Final Report

Depth of Anesthesia

Regression Analysis

Christian Willig

Contents

1. Abstract 2
2. Report body 3
2.1 Research 3
2.2 Data Analysis 4
2.2.1 Statistics 4
2.2.2 Distributions 6
2.2.3 Relationships 7
2.2.4 Outliers 9
2.2.5 Normmality test 10
2.2.6 Data analysis conclusion 10
2.3 Methodology 11
2.4 Evaluation methods 12
2.5 Machine Learning methods application 12
2.5.1 Software 12
2.5.2 Feature Selection 13
2.5.3 Data Transformations 14
2.5.4 Screening process 14
2.5.5 Model selection 14
2.6 Results 15
2.7 Discussion 19
3. References 21
4. Appendix 22
4.1 Reading file 22
4.1 Feature selection code 22
4.1 Screening process code 25
4.2 Random Forest hyperparameter tunning 26
4.3 KNN hyperparameter tunning 27

1. Abstract

This report proposes two new depth of anaesthesia (DoA) indexes for the course CSC6003 - Machine Learning in semester 2 - 2023. We were given 15 cases of BIS index data separated in training and test datasets. The objective of this report is to create a new BIS index that could replace the current one being used at the Well Hospital by using supervised machine learning algorithms. After analyzing the data and feature selection, a screening process was performed in order to assess the performance of four potential models on the training data set. Two models were selected based on their performance: K-nearest neighbor and Random forest. Both models performed moderately well on the test dataset with an overall RMSE of 10.1 and 9.4 respectively. The results were evaluated using the correlation coefficient and Bland-Altman methods.

All code was written in R using several packages that helped on developing this report. The appendix contains all the pieces of code used to implement this work.

2. Report body

The report is structured in the following way. We start with a summary of the studies that have used machine learning algorithms to create alternative BIS indexes. Next, we dive into the data analysis of the dataset which provides an understanding of the data we are using here. The rest of the sections present the methodology followed to build the new index and the results. Finally, all code used in the report are presented in the appendix section.

2.1 Research

The Depth of Anaesthesia (DoA) is a topic that has been researched for some time now and it's very relevant as it measures the level of consciousness a person has while on an anaesthetic drug.

It can be monitored via EEG signals because it reflects the electrical activity of the brain, so in an awake state, these signals show high-frequency activity and when the body interacts with anaesthesia, the signals slow down and become more synchronised. In a deep state the EEG signals show low-frequency activity. Controlling the amount of anaesthesia is the main challenge for anaesthetists due to the effect that a high/low dosage of anaesthesia has in each patient. These effects can lead to coma and postoperative complications or the patient could suffer from pain and waking up in the middle of surgery.

Many commercial models have been developed for monitoring DoA and are based on EEG signals.

Currently the BIS index is the standard device for prediction of DoA, but it has limitations: * It doesn't work well with all drugs * It's not accurate across patients * It's being delayed

Studies have shown that EEG is a non-stationary (Elbert et al., 1994; Natarajan et al., 2004) and it exhibits non-linear behaviors (Lalitha et al., 2007, Elbert et al., 1994; Natarajan et al., 2004) which has derived in feature extractions of EEG signals based on non-linear dynamics.

Machine learning algorithms can be used to identify patterns in EEG signals that are associated with different DoA levels. For example,

- Gu et al., 2019 and Shalbaf et al., 2013 used EEG signals and an **artificial neural network** to monitor DoA.
- Peker et al, 2015 used **Random Forest** for determining the DoA level.
- Liu et al., 2019 used **CNN** to model patient's DoA.
- Yi et al., 2022 used **several machine learning models** to assess DoA (Robust Linear, Gaussian SVM, Squared Exponential Gaussian, etc).

As stated in the introduction of this piece of work, the problem to solve here is a regression problem. Some studies have taken a classification approach where instead of predicting the index they predict the state of DoA, and that converts the problem into a classification problem.

The three main characteristics of the problem that I have identified as relevant for the regression model are:

- **Accuracy**: The model has to be accurate enough in predicing the DoA state. This is important given the consequences of giving the wrong doses of anaesthesia to patients.
- **Speed**: The model has to be quick to make prediction as it is expected to work in real time, like during surgery for example. And also, it needs to be quick during training as it is expected to be retrained for different type of patients and type of drugs.
- **Robustness**: The model should be robust to noise.

There is no conclusion about which ML model is the best for DoA monitoring but there is one certainty about machine learning having the potential to help and improve the anaesthetist work by providing accuracy and robustness in predicting DoA.

2.2 Data Analysis

To begin with my analysis I imported the file “Project data set 1 (for reports 1 and 3) .xlsx” into my workspace. Then I read every sheet named “Train X” and appended the sets together into one. Following the construction of one dataset I checked the structure of the dataset. I have been provided a training dataset consisting of 27,452 observations. The first 10 rows of the dataset are shown in table 1.

Table 1. First 10 Rows of Dataset.

BIS	x1	x2	x3	x4	x5	x6	x7	x8
90.2	0.6892	3.6439	1.7892	1.7787	0.6565	0.3859	1.0310	1.7903
90.2	0.7048	3.6370	1.7809	1.7786	0.6611	0.3889	1.0311	1.7904
90.2	0.7110	3.6455	1.7897	1.7785	0.6649	0.3836	1.0312	1.7904
89.7	0.7159	3.6550	1.7812	1.7785	0.6642	0.3848	1.0314	1.7905
89.4	0.7165	3.6563	1.7846	1.7789	0.6656	0.3842	1.0316	1.7907

2.2.1 Statistics

After understanding the structure of the dataset we run statistical calculations as shown in table 2 in order to assess central tendency and dispersion of the data.

Table 2. Univariate Profile for Simulated Data

vars	n	na	minus	mean	median	var	sd	min	max	skew	kur
BIS	27,452	0	0	43.23	40.20	289.24	17.01	15.80	97.70	1.13	1.13
x1	27,452	0	0	0.78	0.76	0.09	0.31	0.04	1.95	0.20	-0.12
x2	27,452	0	0	0.97	0.56	0.94	0.97	0.19	6.81	2.65	8.15
x3	27,452	0	0	1.79	1.79	0.00	0.00	1.78	1.79	0.01	-1.20
x4	27,452	0	0	1.79	1.79	0.00	0.01	1.54	1.79	-13.81	284.26
x5	27,452	0	0	0.67	0.68	0.00	0.05	0.53	0.84	-0.20	-0.44
x6	27,452	0	0	0.39	0.39	0.00	0.01	0.37	0.45	2.19	4.96
x7	27,452	0	0	1.01	1.02	0.00	0.03	0.88	1.14	0.73	4.25
x8	27,452	0	0	1.78	1.79	0.00	0.00	1.77	1.79	-0.61	-0.02

At a high level We can see that the dataset doesn’t contain any missing values or negative values at all. This is a good scenario and has saved us to deal with missing values.

The mean value for the BIS index is 43.23 with a standard deviation of 17.01 which indicates the BIS observations are mostly around to the mean value.

Several variables have a variance close to zero which is interesting and deserve to have a closer look. Having variables with zero variance could indicate that there is not much information that is being provided, however, we need to have a look at other characteristics like correlation and relationship with the target variable.

Skewness and kurtosis in the variables are everywhere between positive and negative. One that stands out is variable x4 which has a very high negative skewness and kurtosis which will be reflected in its distribution.

2.2.2 Distributions

From the statistical calculations in the previous section we could see that several variables present skewness. Figure 1 corroborates this visually.

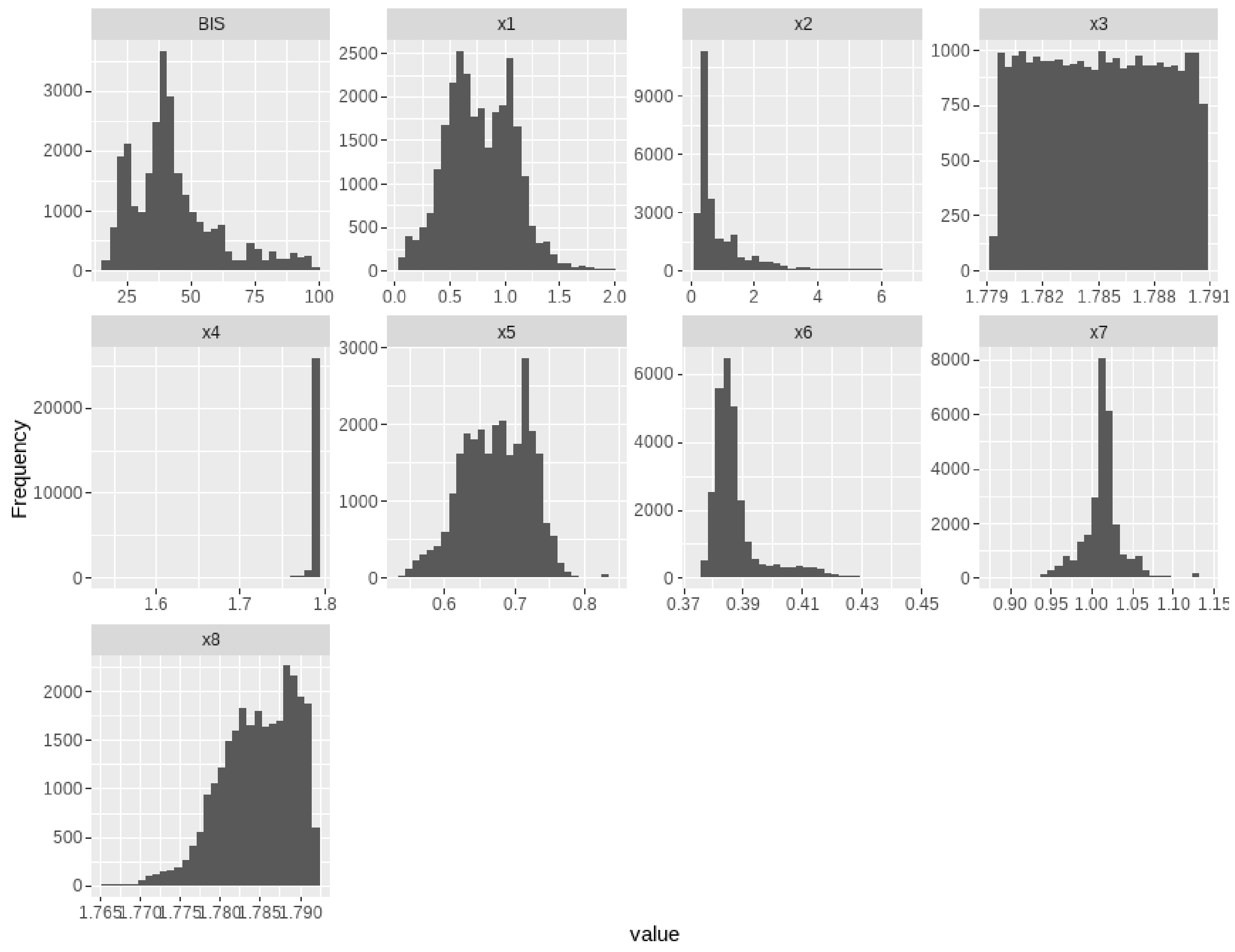


Figure 1: Histogram of variables

- BIS: this variable present a slight right skewness with two modes
- x1: this variable is a symmetric bimodal distribution
- x2: this variable is left skewed
- x3: this is variable present an uniform distribution
- x4: this variable is heavily skewed to the left
- x5: this variable is symmetric, close to normal
- x6: this variable is moderately skewed to the right
- x7: this variable is symmatric
- x8: this variable is skewed to the left

2.2.3 Relationships

Following the distributions we analysed the relationship between the variables, in terms of strength, direction and form.

We start with a scatter plot matrix between the BIS index and all the independent variables.

We can see in Figure 2 that:

- x1, x5, and x7 present a positive linear relationship with the BIS index.
- x2, x6 and x8 present a non-linear relationship with the BIS index.
- x3 and x4 present no linear relationship with the BIS index.

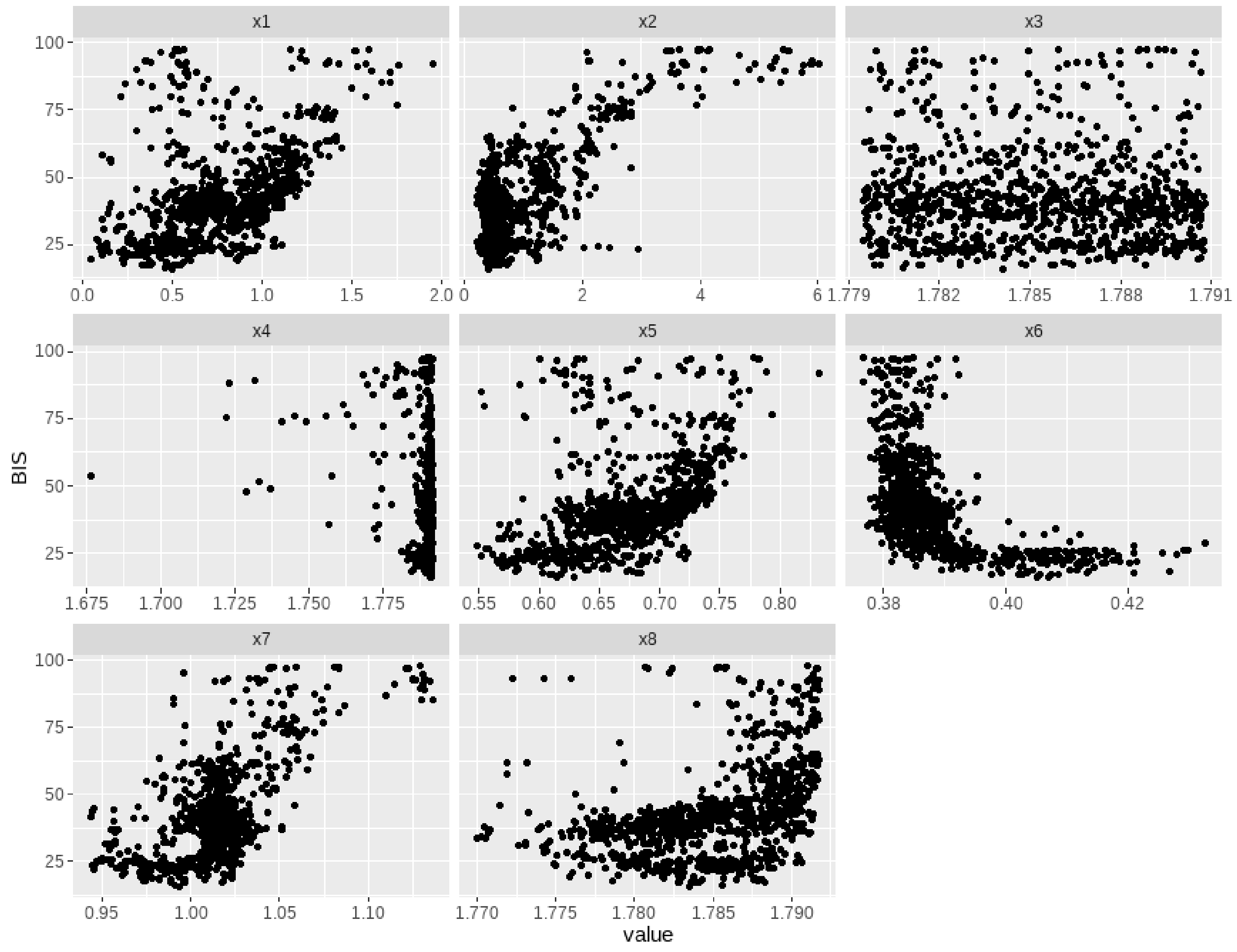


Figure 2: Variable scatter plots

Figure 3 provides a visual summary of a pairwise correlation matrix between all the variables. Here I can see that the BIS variable:

- has a strong positive correlation with x2 and x7
- has a moderate positive correlation with x1, x5 and x8
- has a moderate negative correlation with x6

- has a weak negative correlation with x4
- and has a very weak positive correlation with x2

Another interesting insight here is that x1 with x5 present a very strong correlation, and x2 with x7 present a strong correlation. The first case indicate a multicollinearity problem.

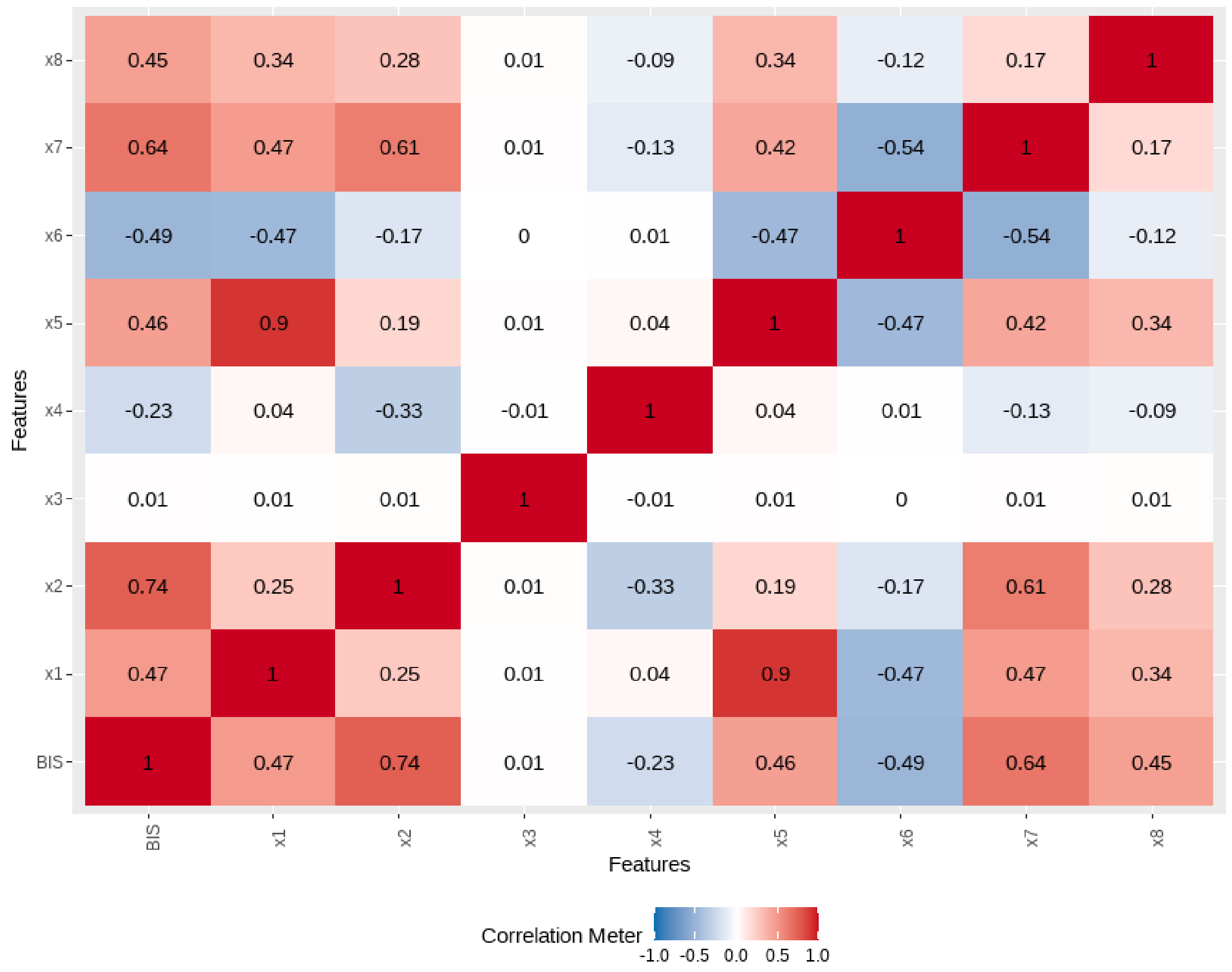


Figure 3: Correlation matrix

2.2.4 Outliers

It is important to identify whether our dataset contains outliers or not as this can influence the performance of some machine learning algorithms. We'll assess for outliers visually and via statistics (standard-deviation-based threshold method).

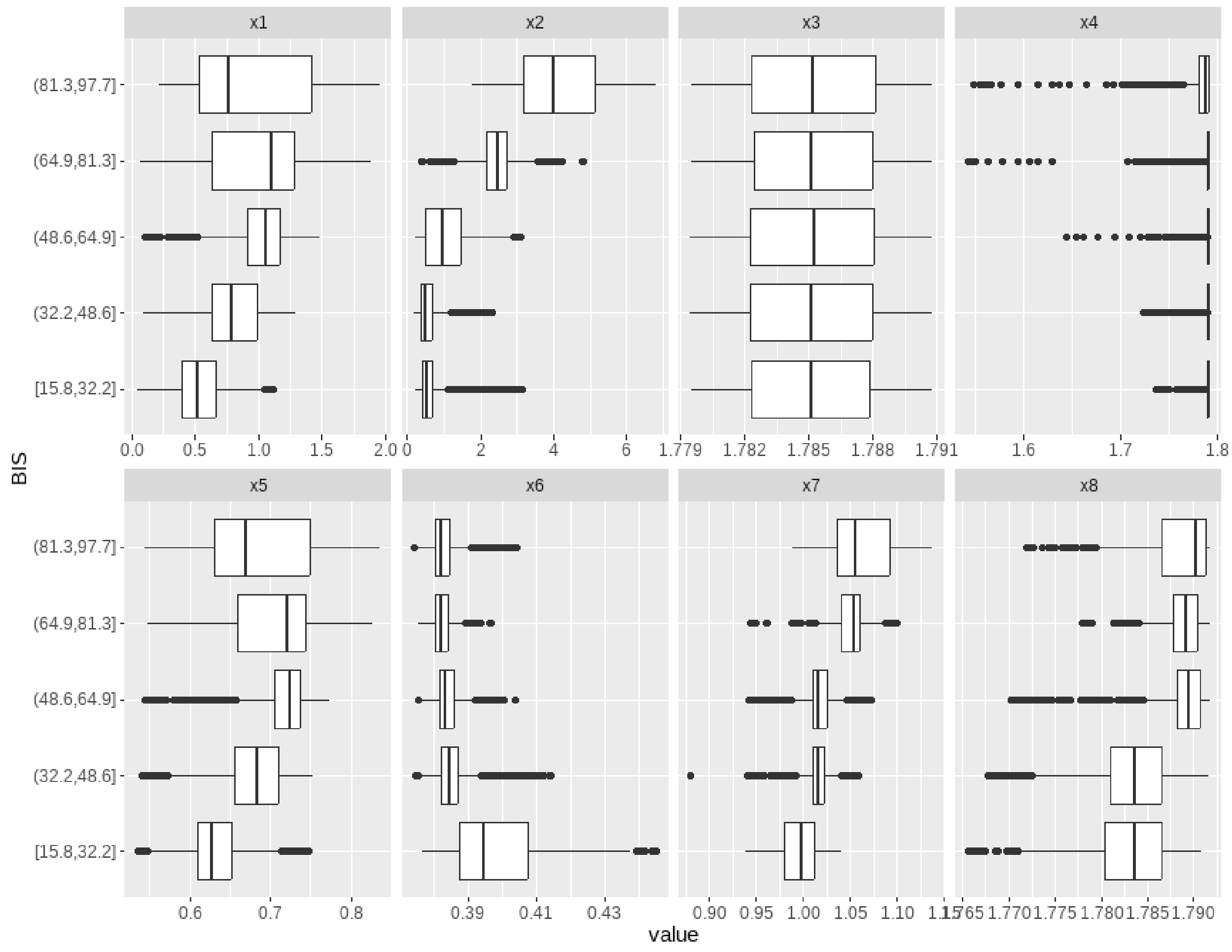


Figure 4: Boxplot of independent variables againt BIS index

Table 3 provides more insights around outliers by calculating the mean value of the outliers, the mean value of the variable with outliers and without outliers. This way I can assess how strong the influence of outlier for every variables is.

Table 3: Outlier statistics

variables	outliers_cnt	outliers_ratio	outliers_mean	with_mean	without_mean
BIS	1,801	6.56	87.05	43.23	40.15
x1	103	0.38	1.84	0.78	0.78
x2	2,147	7.82	3.66	0.97	0.74

variables	outliers_cnt	outliers_ratio	outliers_mean	with_mean	without_mean
x3	0	0.00		1.79	1.79
x4	3,117	11.35	1.78	1.79	1.79
x5	23	0.08	0.83	0.67	0.67
x6	3,016	10.99	0.41	0.39	0.39
x7	3,957	14.41	1.02	1.01	1.01
x8	153	0.56	1.77	1.78	1.78

We can see that here are a number of outliers in pretty much all variables except x3 which makes sense as it is a uniform distribution. x4, x6 and x7 have more than 10% of their observations as outliers. In terms of strength of influence I can see that the mean values change only for the BIS index and x2. All the other variables maintain their mean values.

2.2.5 Normmality test

Checking for normal distribution is common test performed in data analysis. The following table 6 presents the result of Shapiro-Wilk normality test over all variables at once. We can see that based on this test none of the variable passed the normality test.

Table 4: Normality test

vars	statistic	p_value	sample
BIS	0.9064549	0	5,000
x1	0.9907170	0	5,000
x2	0.6694741	0	5,000
x3	0.9555252	0	5,000
x4	0.1977663	0	5,000
x5	0.9854309	0	5,000
x6	0.7415666	0	5,000
x7	0.9129020	0	5,000
x8	0.9584101	0	5,000

2.2.6 Data analysis conclusion

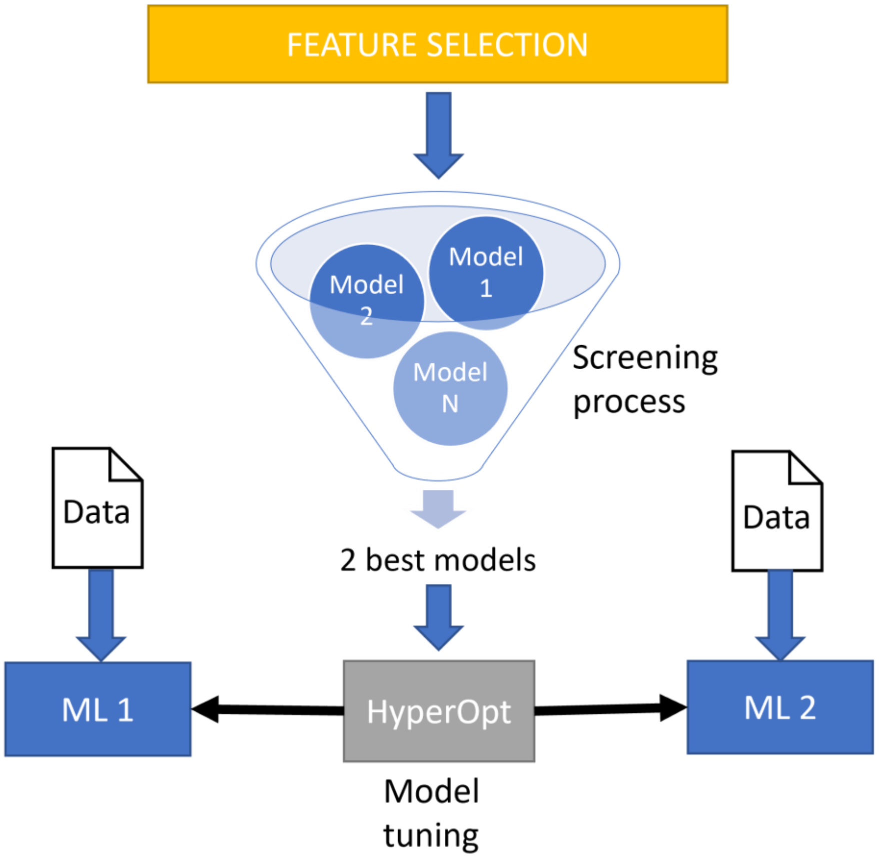
From all the analysis performed above we can conclude that the training dataset contains:

- No missing values.

- No negative values: This makes sense as I wouldn't expect negative values for BIS index or EEG variables.
- Presence of outliers: This is an aspect that could introduce problems in machine learning algorithms as some methods are more sensitive to outliers than others. If an outlier is an error then it has to be filter out of the dataset, otherwise, it needs to be investigated. However, given that I don't have much more information about the patients and the variables I can't assume that they are errors coming from the collection process so I will keep the outliers in the dataset and will apply a transformation to generate a more balanced distribution.
- No normal distribution in any of the variables.
- Correlations between features: There is a high correlation between two independent variables which indicates the existence of multicollinearity.
- Linear correlation between target and features: There is a low to moderate correlation between the BIS index and the features

2.3 Methodology

Several models have been used in the studies reviewed for this report. Most of them are neural networks with combination of other transformations. At a high level, the approach proposed for this project is to perform:



Process

1. a feature selection step then
2. a modelling step and finally
3. a testing step where final models will run against the test data

In the first step we'll perform several procedures to ensure the input data has enough information to predict the new BIS index.

In the modelling step, instead of picking two models from the given list of options, I'm going to select from the list four potential models to evaluate on the training data set (screening). Then I'll pick the best two based on their performance via k-fold cross validation.

After the final selection, the two selected models will be fitted again and their parameters tuned in order to find the best model configuration.

The final outcome will be the prediction of each test dataset via the two tuned selected models.

2.4 Evaluation methods

During the screening process the five pre-selected models will be trained with default parameters using a 10-fold cross-validation training approach. This approach provides two benefits: it reduces the chances of overfitting and provides better estimates on average performance between the models.

They'll be evaluated using root mean square error (RMSE), R square (R^2), mean square error (MSE) and mean absolute error (MAE). R^2 values go from 0 to 1 where high values indicate higher correlation between the truth and the predicted value from training

After the best two models are selected and evaluated on the test datasets, I'll use the scatter plot diagram and Bland-Altman diagram method to assess the agreement between the new BIS index (prediction) and the BIS index. Also, the pearson correlation coefficient will be calculated to measure correlation.

2.5 Machine Learning methods application

2.5.1 Software

Due to previous experience, knowledge gained in other courses in my studies, personal interests and experience at work, I have used R during the implementation of this project. While R comes fully equipped with functions and features, working with RStudio as the IDE is a modular process based on different packages that need to be installed. Each package's objective covers different use cases and like this project involves data analysis, machine learning algorithms, visualization and report writing, I have used the following packages to assist me during the project. These packages are:

- Rmarkdowm, package to write reproducible reports where code and text and can be exported into word, html and pdf.
- Tidyverse, package to do all kind of data wrangling.

- Tidymodels, package that contains a collection of modelling packages that support all the machine learning activities like data pre-processing, training, cross-validation, testing, etc.
- Ggplot2, package that provides visualization capabilities in R.
- doParallel, package for managing CPU cores and perform parallel programming.

2.5.2 Feature Selection

In this step we will perform some model-based feature selection in order to understand how different models rank the feature and based on those output make an informed decision of which feature should be kept for the modelling step.

The following are the procedures that we will use:

Boruta algorithm: This is a model-based procedure that uses random forest to apply a feature importance measure and evaluates it in several iterations. The graph below present the result of this procedure and we can see the most important features are: x2, x8, x6, x5 and x4.

Tree-based method: This is a model-based feature importance procedure that uses a decision tree model's feature importance function. The graph below show the following features as the most important: x7, x6, x2, x1, x8 and x5.

Stepwise selection: This procedure assumes linear regression. It searches for the best possible linear regression model by iterating over different models with different features from the dataset until it minimizes the AIC measure. The result of both way steps here were: x2, x6, x8, x5, x7, x1 and x4.

In summary, the three procedures we run for feature selection didn't pick x3 as a relevant feature, x2 was picked as the most important feature followed by x8.

Table 5: Summary Feature Importance

ranking	Barota	rPart	stepWise
1	x2	x7	x2
2	x8	x6	x6
3	x6	x2	x8
4	x5	x1	x5
5	x4	x8	x7
6		x5	x1
7			x4

Considering the insights from the data analysis section (correlations and skewness) and the first 5 most important features provided by the model-based feature importance procedures above, I have selected the following features for the models: **x2, x5, x6, x7 and x8**.

- x2 was pretty much the most important feature in all three models.
- x5 had a high correlation with x1, however, I selected it due to the fact that x1 was mostly classified as a low importance feature.
- x6 was pretty much the most important feature in all three models.
- x7 was selected high importance feature by one of the models only, however, it presented some good characteristics as a feature, like moderate linear correlation with the BIS and low skewness.
- x8 was selected relatively high in two of the models and presented some good characteristics as a feature, like moderate linear correlation with the BIS, low skewness and low number of outliers.

2.5.3 Data Transformations

The following data transformation steps will also be applied:

1. Standarisation of the features to prevent scaling issues.
2. Application of log normal transformation to the features with moderate to high skewness (x2 and x6).

These two steps were coded in a preprocessing function which is used by all the models.

2.5.4 Screening process

The four models we've initially selected belong to a mixture of algorithms that match the characteristics of this problem. They are also low bias models which means that they make very little assumptions about the target and features relationship. We have also considered one non-linear model given that this characteristic has been stated in several studies.

- K-Nearest Neighbor (instance-based algorithm)
- Support vector machines (non-linear algorithm)
- Neural Networks (non-parametric algorithm)
- Random Forest (tree-based algorithm)
- Multivariate Adaptive Regression Splines (non-linear algorithm)

We have left linear algorithms out due to the non-linear relationship nature between target and features (high bias models).

2.5.5 Model selection

Table 6 shows the results after running the screening process with the four pre-selected algorithms. The table list the best model at the top by RMSE metric. KNN and RF are the two models that performed the best during the cross validation procedure. These models are not tuned and were trained with default parameter values. As result, KNN and RF both showed a R^2 of 0.97 while the other three models ANN, MARS and SVR all obtained an average of 0.85, 0.82 and 0.68 respectively.

Table 6: Screening process metric results

model	mae	mape	rmse	rsq	smape
knn	1.5681	3.8821	2.6901	0.9749	3.8430
rf	1.7596	4.5429	2.7239	0.9746	4.4576
ann	4.7008	12.0830	6.4708	0.8553	11.6022
mars	5.0363	12.9847	7.0998	0.8257	12.3229
svr	6.7331	17.6455	9.5667	0.6849	17.0001

The best two models that were selected are: **Random Forest and KNN**.

Hyperparameter tunning

The model parameters were adjusted by using a grid approach with 25 configuration options.

Each model configuration run is passed through a 10-fold cross validation process to ensure robustness in the metric results.

Random forest parameters:

- mtry: this is an integer number that indicates the number of predictors that will be randomly sampled at each split when creating the tree models.
- trees: this is the number of trees contained in the ensemble. we have used a fixed number of 500.
- min_n: this is an integer number that indicates the minimum number of data points in a node that are required for the node to be split further.

KNN parameters:

- neighbors: this is an integer that indicates the number of neighbors nearby to select.
- weight_func: this is a character value that indicates the distance weighting function to be used.
- dist_power: this is an integer number that indicates the Minkowski Distance Order.

2.6 Results

The following tables and figures present the training and testing results from the two selected final models. Table 7 summarises the metrics obtained from the runs. The hyperparameters that fitted the training dataset resulted in a RMSE of 1.24 for KNN with a R^2 of 0.99, and 1.09 for RF with a R^2 of 0.99 as well.The generalisation metrics on the test dataset were 10.1 for KNN with a R^2 of 0.68 and 9.4 for RF with a R^2 of 0.7.

Table 7: Training and testing metric results

	KNN		RF	
metric	Train	Test	Train	Test
rmse	1.2369506	10.103118	1.0902244	9.4068155
mae	0.7173498	7.464314	0.6538956	6.9402343
mape	1.7589280	24.598235	1.6883355	24.1745195
rsq	0.9947307	0.677090	0.9959896	0.7030126

The predicted versus actual plot in figure x shows how well the regression model predicts the BIS index on the test dataset. The red line represents a perfect regression model.

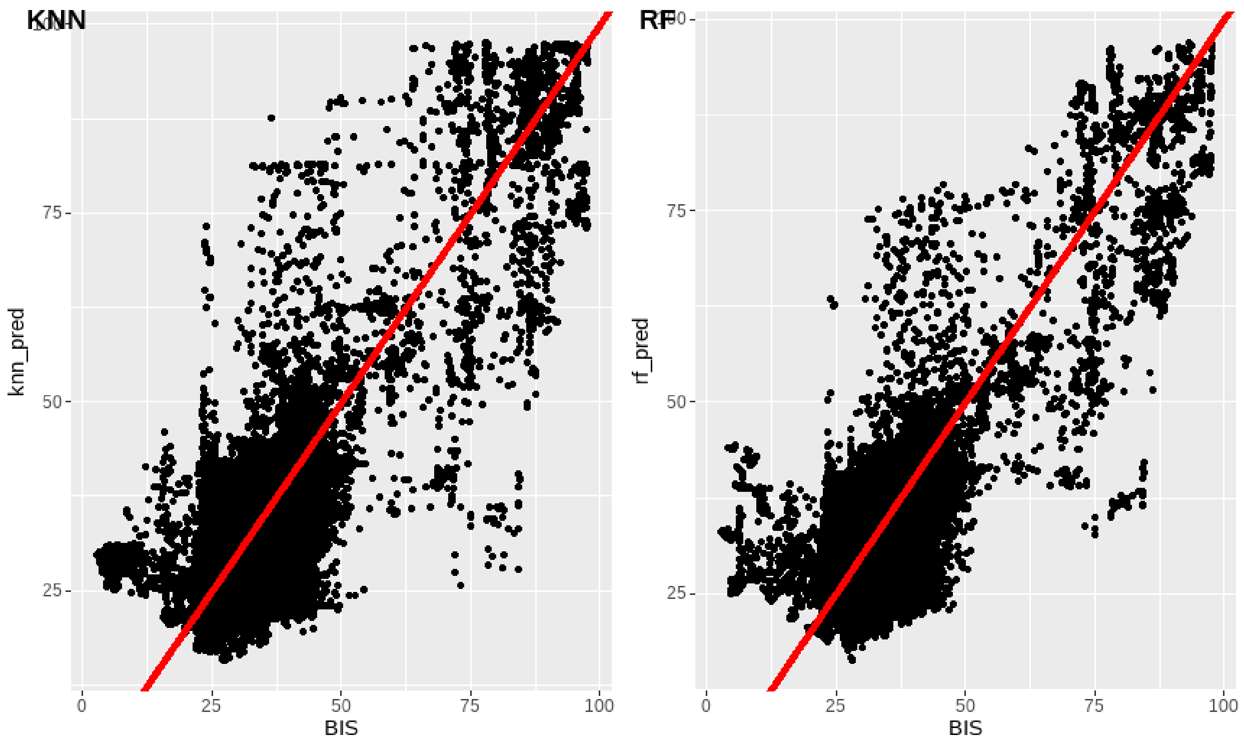


Figure 5: Predicted vs actual scatter plot

BIS vs new index

The following plot shows how the new two index aligned to the previous BIS index for all the 5 cases.

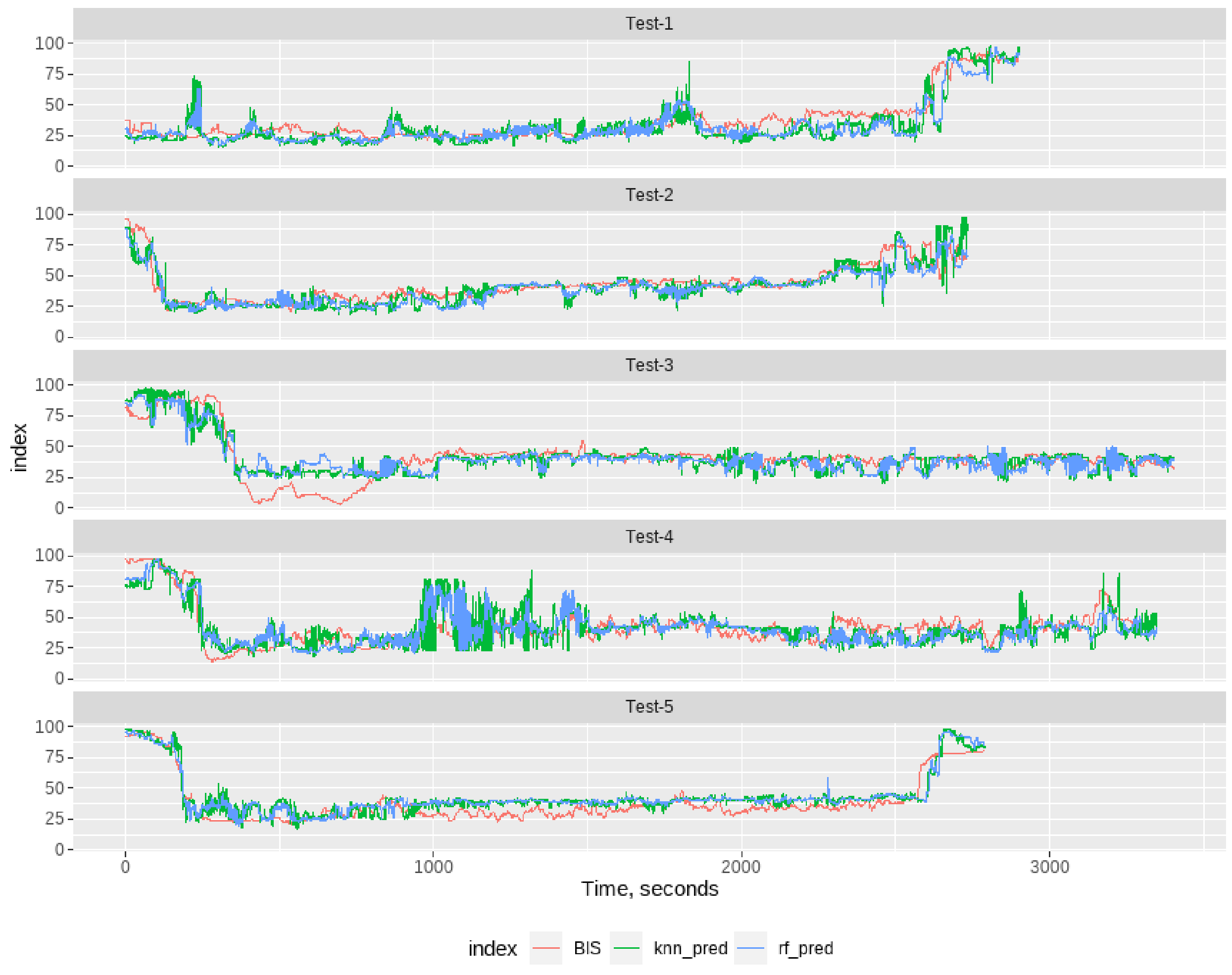


Figure 6: new indexes vs BIS index on test cases

Pearson coefficient by test case

In figure 7 we can see that the pearson coefficient is over 0.8 for all cases except test case 4. This might be due to some patterns not being covered in the training dataset.

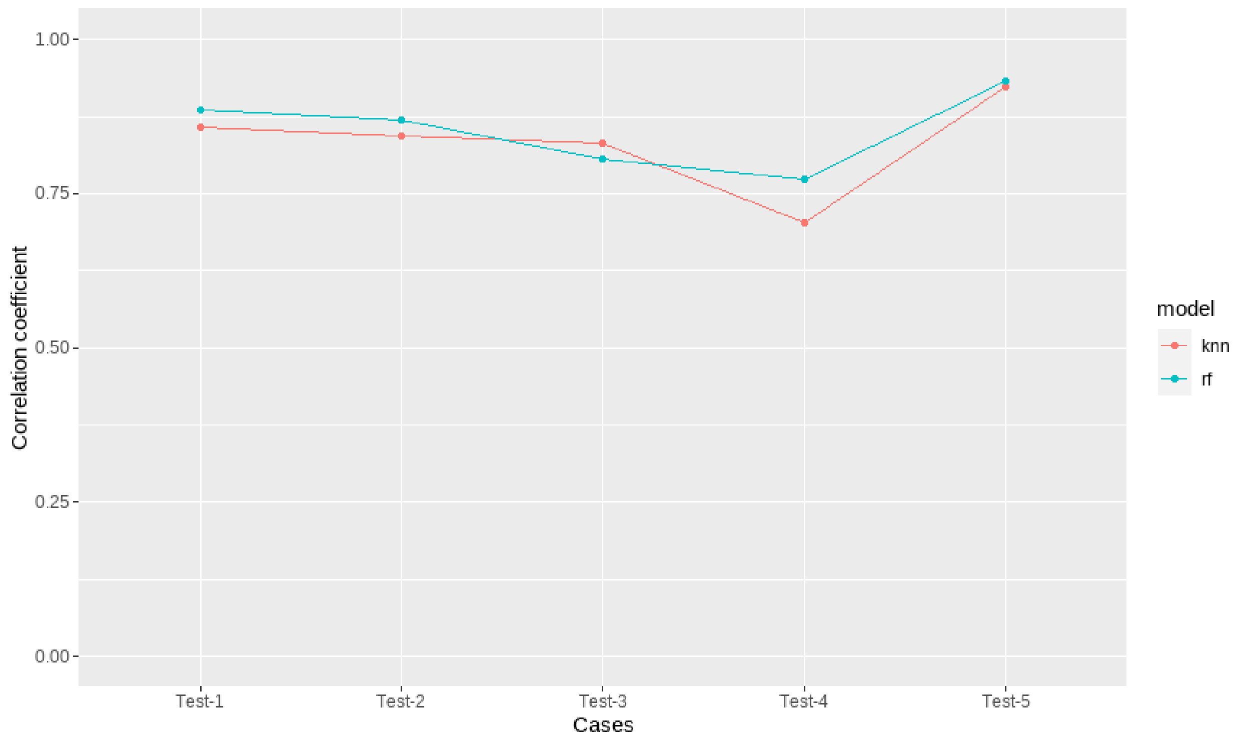


Figure 7: Final models pearson coefficient by test case

Bland-Altman plots

The Bland-Altman plot tells the level of agreement between the new BIS index and the previous one. The estimated bias and 95% confidence intervals for both the knn-based and rf-based index are:

KNN * bias: 0.03 * confidence interval: [-20.18, 20.23]

RF * bias: 0.38 * confidence interval: [-18.41, 19.18]

We can see that the differences have a normal distribution which indicates that 95% of these differences lie between the intervals (red dotted lines)

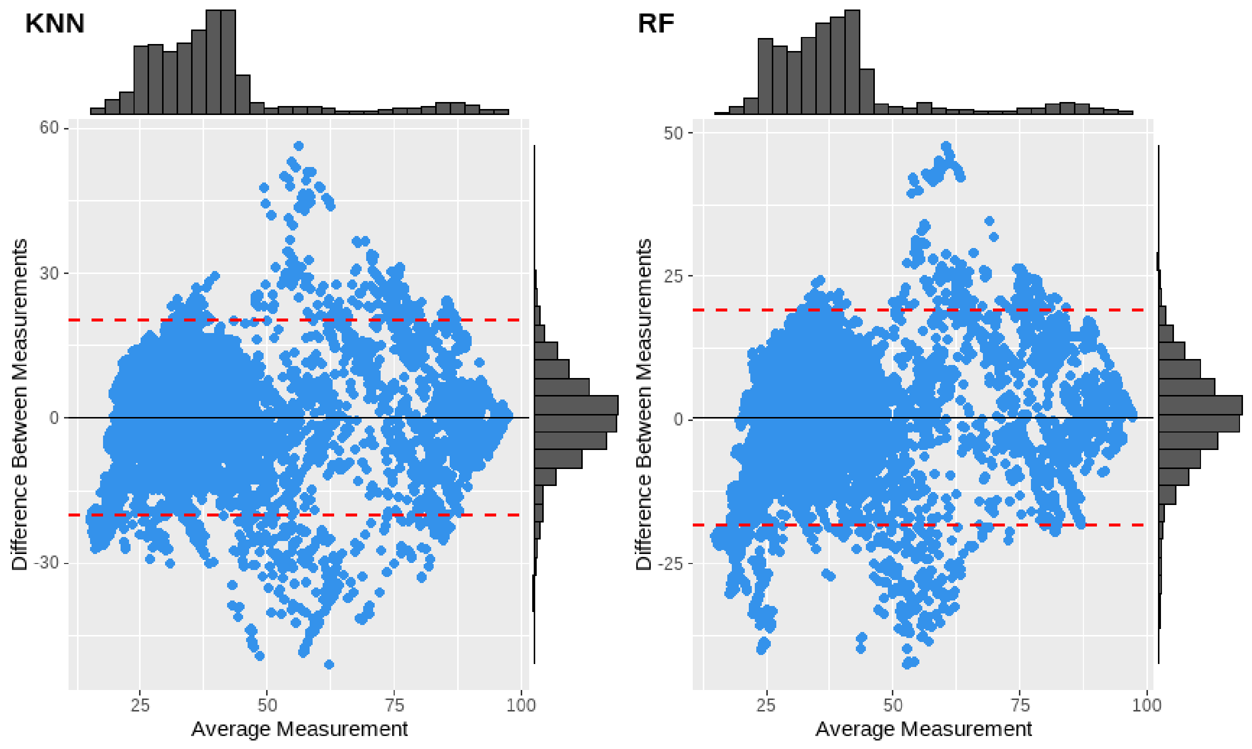


Figure 8: Bland-Altman plots and distributions

2.7 Discussion

In this report, machine learning algorithms were implemented and studied to create a new BIS index. Firstly we run a screening process to select two suitable models from a pre-selection of four models. This approach gives us a unbiased criteria for selecting the two models to train further. One opportunity of improvement could be that we run the screening process with hyperparameters tunning at the same time instead of default values. The final results were evaluated by comparing the correlation and agreement with the BIS index provided in the test cases.

Our two indexes use a k-nearest neighbor and random forest model to make the predictions based on five EEG features. Both models predicted BIS indexes for five cases with different levels of correlation, however, most of them were highly correlated (>0.8). There is one exception, We can see that test-4 case shows the lowest pearson coefficient. We think this could be due to some patterns in the test dataset not being covered in the training dataset. We can see that there is a section in test-4 where BIS values are lower than the minimum values available in the training dataset.

Table 8: Pearson Coefficient by Test case

case	knn	rf
Test-1	0.8578884	0.8857494
Test-2	0.8439821	0.8691932
Test-3	0.8313535	0.8060529
Test-4	0.7028448	0.7723897
Test-5	0.9221789	0.9317893

The challenges faced during the implementation of this report were training time during the screening and hyperparameter tunning processes. This was solved in a way by introducing parallel processing.

Another challenge was the organisation of the steps between pre-processing the data, feature selection, training, testing and calculating metrics. This was solve by using the Tidymodels framework which helped in managing all this in a structured way.

3. References

1. Gu, Y.; Liang, Z.; Hagihira, S. Use of Multiple EEG Features and Artificial Neural Network to Monitor the Depth of Anesthesia. Sensors 2019, 19, 2499

2. Shalbaf, R.; Behnam, H.; Sleigh, J.W.; Steyn-Ross, A.; Voss, L.J. Monitoring the depth of anesthesia using entropy features and an artificial neural network. J. Neurosci. Methods 2013, 218, 17–24.

3. Peker, M.; Arslan, A.; Sen, B.; Celebi, F.V.; But, A. A novel hybrid method for determining the depth of anesthesia level: Combining ReliefF feature selection and random forest algorithm (ReliefF+RF). In Proceedings of the 2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA), Madrid, Spain, 2–4 September 2015; pp. 1–8.

4. Shalbaf, A.; Saffar, M.; Sleigh, J.W.; Shalbaf, R. Monitoring the Depth of Anesthesia Using a New Adaptive Neurofuzzy System. IEEE J. Biomed. Heal. Inform. 2017, 22, 671–677.

5. Liang, Z.; Huang, C.; Li, Y.; Hight, D.F.; Voss, L.J.; Sleigh, J.W.; Li, X.; Bai, Y. Emergence EEG pattern classification in sevoflurane anesthesia. Physiol. Meas. 2018, 39, 045006.

6. Liu, Q.; Cai, J.; Fan, S.-Z.; Abbod, M.F.; Shieh, J.-S.; Kung, Y.; Lin, L. Spectrum Analysis of EEG Signals Using CNN to Model Patient's Consciousness Level Based on Anesthesiologists' Experience. IEEE Access 2019, 7, 53731–53742

7. Huang, Yi, Wen, Peng, Song, Bo and Li, Yan. 2022. "Real-Time Depth of Anaesthesia Assessment Based on Hybrid Statistical Features of EEG." Sensors. 22 (16).

4. Appendix

4.1 Reading file

```
library(tidymodels) # package for data manipulation
library(readxl) # package for reading excel files

# read data set
# accessing all the sheets from dataset
sheet = excel_sheets("Project data set 1 (for reports 1 and 3) .xlsx")

# applying sheet names to dataframe names. Training data is contained within
the first 10 tabs
data_frame = lapply(setNames(sheet[1:10], sheet[1:10]),
                    function(x) read_excel("Project data set 1 (for reports 1
and 3) .xlsx", sheet=x))

# attaching all dataframes together
eeg_data_train = bind_rows(data_frame, .id="Sheet") %>% select(-Sheet)

# applying sheet names to dataframe names. Test data is contained in the last
5 tabs
data_frame = lapply(setNames(sheet[11:15], sheet[11:15]),
                    function(x) read_excel("Project data set 1 (for reports 1
and 3) .xlsx", sheet=x))


eeg_data_test = bind_rows(data_frame, .id="Sheet") %>% select(-Sheet)
```

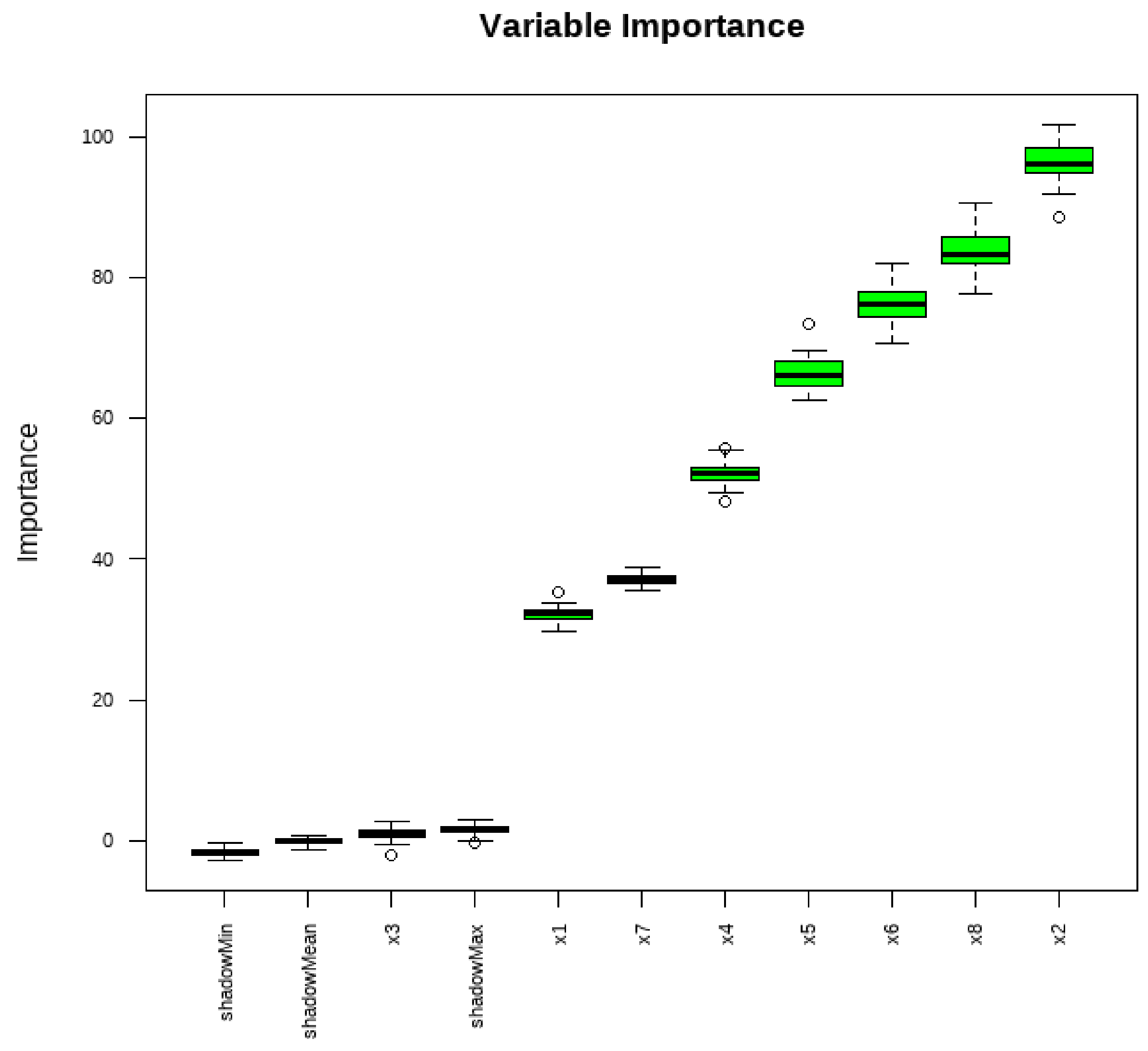
4.1 Feature selection code

Boruta

```
library(Boruta)
# Decide if a variable is important or not using Boruta
boruta_output <- Boruta(BIS ~ ., data=na.omit(eeg_data_train), doTrace=2)  #
perform Boruta search

boruta_signif <- names(boruta_output$finalDecision[boruta_output$finalDecisio
n %in% c("Confirmed", "Tentative")])  # collect Confirmed and Tentative varia
bles
print(boruta_signif)  # significant variables

plot(boruta_output, cex.axis=.7, las=2, xlab="", main="Variable Importance")
# plot variable importance
```



RF-based feature importance

rPart

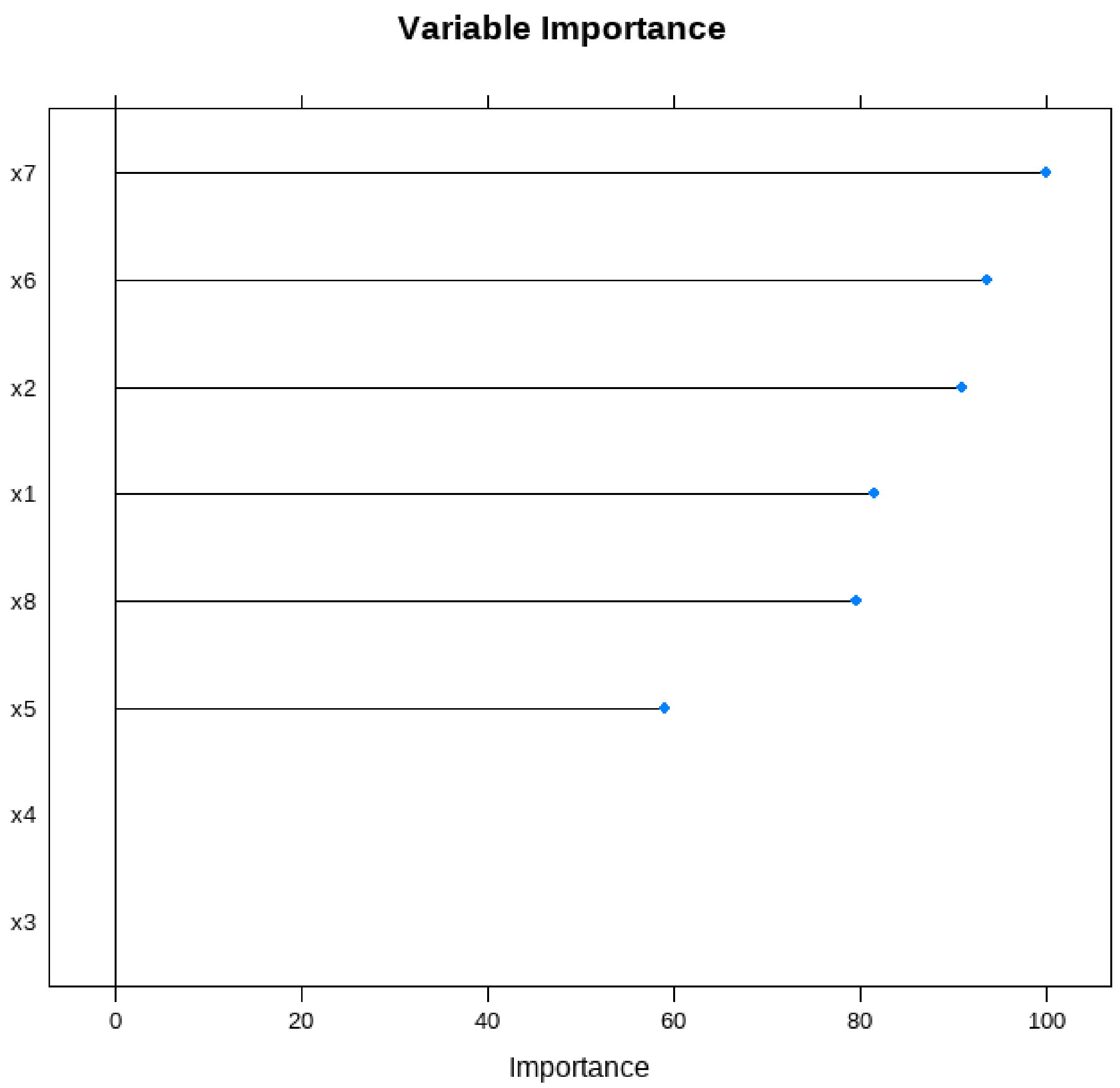
```
library(caret)

set.seed(100)
rPartMod <- train(BIS ~ ., data = eeg_data_train, method="rpart")

rpartImp <- varImp(rPartMod)

print(rpartImp)

plot(rpartImp, top = 8, main='Variable Importance') # plot variable importanc
e
```



DT-based feature importance

stepWise both ways

```
base.mod <- lm(BIS ~ 1 , data= eeg_data_train)  # base intercept only model

all.mod <- lm(BIS ~ . , data= eeg_data_train) # full model with all predictor
s

stepMod <- stats::step(base.mod, scope = list(lower = base.mod, upper = all.m
od), direction = "both", trace = 0, steps = 1000)  # perform

#step-wise algorithm
shortlistedVars <- names(unlist(stepMod[[1]])) # get the shortlisted variable
.
shortlistedVars <- shortlistedVars[!shortlistedVars %in% "(Intercept)"]  # re
```

```r
move intercept
print(shortlistedVars)
```

4.1 Screening process code

```r
library(tidymodels)

# Defining the pre-processing steps
data_recipe <- recipe(BIS ~ ., data = eeg_data_train) %>%
  # select columns that will be used
  step_select(BIS, x2, x5, x6, x7, x8, skip = T) %>%
  # apply log transformation function to features
  step_log(x2, x6) %>%
  # normalise all predictors
  step_normalize(all_numeric_predictors())

# Defining the models for the screening
# Multi-adaptive Regression Splines
mars_reg_spec <-
  mars() %>%
  set_mode("regression") %>%
  set_engine("earth")

# knn
knn_mod_spec <-
  nearest_neighbor() %>%
  # This model can be used for classification or regression, so set mode
  set_mode("regression") %>%
  set_engine("kknn")

# SVR
svr_mod_spec <-
  svm_linear() %>%
  set_engine("kernlab") %>%
  set_mode("regression")

# ANN
nn_mod_spec <-
  mlp() %>%
  # This model can be used for classification or regression, so set mode
  set_mode("regression") %>%
  set_engine("nnet")

# RF
rf_mod_spec <-
  rand_forest() %>%
  set_engine("ranger") %>%
  set_mode("regression")

# Define the cross validation strategy, 10 folds
```

```
set.seed(345)
folds <- vfold_cv(eeg_data_train, v = 10)

# Define the workflow to be applied to all models: preprocess inputs and trai
n with 10-fold CV.
spot_check_wf <-
  workflow_set(
    preproc = list(none = data_recipe),
    models = list(knn = knn_mod_spec, svr = svr_mod_spec, nn = nn_mod_spec, r
f = rf_mod_spec, mars = mars_reg_spec)
  )

# To speed things up, let's run them in parallel using all my cores
library(doParallel)
all_cores <- parallel::detectCores(logical = FALSE)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)

# Fitting process
start <- Sys.time()
set.seed(456)
spot_check_rs <-
  spot_check_wf %>%
  workflow_map("fit_resamples", resamples = folds, metrics = metric_set(rmse,
rsq, mae, mape, smape))
end <- Sys.time()
print(end - start)
```

4.2 Random Forest hyperparameter tunning

```
# Data transformation
data_recipe <- recipe(BIS ~ ., data = eeg_data_train) %>%
  step_select(BIS, x2, x5, x6, x7, x8, skip = T) %>%
  step_log(x2, x6) %>%
  step_normalize(all_numeric_predictors())

# Defining the model Random forest
rf_mod_spec <-
  rand_forest(
    mtry = tune(),
    trees = 500,
    min_n = tune()
  ) %>%
  set_mode("regression") %>%
  set_engine("ranger")

set.seed(345)
folds <- vfold_cv(eeg_data_train, v = 5)

rf_grid <- grid_regular(
```

```r
  mtry(range = c(1, 5)),
  min_n(range = c(2, 8)),
  levels = 5
)

ctrl <- control_grid(verbose = FALSE, save_pred = TRUE)

library(doParallel)
all_cores <- parallel::detectCores(logical = FALSE)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)

start <- Sys.time()
set.seed(0239)
rf_tune <- tune_grid(rf_mod_spec, data_recipe, folds, metrics = metric_set(rm
se, rsq, mae, mape, smape), grid = 20, control = ctrl)
end <- Sys.time()
print(end - start)

show_best(rf_tune, metric = "rmse")

rf_best <- rf_tune %>% select_best(metric = "rmse")

# mtry min_n .metric .estimator  mean     n std_err .config
# <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
# 1     4     5 rmse    standard    2.70     5  0.0356 Preprocessor1_Model17
# 2     4     9 rmse    standard    2.76     5  0.0327 Preprocessor1_Model09
# 3     3    12 rmse    standard    2.82     5  0.0311 Preprocessor1_Model06
# 4     2     6 rmse    standard    2.83     5  0.0285 Preprocessor1_Model07
# 5     2    10 rmse    standard    2.91     5  0.0309 Preprocessor1_Model03
```

4.3 KNN hyperparameter tunning

```r
data_recipe <- recipe(BIS ~ ., data = eeg_data_train) %>%
  step_select(BIS, x2, x5, x6, x7, x8, skip = T) %>%
  step_log(x2, x6) %>%
  step_normalize(all_numeric_predictors())

# Defining the models
# knn
knn_mod_spec <-
  nearest_neighbor(
    neighbors = tune(),
    weight_func = tune(),
    dist_power = tune()
  ) %>%
  set_mode("regression") %>%
  set_engine("kknn")

set.seed(345)
```

```
folds <- vfold_cv(eeg_data_train, v = 10)

knn_grid <- grid_regular(
  neighbors(),
  weight_func(),
  dist_power(),
  levels = list(neighbors = 4, weight_func = 4, dist_power = 1)
)

ctrl <- control_grid(verbose = FALSE, save_pred = TRUE)

library(doParallel)
all_cores <- parallel::detectCores(logical = FALSE)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)

start <- Sys.time()
set.seed(0239)
knn_tune <- tune_grid(knn_mod_spec, data_recipe, folds, metrics = metric_set(
rmse, rsq, mae, mape, smape), grid = knn_grid, control = ctrl)
end <- Sys.time()
print(end - start)

show_best(knn_tune, metric = "rmse")

# neighbors weight_func  dist_power .metric .estimator  mean     n std_err .c
onfig
# <int> <chr>               <dbl> <chr>   <chr>       <dbl> <int>   <dbl> <chr>
# 1          7 biweight              1 rmse    standard    2.47    10  0.0431
Preprocessor1_Model03
# 2         10 biweight              1 rmse    standard    2.49    10  0.0424
Preprocessor1_Model04
# 3          7 triangular            1 rmse    standard    2.49    10  0.0419
Preprocessor1_Model15
# 4          4 triangular            1 rmse    standard    2.50    10  0.0440
Preprocessor1_Model14
# 5          4 epanechnikov          1 rmse    standard    2.51    10  0.0445
Preprocessor1_Model06
```