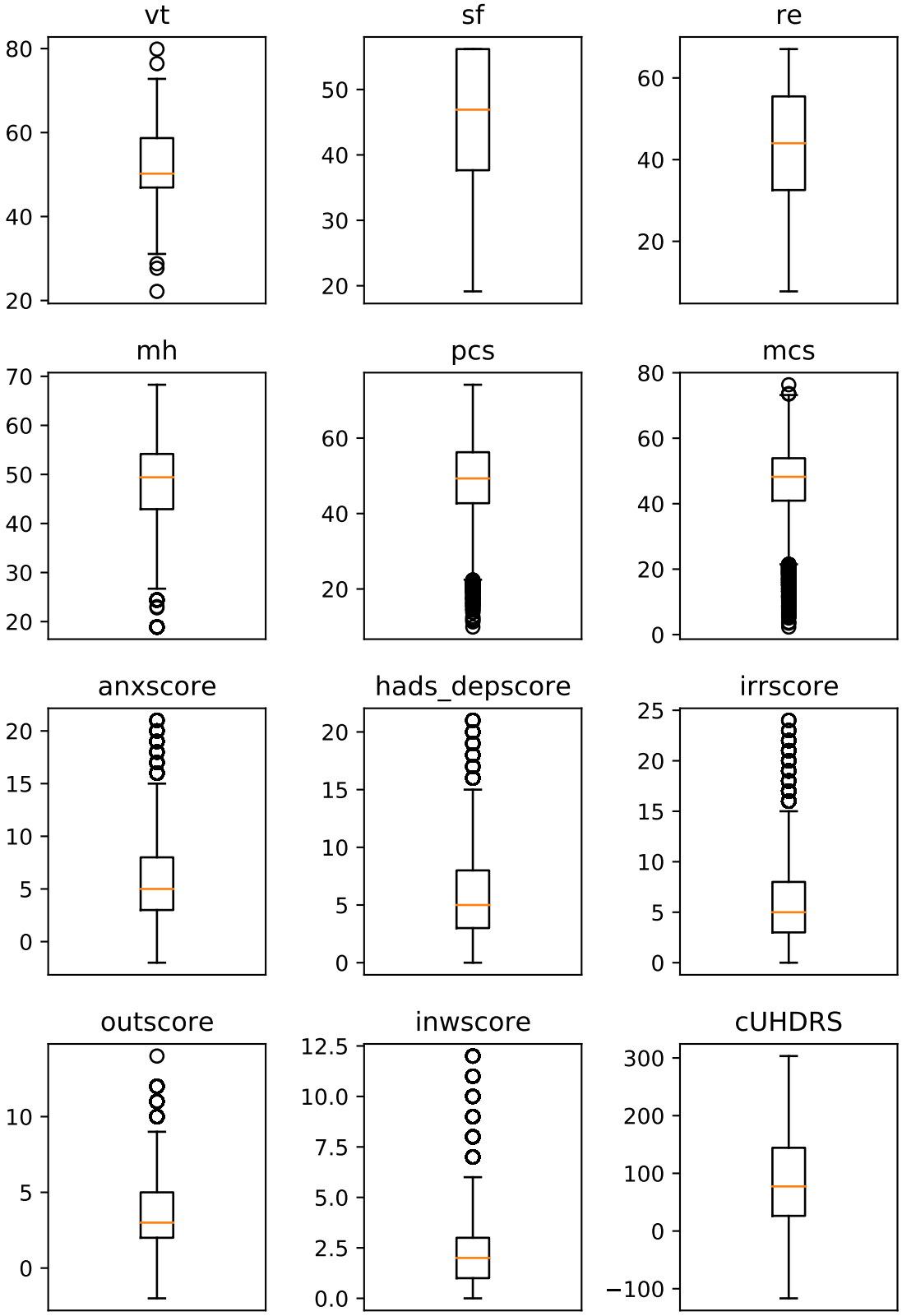


Figure D.1: *Distribution of all numerical variables in the imputed dataset.*

Appendix E

Dummy Encoded Variables

Table E.1: *List of all the variables which have been transformed using dummy encoding and the newly created variables resulting from that process.*

variable	dummies
handed	handed_1, handed_2, handed_3
hxsid	hxsid_0, hxsid_1
momhd	momhd_0, momhd_1
dadhd	dadhd_0, dadhd_1
sxraterm	sxraterm_1, sxraterm_2, sxraterm_3, sxraterm_4, sxraterm_5, sxraterm_6
ccmtr	ccmtr_0, ccmtr_1
sxsubjm	sxsubjm_1, sxsubjm_2, sxsubjm_3, sxsubjm_4, sxsubjm_5, sxsubjm_6
sxfamm	sxfamm_1, sxfamm_2, sxfamm_3, sxfamm_4, sxfamm_5, sxfamm_6
ccdep	ccdep_0, ccdep_1
ccirb	ccirb_0, ccirb_1
ccvab	ccvab_0, ccvab_1
ccapt	ccapt_0, ccapt_1
ccpob	ccpob_0, ccpob_1
ccpsy	ccpsy_0, ccpsy_1
cccog	cccog_0, cccog_1
hxalcab	hxalcab_0, hxalcab_1
hxtobab	hxtobab_0, hxtobab_1
hxdrugab	hxdrugab_0, hxdrugab_1
cafab	cafab_0, cafab_1
cafpd	cafpd_0, cafpd_1
drugab	drugab_0, drugab_1
jobpaid	jobpaid_0, jobpaid_1
emplnrd	emplnrd_0, emplnrd_1
ssdb	ssdb_0, ssdb_1
rtrnwk	rtrnwk_0, rtrnwk_1

Appendix F

Metrics drive Models

Table F.1: *All the hyperparameters and metrics of the models trained to predict the driving capability. The selected models, the models with the lowest test MAE, are set to a bold font.*

n_estimators	C	L2	hidden_size/ n_filters	training time	train AUC	train Accuracy	train F1	test AUC	test Accuracy	test F1	model
-	-	**1e-07**	**128**	**0h 3m 51s**	**0.966**	**0.899**	**0.909**	**0.961**	**0.891**	**0.903**	**SimpleRNN**
-	-	1e-05	128	0h 3m 46s	0.966	0.899	0.909	0.961	0.891	0.903	SimpleRNN
-	-	1e-07	256	0h 4m 58s	0.968	0.902	0.912	0.961	0.890	0.901	SimpleRNN
-	-	1e-05	256	0h 4m 57s	0.968	0.902	0.912	0.961	0.890	0.901	SimpleRNN
-	-	1e-07	512	0h 9m 54s	0.967	0.899	0.908	0.961	0.889	0.900	SimpleRNN
-	-	1e-05	512	0h 10m 35s	0.968	0.903	0.913	0.961	0.888	0.898	SimpleRNN
-	-	1e-03	128	0h 17m 10s	0.975	0.917	0.925	0.956	0.887	0.898	SimpleRNN
-	-	1e-03	256	0h 31m 5s	0.979	0.923	0.931	0.953	0.882	0.892	SimpleRNN
-	**0.1**	-	-	**0h 5m 42s**	**0.894**	**0.897**	**0.909**	**0.876**	**0.880**	**0.894**	**SVM**
-	1.0	-	-	0h 8m 29s	0.894	0.897	0.909	0.873	0.878	0.893	SVM
-	10.0	-	-	0h 9m 25s	0.859	0.849	0.849	0.844	0.835	0.837	SVM
1600	-	-	-	**0h 1m 22s**	**1.000**	**1.000**	**1.000**	**0.881**	**0.883**	**0.895**	**RF**
200	-	-	-	0h 0m 10s	1.000	1.000	1.000	0.880	0.882	0.894	RF
400	-	-	-	0h 0m 21s	1.000	1.000	1.000	0.880	0.882	0.894	RF
800	-	-	-	0h 0m 41s	1.000	1.000	1.000	0.879	0.881	0.894	RF
100	-	-	-	0h 0m 5s	1.000	1.000	1.000	0.876	0.878	0.891	RF

-	-	**1e-07**	**256**	**0h 4m 54s**	**0.966**	**0.898**	**0.908**	**0.960**	**0.891**	**0.902**	**NN**
-	-	1e-07	128	0h 3m 41s	0.966	0.898	0.907	0.960	0.890	0.900	NN
-	-	1e-05	256	0h 4m 37s	0.965	0.896	0.905	0.960	0.888	0.898	NN
-	-	1e-05	128	0h 3m 22s	0.964	0.896	0.906	0.960	0.888	0.898	NN
-	-	1e-03	128	0h 15m 54s	0.978	0.923	0.930	0.958	0.885	0.896	NN
-	-	1e-03	256	0h 18m 15s	0.980	0.927	0.934	0.957	0.884	0.895	NN
-	-	**1e-05**	**128**	**0h 11m 49s**	**0.966**	**0.898**	**0.908**	**0.961**	**0.892**	**0.903**	**LSTM**
-	-	1e-07	128	0h 11m 51s	0.967	0.899	0.908	0.961	0.892	0.903	LSTM
-	-	1e-05	256	0h 23m 1s	0.966	0.899	0.909	0.961	0.892	0.903	LSTM
-	-	1e-07	256	0h 20m 57s	0.966	0.896	0.905	0.961	0.892	0.903	LSTM
-	-	1e-03	128	0h 39m 31s	0.967	0.900	0.910	0.960	0.892	0.905	LSTM
-	-	1e-03	256	1h 26m 20s	0.966	0.898	0.907	0.959	0.890	0.901	LSTM
-	-	**1e-07**	**256**	**0h 15m 54s**	**0.965**	**0.897**	**0.907**	**0.962**	**0.893**	**0.906**	**GRU**
-	-	1e-05	256	0h 16m 51s	0.966	0.896	0.904	0.962	0.891	0.902	GRU
-	-	1e-05	128	0h 7m 51s	0.965	0.896	0.905	0.962	0.888	0.896	GRU
-	-	1e-07	128	0h 8m 14s	0.965	0.896	0.905	0.962	0.888	0.896	GRU
-	-	1e-03	256	1h 11m 28s	0.967	0.899	0.908	0.960	0.890	0.903	GRU
-	-	1e-03	128	0h 41m 39s	0.967	0.901	0.910	0.960	0.883	0.891	GRU
-	-	**1e-07**	**256**	**0h 14m 16s**	**0.966**	**0.898**	**0.907**	**0.961**	**0.891**	**0.902**	**CNN**
-	-	1e-05	256	0h 14m 19s	0.966	0.898	0.907	0.961	0.890	0.899	CNN
-	-	1e-05	128	0h 8m 46s	0.965	0.895	0.904	0.960	0.891	0.904	CNN
-	-	1e-03	256	0h 35m 2s	0.981	0.929	0.935	0.960	0.890	0.903	CNN
-	-	1e-07	128	0h 8m 47s	0.965	0.894	0.903	0.960	0.890	0.902	CNN
-	-	1e-03	128	0h 28m 57s	0.979	0.925	0.932	0.960	0.891	0.901	CNN

Appendix G

Metrics cUHDRS Models

Table G.1: *All the hyperparameters and metrics of the models trained to predict the cUHDRS. The selected models, the models with the lowest test MAE, are set to a bold font.*

n_estimators	C	Epsilon	L2	hidden_size/ n_filters	training time	train MAE	train RMSE	train Max AE	train R^2	test MAE	test RMSE	test Max AE	test R^2	model
-	-	-	**1e-03**	**256**	**1h 24m 49s**	**12.367**	**16.371**	**120.273**	**0.943**	**12.524**	**16.485**	**116.075**	**0.943**	**CNN**
-	-	-	1e-03	128	1h 45m 50s	12.397	16.411	119.582	0.943	12.556	16.532	112.698	0.943	CNN
-	-	-	1e-07	128	0h 9m 53s	11.567	15.409	123.794	0.950	13.105	17.261	105.207	0.938	CNN
-	-	-	1e-07	256	0h 9m 47s	11.404	15.200	126.556	0.951	13.217	17.374	126.261	0.937	CNN
-	-	-	1e-05	128	0h 17m 12s	10.309	13.912	115.055	0.959	13.242	17.451	105.053	0.936	CNN
-	-	-	1e-05	256	0h 17m 45s	8.966	12.581	103.829	0.966	13.753	17.951	126.240	0.933	CNN
-	-	-	-	-	**0h 0m 1s**	**14.711**	**19.896**	**146.426**	**0.916**	**14.439**	**19.263**	**123.670**	**0.922**	**Constant**
-	-	-	**1e-07**	**256**	**0h 15m 17s**	**11.945**	**15.999**	**123.006**	**0.946**	**12.460**	**16.508**	**124.821**	**0.943**	**GRU**
-	-	-	1e-07	128	0h 13m 47s	11.967	16.005	119.976	0.946	12.485	16.513	123.741	0.943	GRU
-	-	-	1e-03	128	3h 2m 46s	12.566	16.708	122.206	0.941	12.704	16.786	123.731	0.941	GRU
-	-	-	1e-03	256	4h 26m 32s	12.546	16.671	122.399	0.941	12.760	16.831	122.518	0.941	GRU
-	-	-	1e-05	128	0h 48m 3s	11.024	14.691	106.926	0.954	12.911	17.003	114.913	0.940	GRU
-	-	-	1e-05	256	1h 1m 0s	10.660	14.268	103.822	0.957	13.093	17.243	109.848	0.938	GRU
-	-	-	**1e-07**	**256**	**0h 18m 13s**	**12.023**	**16.135**	**121.637**	**0.945**	**12.455**	**16.512**	**121.348**	**0.943**	**LSTM**
-	-	-	1e-07	128	0h 16m 13s	12.015	16.102	127.082	0.945	12.456	16.570	126.145	0.943	LSTM
-	-	-	1e-05	128	0h 50m 17s	11.286	15.129	117.465	0.951	12.844	16.988	117.960	0.940	LSTM

-	-	-	1e-03	128	4h 54m 39s	12.429	16.540	121.877	0.942	12.904	16.944	103.656	0.940	LSTM
-	-	-	1e-05	256	1h 12m 7s	11.129	14.944	111.684	0.953	12.965	17.083	112.433	0.939	LSTM
-	-	-	1e-03	256	6h 42m 3s	12.474	16.597	121.981	0.942	12.968	17.051	107.840	0.939	LSTM
-	-	-	**1e-03**	**256**	**0h 52m 42s**	**12.799**	**16.828**	**123.259**	**0.940**	**12.927**	**17.041**	**112.566**	**0.939**	**NN**
-	-	-	1e-03	128	0h 42m 38s	12.813	16.849	123.538	0.940	12.930	17.031	111.269	0.939	NN
-	-	-	1e-07	128	0h 3m 56s	11.618	15.439	113.736	0.949	13.606	17.985	117.506	0.932	NN
-	-	-	1e-07	256	0h 3m 22s	11.688	15.550	114.912	0.949	13.660	17.950	117.469	0.933	NN
-	-	-	1e-05	128	0h 10m 31s	9.008	12.432	104.721	0.967	14.085	18.562	123.995	0.928	NN
-	-	-	1e-05	256	0h 13m 32s	6.503	9.364	87.933	0.981	14.942	19.688	160.343	0.919	NN
1600	-	-	-	-	**0h 2m 44s**	**4.381**	**5.897**	**49.016**	**0.993**	**13.005**	**17.232**	**101.307**	**0.938**	**RF**
800	-	-	-	-	0h 1m 22s	4.386	5.902	49.229	0.993	13.026	17.254	100.904	0.938	RF
200	-	-	-	-	0h 0m 22s	4.413	5.956	54.806	0.992	13.050	17.287	98.639	0.938	RF
400	-	-	-	-	0h 0m 42s	4.391	5.918	50.878	0.993	13.050	17.289	100.811	0.938	RF
100	-	-	-	-	0h 0m 12s	4.460	6.035	53.871	0.992	13.057	17.317	98.164	0.937	RF
-	**0.1**	**1e-02**	-	-	**0h 3m 11s**	**13.195**	**17.847**	**130.324**	**0.932**	**14.001**	**18.512**	**96.630**	**0.928**	**SVM**
-	0.1	1e-01	-	-	0h 2m 32s	13.829	17.932	124.306	0.932	14.484	18.831	96.334	0.926	SVM
-	10.0	1e-02	-	-	0h 5m 14s	14.629	18.861	118.194	0.925	15.328	19.869	99.807	0.917	SVM
-	1.0	1e-01	-	-	0h 4m 23s	15.227	19.616	135.995	0.918	15.637	20.336	104.745	0.914	SVM
-	1.0	1e-02	-	-	0h 5m 10s	17.442	22.155	132.948	0.896	18.218	22.942	113.447	0.890	SVM
-	10.0	1e-01	-	-	0h 4m 29s	17.693	22.110	110.675	0.896	18.440	23.329	107.611	0.886	SVM
-	0.1	1e+00	-	-	0h 0m 0s	59.212	70.623	186.000	-0.058	60.031	71.251	180.064	-0.061	SVM
-	10.0	1e+00	-	-	0h 0m 0s	59.212	70.623	186.000	-0.058	60.031	71.251	180.064	-0.061	SVM
-	1.0	1e+00	-	-	0h 0m 0s	59.212	70.623	186.000	-0.058	60.031	71.251	180.064	-0.061	SVM
-	-	-	**1e-03**	**128**	**0h 33m 24s**	**12.561**	**16.719**	**122.475**	**0.941**	**12.712**	**16.830**	**130.127**	**0.941**	**SimpleRNN**
-	-	-	1e-03	256	0h 31m 25s	12.582	16.748	121.864	0.941	12.746	16.870	129.699	0.941	SimpleRNN
-	-	-	1e-03	512	0h 51m 32s	12.578	16.752	121.586	0.940	12.758	16.890	131.863	0.940	SimpleRNN
-	-	-	1e-07	128	0h 6m 15s	11.985	15.902	110.974	0.946	13.707	18.025	126.805	0.932	SimpleRNN
-	-	-	1e-05	128	0h 14m 35s	10.362	13.937	98.167	0.959	14.006	18.570	118.698	0.928	SimpleRNN
-	-	-	1e-07	256	0h 5m 55s	11.472	15.335	116.103	0.950	14.181	18.605	115.610	0.928	SimpleRNN
-	-	-	1e-07	512	0h 8m 27s	10.484	14.318	104.004	0.957	14.392	18.920	101.261	0.925	SimpleRNN
-	-	-	1e-05	256	1h 0m 7s	5.885	9.729	116.651	0.980	16.998	22.395	127.649	0.895	SimpleRNN
-	-	-	1e-05	512	2h 42m 47s	4.865	8.561	100.305	0.984	17.989	23.675	142.830	0.883	SimpleRNN

Appendix H

Metrics TMS Models

Table H.1: *All the hyperparameters and metrics of the models trained to predict the TMS. The selected models, the models with the lowest test MAE, are set to a bold font.*

n_estimators	C	Epsilon	L2	hidden_size/ n_filters	training time	train MAE	train RMSE	train Max AE	train R^2	test MAE	test RMSE	test Max AE	test R^2	model
-	-	-	**1e-03**	**128**	**0h 38m 31s**	**5.191**	**7.336**	**67.492**	**0.921**	**5.371**	**7.600**	**53.756**	**0.918**	**CNN**
-	-	-	1e-03	256	1h 13m 2s	5.191	7.325	67.374	0.921	5.378	7.596	53.482	0.918	CNN
-	-	-	1e-07	128	0h 4m 36s	5.116	7.181	64.118	0.924	5.623	7.845	57.701	0.913	CNN
-	-	-	1e-05	128	0h 5m 25s	4.979	6.975	60.839	0.929	5.638	7.866	56.274	0.912	CNN
-	-	-	1e-07	256	0h 7m 26s	5.021	7.034	63.530	0.928	5.670	7.937	57.502	0.911	CNN
-	-	-	1e-05	256	0h 8m 34s	4.785	6.739	60.520	0.933	5.686	7.954	58.835	0.910	CNN
-	-	-	-	-	**0h 0m 1s**	**6.065**	**8.947**	**70.683**	**0.883**	**6.002**	**8.912**	**64.873**	**0.888**	**Constant**
-	-	-	**1e-07**	**256**	**0h 10m 49s**	**5.161**	**7.297**	**63.339**	**0.922**	**5.402**	**7.640**	**54.157**	**0.917**	**GRU**
-	-	-	1e-03	256	2h 11m 19s	5.221	7.409	69.202	0.920	5.410	7.687	55.350	0.916	GRU
-	-	-	1e-03	128	1h 9m 8s	5.231	7.406	68.743	0.920	5.413	7.672	55.096	0.917	GRU
-	-	-	1e-07	128	0h 5m 25s	5.197	7.337	65.461	0.921	5.420	7.642	53.836	0.917	GRU
-	-	-	1e-05	128	0h 11m 44s	4.990	7.022	58.852	0.928	5.490	7.715	54.318	0.916	GRU
-	-	-	1e-05	256	0h 25m 17s	4.887	6.864	52.556	0.931	5.525	7.772	54.210	0.915	GRU
-	-	-	**1e-07**	**256**	**0h 14m 49s**	**5.179**	**7.348**	**66.248**	**0.921**	**5.418**	**7.653**	**54.731**	**0.917**	**LSTM**
-	-	-	1e-07	128	0h 7m 3s	5.193	7.373	65.628	0.920	5.457	7.698	54.463	0.916	LSTM
-	-	-	1e-03	128	1h 50m 45s	5.200	7.334	68.851	0.921	5.490	7.751	55.776	0.915	LSTM

-	-	-	1e-03	256	4h 19m 54s	5.164	7.294	68.882	0.922	5.514	7.798	55.708	0.914	LSTM
-	-	-	1e-05	256	0h 33m 7s	4.959	7.009	61.326	0.928	5.540	7.776	55.864	0.914	LSTM
-	-	-	1e-05	128	0h 16m 56s	4.955	6.994	60.235	0.928	5.552	7.797	54.133	0.914	LSTM
-	-	-	**1e-03**	**128**	**0h 26m 17s**	**5.224**	**7.372**	**69.501**	**0.920**	**5.466**	**7.701**	**53.270**	**0.916**	**NN**
-	-	-	1e-03	256	0h 34m 22s	5.246	7.369	69.552	0.920	5.485	7.700	53.289	0.916	NN
-	-	-	1e-07	128	0h 1m 55s	5.042	7.029	60.499	0.928	5.800	8.091	56.739	0.907	NN
-	-	-	1e-05	128	0h 2m 44s	4.708	6.586	51.130	0.936	5.809	8.102	58.695	0.907	NN
-	-	-	1e-07	256	0h 2m 32s	5.049	7.034	62.678	0.928	5.852	8.149	54.925	0.906	NN
-	-	-	1e-05	256	0h 4m 5s	4.253	6.119	42.480	0.945	5.900	8.283	57.041	0.903	NN
1600	-	-	-	-	**0h 10m 9s**	**1.878**	**2.701**	**25.809**	**0.989**	**5.734**	**8.101**	**55.213**	**0.907**	**RF**
800	-	-	-	-	0h 5m 5s	1.879	2.703	25.792	0.989	5.747	8.112	55.417	0.907	RF
400	-	-	-	-	0h 2m 33s	1.882	2.708	24.107	0.989	5.751	8.119	55.500	0.907	RF
200	-	-	-	-	0h 1m 19s	1.893	2.732	24.540	0.989	5.771	8.141	56.115	0.906	RF
100	-	-	-	-	0h 0m 40s	1.904	2.752	20.604	0.989	5.790	8.160	55.839	0.906	RF
-	0.1	1e+00	-	-	0h 0m 1s	33.617	38.911	61.000	-1.218	33.758	39.073	61.000	-1.161	SVM
-	1.0	1e+00	-	-	0h 0m 1s	33.617	38.911	61.000	-1.218	33.758	39.073	61.000	-1.161	SVM
-	10.0	1e+00	-	-	0h 0m 1s	33.617	38.911	61.000	-1.218	33.758	39.073	61.000	-1.161	SVM
-	**0.1**	**1e-02**	-	-	**0h 3m 27s**	**5.657**	**8.032**	**72.931**	**0.905**	**6.108**	**8.478**	**54.572**	**0.898**	**SVM**
-	1.0	1e-02	-	-	0h 6m 3s	5.782	8.101	74.274	0.904	6.232	8.597	53.748	0.895	SVM
-	0.1	1e-01	-	-	0h 3m 1s	5.806	7.982	74.625	0.907	6.268	8.541	52.844	0.897	SVM
-	1.0	1e-01	-	-	0h 5m 38s	5.938	8.035	72.370	0.905	6.413	8.659	51.478	0.894	SVM
-	10.0	1e-02	-	-	0h 6m 15s	6.522	8.539	65.889	0.893	7.053	9.216	49.456	0.880	SVM
-	10.0	1e-01	-	-	0h 5m 50s	7.953	9.886	58.251	0.857	8.444	10.571	45.453	0.842	SVM
-	-	-	**1e-03**	**128**	**0h 13m 41s**	**5.239**	**7.434**	**69.411**	**0.919**	**5.410**	**7.685**	**57.019**	**0.916**	**SimpleRNN**
-	-	-	1e-03	256	0h 21m 35s	5.230	7.428	69.661	0.919	5.427	7.708	56.549	0.916	SimpleRNN
-	-	-	1e-05	128	0h 2m 47s	5.134	7.173	57.268	0.925	5.745	8.093	58.204	0.907	SimpleRNN
-	-	-	1e-07	128	0h 2m 29s	5.243	7.300	59.840	0.922	5.768	8.095	58.277	0.907	SimpleRNN
-	-	-	1e-05	256	0h 3m 58s	4.921	6.846	50.438	0.931	5.986	8.375	59.011	0.901	SimpleRNN
-	-	-	1e-07	256	0h 3m 48s	4.977	6.923	52.293	0.930	6.000	8.387	59.067	0.900	SimpleRNN

Appendix I

Metrics TFC Models

Table I.1: *All the hyperparameters and metrics of the models trained to predict the TFC. The selected models, the models with the lowest test MAE, are set to a bold font.*

n_estimators	C	Epsilon	L2	hidden_size/ n_filters	training time	train MAE	train RMSE	train Max AE	train R^2	test MAE	test RMSE	test Max AE	test R^2	model
-	-	-	**1e-03**	**128**	**0h 36m 4s**	**0.768**	**1.148**	**10.152**	**0.917**	**0.866**	**1.295**	**8.549**	**0.898**	**CNN**
-	-	-	1e-03	256	0h 32m 15s	0.738	1.103	10.110	0.924	0.871	1.302	8.910	0.897	CNN
-	-	-	1e-05	256	0h 7m 28s	0.832	1.227	10.736	0.906	0.897	1.319	8.734	0.895	CNN
-	-	-	1e-05	128	0h 7m 7s	0.851	1.242	10.338	0.903	0.904	1.324	8.767	0.894	CNN
-	-	-	1e-07	256	0h 7m 6s	0.856	1.243	10.709	0.903	0.909	1.321	8.903	0.894	CNN
-	-	-	1e-07	128	0h 7m 23s	0.850	1.235	10.259	0.905	0.910	1.325	8.944	0.894	CNN
-	-	-	-	-	**0h 0m 1s**	**0.940**	**1.610**	**12.000**	**0.838**	**0.896**	**1.559**	**10.000**	**0.853**	**Constant**
-	-	-	**1e-07**	**256**	**0h 11m 53s**	**0.856**	**1.264**	**10.464**	**0.900**	**0.873**	**1.292**	**8.714**	**0.899**	**GRU**
-	-	-	1e-07	128	0h 10m 32s	0.854	1.263	10.554	0.900	0.873	1.294	8.742	0.899	GRU
-	-	-	1e-03	128	1h 23m 3s	0.867	1.275	10.437	0.898	0.877	1.295	8.351	0.899	GRU
-	-	-	1e-05	128	0h 15m 36s	0.834	1.234	10.556	0.905	0.879	1.299	8.841	0.898	GRU
-	-	-	1e-05	256	0h 20m 39s	0.826	1.219	10.390	0.907	0.883	1.301	8.770	0.897	GRU
-	-	-	1e-03	256	1h 31m 23s	0.872	1.276	10.473	0.898	0.884	1.300	8.378	0.898	GRU
-	-	-	**1e-07**	**128**	**0h 11m 34s**	**0.860**	**1.275**	**10.472**	**0.898**	**0.869**	**1.296**	**8.527**	**0.898**	**LSTM**
-	-	-	1e-07	256	0h 13m 3s	0.862	1.277	10.490	0.898	0.873	1.296	8.882	0.898	LSTM
-	-	-	1e-05	128	0h 19m 25s	0.835	1.240	10.461	0.904	0.876	1.304	8.668	0.897	LSTM

-	-	-	1e-05	256	0h 25m 38s	0.828	1.227	10.446	0.906	0.884	1.307	8.790	0.897	LSTM
-	-	-	1e-03	128	2h 40m 55s	0.853	1.258	10.408	0.901	0.888	1.310	8.374	0.896	LSTM
-	-	-	1e-03	256	3h 2m 10s	0.853	1.257	10.455	0.901	0.894	1.316	8.334	0.895	LSTM
-	-	-	**1e-03**	**256**	**0h 23m 33s**	**0.770**	**1.150**	**10.356**	**0.917**	**0.872**	**1.307**	**8.920**	**0.897**	**NN**
-	-	-	1e-03	128	0h 26m 8s	0.789	1.166	10.359	0.915	0.882	1.309	8.742	0.896	NN
-	-	-	1e-05	256	0h 3m 59s	0.793	1.170	10.686	0.914	0.914	1.336	8.357	0.892	NN
-	-	-	1e-07	256	0h 3m 26s	0.832	1.212	10.596	0.908	0.920	1.335	8.499	0.892	NN
-	-	-	1e-07	128	0h 3m 55s	0.829	1.209	10.123	0.909	0.923	1.341	8.662	0.891	NN
-	-	-	1e-05	128	0h 3m 57s	0.829	1.206	10.178	0.909	0.924	1.338	8.665	0.892	NN
1600	-	-	-	-	**0h 3m 30s**	**0.308**	**0.468**	**3.815**	**0.986**	**0.940**	**1.379**	**8.474**	**0.885**	**RF**
800	-	-	-	-	0h 1m 46s	0.308	0.469	3.715	0.986	0.941	1.379	8.451	0.885	RF
400	-	-	-	-	0h 0m 53s	0.308	0.470	3.750	0.986	0.941	1.380	8.570	0.885	RF
200	-	-	-	-	0h 0m 27s	0.310	0.474	3.720	0.986	0.942	1.382	8.675	0.884	RF
100	-	-	-	-	0h 0m 14s	0.312	0.478	3.420	0.986	0.945	1.386	8.720	0.884	RF
-	**0.1**	**1e-02**	-	-	**0h 2m 50s**	**0.977**	**1.464**	**10.536**	**0.866**	**0.998**	**1.470**	**10.535**	**0.869**	**SVM**
-	1.0	1e-02	-	-	0h 5m 34s	0.995	1.480	10.657	0.863	1.023	1.495	10.375	0.865	SVM
-	0.1	1e-01	-	-	0h 2m 28s	1.006	1.436	10.371	0.871	1.034	1.465	9.957	0.870	SVM
-	1.0	1e-01	-	-	0h 5m 30s	1.098	1.477	9.977	0.863	1.133	1.524	10.457	0.860	SVM
-	10.0	1e-01	-	-	0h 5m 56s	1.368	1.785	11.247	0.800	1.405	1.826	9.398	0.798	SVM
-	10.0	1e-02	-	-	0h 6m 5s	1.569	1.942	9.048	0.764	1.606	1.987	10.759	0.761	SVM
-	0.1	1e+00	-	-	0h 0m 0s	4.118	4.669	6.500	-0.365	4.158	4.700	6.500	-0.337	SVM
-	1.0	1e+00	-	-	0h 0m 0s	4.118	4.669	6.500	-0.365	4.158	4.700	6.500	-0.337	SVM
-	10.0	1e+00	-	-	0h 0m 0s	4.118	4.669	6.500	-0.365	4.158	4.700	6.500	-0.337	SVM
-	-	-	**1e-03**	**128**	**0h 35m 42s**	**0.862**	**1.268**	**10.426**	**0.899**	**0.880**	**1.301**	**8.414**	**0.898**	**SimpleRNN**
-	-	-	1e-03	256	0h 41m 54s	0.865	1.270	10.371	0.899	0.884	1.304	8.364	0.897	SimpleRNN
-	-	-	1e-07	128	0h 5m 6s	0.857	1.249	10.584	0.902	0.928	1.353	9.002	0.889	SimpleRNN
-	-	-	1e-05	128	0h 5m 20s	0.851	1.239	10.500	0.904	0.928	1.352	9.020	0.889	SimpleRNN
-	-	-	1e-05	256	0h 5m 14s	0.825	1.206	10.394	0.909	0.953	1.374	8.688	0.886	SimpleRNN
-	-	-	1e-07	256	0h 4m 51s	0.844	1.225	10.455	0.906	0.957	1.375	8.724	0.886	SimpleRNN

Appendix J

Metrics SDMT Models

Table J.1: *All the hyperparameters and metrics of the models trained to predict SDMT. The selected models, the models with the lowest test MAE, are set to a bold font.*

n_estimators	C	Epsilon	L2	hidden_size/ n_filters	training time	train MAE	train RMSE	train Max AE	train R^2	test MAE	test RMSE	test Max AE	test R^2	model
-	-	-	**1e-03**	**128**	**0h 59m 45s**	**4.097**	**5.551**	**50.948**	**0.913**	**4.253**	**5.773**	**55.688**	**0.905**	**CNN**
-	-	-	1e-03	256	1h 21m 40s	4.089	5.540	50.823	0.914	4.256	5.772	55.933	0.905	CNN
-	-	-	1e-07	128	0h 10m 23s	3.742	5.088	41.514	0.927	4.505	6.051	52.893	0.895	CNN
-	-	-	1e-05	128	0h 14m 52s	3.497	4.818	38.908	0.935	4.506	6.063	53.174	0.895	CNN
-	-	-	1e-07	256	0h 10m 28s	3.658	4.996	39.632	0.930	4.530	6.073	55.142	0.894	CNN
-	-	-	1e-05	256	0h 11m 50s	3.470	4.805	37.302	0.935	4.540	6.095	54.549	0.894	CNN
-	-	-	-	-	**0h 0m 1s**	**4.712**	**6.504**	**63.000**	**0.881**	**4.607**	**6.373**	**49.000**	**0.884**	**Constant**
-	-	-	**1e-07**	**256**	**0h 11m 47s**	**3.987**	**5.439**	**46.941**	**0.917**	**4.149**	**5.658**	**54.264**	**0.908**	**GRU**
-	-	-	1e-07	128	0h 17m 32s	3.966	5.407	46.490	0.918	4.189	5.694	55.953	0.907	GRU
-	-	-	1e-05	128	0h 48m 5s	3.708	5.079	41.462	0.927	4.282	5.805	55.252	0.903	GRU
-	-	-	1e-05	256	0h 46m 15s	3.592	4.933	41.765	0.932	4.347	5.882	56.152	0.901	GRU
-	-	-	1e-03	128	2h 22m 59s	4.278	5.762	49.748	0.907	4.383	5.917	55.739	0.900	GRU
-	-	-	1e-03	256	2h 51m 42s	4.278	5.765	50.463	0.907	4.396	5.934	56.289	0.899	GRU
-	-	-	**1e-07**	**128**	**0h 16m 12s**	**4.004**	**5.443**	**44.381**	**0.917**	**4.191**	**5.715**	**57.294**	**0.906**	**LSTM**
-	-	-	1e-07	256	0h 26m 42s	3.949	5.385	45.082	0.918	4.192	5.703	55.120	0.907	LSTM
-	-	-	1e-05	128	1h 0m 54s	3.748	5.128	38.748	0.926	4.296	5.842	57.035	0.902	LSTM

-	-	-	1e-05	256	1h 23m 55s	3.692	5.066	39.148	0.928	4.320	5.862	55.468	0.902	LSTM
-	-	-	1e-03	128	5h 58m 7s	4.234	5.721	47.630	0.908	4.431	5.967	54.518	0.898	LSTM
-	-	-	1e-03	256	6h 57m 15s	4.258	5.744	47.335	0.907	4.440	5.988	55.114	0.897	LSTM
-	-	-	**1e-03**	**128**	**0h 38m 4s**	**4.270**	**5.753**	**49.966**	**0.907**	**4.438**	**5.953**	**55.176**	**0.898**	**NN**
-	-	-	1e-03	256	0h 44m 22s	4.272	5.756	50.407	0.907	4.441	5.952	54.762	0.899	NN
-	-	-	1e-07	256	0h 4m 1s	3.748	5.127	43.324	0.926	4.655	6.195	55.470	0.890	NN
-	-	-	1e-07	128	0h 4m 46s	3.843	5.222	42.288	0.923	4.657	6.212	55.009	0.889	NN
-	-	-	1e-05	128	0h 8m 26s	3.377	4.700	35.831	0.938	4.667	6.224	54.876	0.889	NN
-	-	-	1e-05	256	0h 10m 25s	2.741	3.971	35.794	0.956	4.842	6.420	58.594	0.882	NN
1600	-	-	-	-	**0h 2m 45s**	**1.451**	**2.002**	**18.934**	**0.989**	**4.292**	**5.850**	**58.404**	**0.902**	**RF**
800	-	-	-	-	0h 1m 26s	1.453	2.004	18.785	0.989	4.293	5.853	58.324	0.902	RF
200	-	-	-	-	0h 0m 24s	1.462	2.022	19.640	0.989	4.301	5.860	58.220	0.902	RF
400	-	-	-	-	0h 0m 44s	1.455	2.010	18.310	0.989	4.301	5.858	58.122	0.902	RF
100	-	-	-	-	0h 0m 13s	1.475	2.047	18.520	0.988	4.310	5.876	57.980	0.901	RF
-	1.0	1e+00	-	-	0h 0m 1s	24.990	29.435	55.000	-1.435	24.750	29.135	55.000	-1.432	SVM
-	10.0	1e+00	-	-	0h 0m 0s	24.990	29.435	55.000	-1.435	24.750	29.135	55.000	-1.432	SVM
-	0.1	1e+00	-	-	0h 0m 0s	24.990	29.435	55.000	-1.435	24.750	29.135	55.000	-1.432	SVM
-	**0.1**	**1e-02**	-	-	**0h 4m 14s**	**4.346**	**5.952**	**51.573**	**0.900**	**4.561**	**6.067**	**53.957**	**0.895**	**SVM**
-	0.1	1e-01	-	-	0h 3m 40s	4.457	5.935	49.007	0.901	4.616	6.115	56.220	0.893	SVM
-	1.0	1e-02	-	-	0h 7m 25s	4.526	6.114	51.358	0.895	4.754	6.295	54.696	0.886	SVM
-	1.0	1e-01	-	-	0h 6m 30s	4.842	6.334	48.637	0.887	4.980	6.527	55.784	0.878	SVM
-	10.0	1e-02	-	-	0h 7m 33s	4.892	6.400	53.648	0.885	5.153	6.708	57.611	0.871	SVM
-	10.0	1e-01	-	-	0h 6m 40s	5.687	7.191	52.866	0.855	5.974	7.546	57.244	0.837	SVM
-	-	-	**1e-03**	**128**	**0h 41m 5s**	**4.271**	**5.760**	**50.613**	**0.907**	**4.362**	**5.886**	**55.609**	**0.901**	**SimpleRNN**
-	-	-	1e-03	256	0h 53m 16s	4.270	5.762	51.208	0.907	4.371	5.900	55.718	0.900	SimpleRNN
-	-	-	1e-05	128	0h 11m 29s	3.774	5.119	38.467	0.926	4.616	6.117	53.136	0.893	SimpleRNN
-	-	-	1e-07	128	0h 7m 57s	4.016	5.386	37.335	0.918	4.630	6.107	55.114	0.893	SimpleRNN
-	-	-	1e-07	256	0h 7m 5s	3.915	5.302	41.865	0.921	4.772	6.304	56.993	0.886	SimpleRNN
-	-	-	1e-05	256	0h 11m 21s	3.437	4.805	36.218	0.935	4.811	6.360	58.973	0.884	SimpleRNN

Appendix K

Metrics SWRT Models

Table K.1: *All the hyperparameters and metrics of the models trained to predict SWRT. The selected models, the models with the lowest test MAE, are set to a bold font.*

n_estimators	C	Epsilon	L2	hidden_size/ n_filters	training time	train MAE	train RMSE	train Max AE	train R^2	test MAE	test RMSE	test Max AE	test R^2	model
-	-	-	**1e-03**	**256**	**1h 13m 34s**	**7.661**	**10.292**	**93.537**	**0.875**	**7.881**	**10.350**	**74.272**	**0.877**	**CNN**
-	-	-	1e-03	128	0h 52m 24s	7.711	10.345	94.480	0.874	7.896	10.355	74.192	0.877	CNN
-	-	-	1e-07	128	0h 6m 7s	7.447	10.002	85.899	0.882	8.304	10.860	74.216	0.864	CNN
-	-	-	1e-05	128	0h 7m 50s	7.123	9.651	85.473	0.890	8.320	10.867	75.569	0.864	CNN
-	-	-	1e-07	256	0h 8m 33s	7.377	9.912	87.299	0.884	8.401	10.930	83.196	0.863	CNN
-	-	-	1e-05	256	0h 10m 55s	6.810	9.298	85.314	0.898	8.410	10.993	85.587	0.861	CNN
-	-	-	-	-	**0h 0m 1s**	**8.975**	**12.481**	**106.000**	**0.816**	**8.796**	**11.961**	**78.000**	**0.835**	**Constant**
-	-	-	**1e-07**	**256**	**0h 10m 40s**	**7.638**	**10.369**	**91.931**	**0.873**	**7.839**	**10.367**	**75.963**	**0.876**	**GRU**
-	-	-	1e-07	128	0h 6m 29s	7.625	10.346	87.628	0.874	7.861	10.406	75.884	0.875	GRU
-	-	-	1e-05	128	0h 16m 24s	7.287	9.900	84.964	0.884	7.963	10.550	77.586	0.872	GRU
-	-	-	1e-03	128	1h 14m 23s	7.921	10.622	83.435	0.867	7.999	10.532	77.795	0.872	GRU
-	-	-	1e-03	256	2h 48m 26s	7.903	10.597	83.262	0.868	8.048	10.561	76.143	0.872	GRU
-	-	-	1e-05	256	0h 35m 6s	7.106	9.648	84.298	0.890	8.079	10.642	75.065	0.870	GRU
-	-	-	**1e-07**	**256**	**0h 15m 47s**	**7.649**	**10.362**	**87.075**	**0.873**	**7.865**	**10.400**	**76.156**	**0.876**	**LSTM**
-	-	-	1e-07	128	0h 9m 5s	7.591	10.288	85.164	0.875	7.886	10.427	76.643	0.875	LSTM
-	-	-	1e-05	256	0h 58m 0s	7.181	9.775	84.730	0.887	8.044	10.619	77.740	0.870	LSTM

-	-	-	1e-05	128	0h 26m 38s	7.241	9.825	83.914	0.886	8.055	10.637	78.743	0.870	LSTM
-	-	-	1e-03	128	2h 50m 1s	7.829	10.501	83.250	0.870	8.102	10.630	76.378	0.870	LSTM
-	-	-	1e-03	256	6h 14m 6s	7.818	10.484	83.341	0.870	8.134	10.654	72.785	0.869	LSTM
-	-	-	**1e-03**	**256**	**0h 37m 38s**	**8.020**	**10.666**	**91.082**	**0.866**	**8.152**	**10.675**	**69.926**	**0.869**	**NN**
-	-	-	1e-03	128	0h 25m 2s	8.009	10.659	90.509	0.866	8.158	10.668	69.804	0.869	NN
-	-	-	1e-05	128	0h 3m 19s	6.955	9.464	84.913	0.894	8.546	11.180	65.454	0.856	NN
-	-	-	1e-07	128	0h 2m 10s	7.664	10.224	84.329	0.877	8.579	11.210	65.719	0.855	NN
-	-	-	1e-07	256	0h 3m 20s	7.240	9.804	85.840	0.887	8.596	11.185	73.511	0.856	NN
-	-	-	1e-05	256	0h 6m 28s	5.899	8.346	85.291	0.918	8.826	11.484	70.461	0.848	NN
1600	-	-	-	-	**0h 8m 49s**	**2.757**	**3.771**	**38.378**	**0.983**	**8.069**	**10.706**	**68.617**	**0.868**	**RF**
400	-	-	-	-	0h 2m 13s	2.767	3.782	33.312	0.983	8.078	10.722	69.152	0.868	RF
800	-	-	-	-	0h 4m 24s	2.761	3.773	35.754	0.983	8.080	10.715	68.865	0.868	RF
100	-	-	-	-	0h 0m 35s	2.809	3.856	33.780	0.982	8.088	10.741	71.250	0.867	RF
200	-	-	-	-	0h 1m 7s	2.780	3.807	34.915	0.983	8.090	10.733	69.265	0.867	RF
-	0.1	1e+00	-	-	0h 0m 1s	30.911	37.450	89.000	-0.654	30.800	37.514	89.000	-0.620	SVM
-	10.0	1e+00	-	-	0h 0m 1s	30.911	37.450	89.000	-0.654	30.800	37.514	89.000	-0.620	SVM
-	1.0	1e+00	-	-	0h 0m 1s	30.911	37.450	89.000	-0.654	30.800	37.514	89.000	-0.620	SVM
-	**0.1**	**1e-01**	-	-	**0h 3m 8s**	**8.391**	**11.155**	**100.500**	**0.853**	**8.503**	**11.188**	**65.778**	**0.856**	**SVM**
-	0.1	1e-02	-	-	0h 3m 16s	8.209	11.221	97.292	0.852	8.568	11.289	61.903	0.853	SVM
-	1.0	1e-01	-	-	0h 5m 26s	8.524	11.222	102.405	0.851	8.616	11.324	64.380	0.852	SVM
-	10.0	1e-02	-	-	0h 6m 5s	9.191	12.039	98.635	0.829	9.379	12.213	73.165	0.828	SVM
-	1.0	1e-02	-	-	0h 5m 52s	9.196	12.265	97.895	0.823	9.568	12.372	59.536	0.824	SVM
-	10.0	1e-01	-	-	0h 5m 35s	9.887	12.668	99.242	0.811	9.906	12.741	76.366	0.813	SVM
-	-	-	**1e-03**	**128**	**0h 20m 38s**	**7.901**	**10.614**	**83.825**	**0.867**	**8.021**	**10.545**	**77.152**	**0.872**	**SimpleRNN**
-	-	-	1e-03	256	0h 36m 17s	7.910	10.621	84.210	0.867	8.026	10.561	77.473	0.872	SimpleRNN
-	-	-	1e-05	128	0h 5m 25s	7.296	9.811	84.923	0.886	8.597	11.210	71.769	0.855	SimpleRNN
-	-	-	1e-07	128	0h 4m 56s	7.536	10.081	84.828	0.880	8.608	11.202	77.769	0.856	SimpleRNN
-	-	-	1e-07	128	0h 4m 59s	7.536	10.081	84.828	0.880	8.608	11.202	77.769	0.856	SimpleRNN
-	-	-	1e-05	256	0h 4m 53s	7.158	9.731	82.413	0.888	8.803	11.501	62.812	0.848	SimpleRNN
-	-	-	1e-07	256	0h 4m 19s	7.393	9.983	83.168	0.882	8.829	11.523	64.051	0.847	SimpleRNN