Universiteit Leiden

Master Computer Science

Investigating the epoch size and feature engineering for Automated Machine Learning in EEG data analysis

Name:	Konstantina Papada
Student ID:	s2477254
Date:	08 June 2023
Specialisation:	Data Science
1st supervisor:	Furong Ye
2nd supervisor:	Victor Geraedts

Master's Thesis in Computer Science
Leiden Institute of Advanced Computer Science (LIACS)

Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Contents

List of Figures

List of Tables

Acknowledgment

I would like to express my deepest gratitude to my supervisor, Furong Ye, for his guidance, support, valuable insights throughout the research process and expertise. Thanks should also go to my supervisor, Victor Geraedts, who generously provided knowledge, valuable feedback and domain expertise for Parkinson's Disease.

I'm also grateful to my family and Dionysis's family for their love, support and understanding. Lastly, I'm deeply grateful to Dionysis for his encouragement, patience, support and understanding during my academic obligations.

Abstract

This study investigates the impact of the length of epochs and feature engineering techniques on Electroencephalography (EEG) classification tasks. The work is implemented on a dataset consisting of EEG data of 47 Parkinson's Disease patients. Five epochs of 10 seconds exist for each patient. We partition each 10-second epoch into two 5-second epochs to compare the results of using different lengths of epochs. For feature engineering, we compare two techniques: a combination of ts-fresh and Boruta and Catch22. The former applies ts-fresh to extract an amount of time series features for each epoch of EEG data and uses Boruta to select a small set of significant features for the classification model. Catch22 is a collection of 22 canonical time-series characteristics. An automated random forest model tuned using Bayesian optimization, is applied for the classification tasks based on the features provided by the feature engineering step.

Using the combination of ts-fresh and boruta shows similar performances for the 10-second and 5-second epochs data. However, when we conduct experiments using the average of 5 epochs, 10-second data presents better performance with an F1-score of 92%. While for the experiments using individual epochs (e.g., conducting five independent experiments using one epoch), 5-second data obtains better performance with a maximum F1-score of 96.5%. In addition, we conduct two experiments on Catch22, namely the Catch22-Compact method (i.e., selecting 22 out of 5*22 features) and the Catch22-Comprehensive method (using 110 features). The Catch22-Compact method obtains a maximal F1-score of 97% when performing on individual epochs and an F1-score of 93.4% for the average of 5 epochs for 10-second data, and it obtains a maximum F1-score of 99.3% and 96.5% for the same settings of 5-second data.

In conclusion, this study demonstrates that classification performance depends on feature engineering techniques and the EEG epoch length. The Catch22-Compact method is the best compared to the other tested feature engineering methods across all data settings in this thesis.

1 Introduction

More than 10 million people have been diagnosed with Parkinson's Disease(PD) worldwide which is deemed the second most common neurodegenerative disease[1]. Parkinson's disease is a neurodegenerative disorder which is characterized by motor symptoms as well as non-motor symptoms such as cognitive impairment [6]. Cognitive function may be difficult to assess due to impaired visuospatial ability, motor symptoms may impair drawing/writing exercise, and speech difficulties may impair verbal fluency. On the other hand, biomarkers could provide a more objective and efficient way to track changes in the cognitive function of patients with PD. Electroencephalography (EEG) is considered a potential biomarker for cognitive function in Parkinson's disease due to its unbiased nature, however, qualitative assessment requires expertise and quantitative analyses require long epochs without artifacts [7]. EEG is a non-invasive technique that measures the electrical activity of the brain using electrodes placed on the scalp [7]. EEG is a useful tool which is used to predict cognition [7]. This examination has a duration of almost 30 minutes and each patient should follow strict instructions[7]. As a result of the strict instructions, only a small part of the produced data (i.e., some seconds) contains fewer artifacts.

This research aimed to study the impact of the size of EEG epochs on classifying whether a patient has "good cognition" or "poor cognition" by using an automated machine learning pipeline. A random forest model is trained using patients who have been classified as having either *good* or *poor* cognitive performance based on clinical assessment. The dataset consists of 47 patients and each patient has five epochs from an EEG examination. 27 patients are labelled as having "*good cognition*" and 20 patients as having "*poor cognition*". The labels of patients were defined in accordance with research [12].

In addition, two feature engineering techniques were employed in the experiments to transform the raw data into meaningful and informative features, thereby enhancing the performance and interpretability of the machine learning model. The performance of these techniques was compared to determine the best one for the data. The first technique is ts-fresh combined with boruta where ts-fresh extracted a large number of features from each time series and Boruta selected the most relevant features for training the model. The second technique is Catch22, which only extracted and selected 22 features of each time series which is significantly less than ts-fresh. The reason for choosing this technique is to reduce the dimensionality of the data and speed up model training.

The rest of this project is organized as follows. In Section 2, an overview of the related work is given. In Section 3, the datasets and automated machine learning pipeline were described in detail, as well as the two feature engineering techniques that were used to extract and select features. Section 4, the results of the experiments are shown and the performance of evaluation metrics is interpreted. Section 5, is the conclusion and the future work is shown in Section 6.

[1] https://www.parkinson.org/understanding-parkinsons/statistics

1.1 Problem Definition

Doctors use EEG examination as a biomarker to model cognition. Furthermore, an EEG examination can be performed in patients with Parkinson's disease who are unable to undergo cognitive testing to be classified as having "good cognition" or "poor cognition". This is a supervised classification problem which predicts the cognition of a patient with PD based on EEG examination. An EEG examination consists of several time series which originate from the measurements of electrodes which are used during the examination. The study will utilize the generated time series data of 47 patients with PD as the input. Since it is a binary classification problem, the objective is to classify patients into two classes: poor cognition (0) and good cognition (1) in accordance with research [12].

Moreover, the evaluation metrics will be the F1-score and accuracy, where the F1-score measures the performance of a binary classification problem, and the accuracy measures the percentage of correct prediction of the model. The process will involve data cleaning and an automated machine learning pipeline which consists of the following steps: feature engineering, model selection and hyperparameter optimization. The data will be split into training and test sets, and cross-validation will be used to tune the hyperparameters and select the best model.

The main challenge in cognitive prediction lies in the difficulty of interpreting EEG data due to the potential presence of artifacts that can hinder accurate analysis and interpretation. Currently, a selection of 5x10 sec. epochs (i.e., segments) are considered to be required for accurate reflection of the total EEG [12]. This project will investigate whether fewer seconds can give equally reliable results based on the metrics. This would mean that fewer EEGs are discarded due to too many artifacts and decreased pre-processing time due to less stringent criteria for epoch selection.

1.2 Objective and Research Questions

The objective of the thesis project is to assess whether fewer data can give reliable results for the cognition of patients with Parkinson's Disease. In addition, it was investigated whether fewer seconds of EEG examination can predict "poor cognition" or "good cognition" following the same procedure such as the 10 seconds of examination. For this reason, the used dataset consists of five epochs of an EEG of 10 seconds, while two subsets were generated by splitting the dataset of 10 seconds such that the length of 5 seconds was obtained. These subsets consist of five epochs of an EEG of 5 seconds, namely Part 1 and Part 2. Moreover, two different feature engineering techniques are compared to find the technique with the most reliable results.

The research questions to reaching this objective are:

1. What is the impact of the length of EEG epochs for the performance of machine learning models on classifying Parkinson's patient cases?

2. What impact does each feature engineering technique (i.e., ts-fresh - boruta, Catch22) have on the model that predicts whether a patient with PD has poor or good cognition?

These research questions align with the objective of using fewer seconds of data and classification of patients based on cognitive function. They aim to investigate the existing feature engineering techniques by using the 10-second dataset and the two subsets of 5-second and explore potential improvements, and evaluate the performance of the Random Forest model.

2 Related works

In [19], the authors perform an extensive literature review, analyzing a wide range of papers focused on automated machine learning (AutoML) techniques and their potential applications in healthcare. The objective is to explore improvements in AutoML and consider how these techniques can be leveraged to address healthcare challenges such as disease diagnosis. The review presents several noteworthy findings. Firstly, various AutoML approaches are identified, including automated feature engineering, hyperparameter optimization, pipeline optimizer, and neural architecture search. The advantages and limitations of each approach are presented while providing information about the effectiveness of various healthcare tasks. Furthermore, it shows how automated techniques can help select features and optimize models for better accuracy and efficiency. It also shows how AutoML can improve patient care and medical decision-making. However, it is important to recognize that there are challenges in implementing AutoML in healthcare settings, such as the interpretability of automated models, and the need for domain-specific knowledge integration. These challenges need more research and development. Overall, this paper makes a substantial contribution to the field of automated machine learning in healthcare.

In [12], the authors used an automated machine learning pipeline to evaluate the cognitive function of PD patients. The pipeline consists of feature extraction (i.e., ts-fresh), feature selection (i.e., Boruta), Random Forest classifier and hyperparameter optimization. The patients have been classified into two classes "good cognition" and "poor cognition". Five epochs were selected from the EEG of each patient which contains the recorded results of 21 electrodes. These records of electrodes are time series used as input in the machine learning pipeline. Then, the ts-fresh method was used to extract features and the boruta method to select the most important features. After that, the selected features were used to train the model, and a hyperparameter method, namely Mixed-integer Parallel Efficient Global Optimization, was performed to optimize the hyperparameters of the model. By using the automated extract method for the features the model achieved an accuracy of 84%, while the extracted features in combination with clinical features increased the accuracy to 91%.

In the context of the same project similar to the research above in [8], the same steps of an automated EEG-based machine-learning pipeline were performed. The data come from patients who suffer from Parkinson's Disease during Deep Brain Stimulation screening. There are six neuropsychological domains. The test scores of all patients were standardized and the average per domain was computed. The number of patients was 112, from them 20 were selected with a high cognitive score, 20 were selected with a low cognitive score, and the remaining people were selected with intermediate cognitive performance. It also used 21 electrodes that are the global peak frequencies, while the global was compared with occipital peak frequencies. By using as a dataset the global peak frequencies in the pipeline and applying cross-validation the accuracy was 0.92. On the contrary, the usage of occipital peak frequencies gave a lower accuracy of 0.67.

In [10], electromyography (EMG) time series were used in an automated machine learning pipeline to predict whether a person is healthy or not. The dataset consisted of 65 subjects, of which 40 have Inclusion Body Myositis (IBM) and amyotrophic lateral sclerosis (ALS), while 25 are healthy. This is a supervised classification problem and the classification labels are Diseases

for IBM and ALS patients and CTRL for healthy subjects. The automated machine-learning pipeline of this work follows the same steps mentioned above.

In this project, the aim is to compare two different methods of feature engineering namely ts-fresh combined with boruta and Catch22. Moreover, a dataset of 10 seconds and two subsets of 5 seconds was used to evaluate those methods and find if one epoch with fewer sizes can give reliable results compared to the average of features of five epochs with fewer sizes based on the analysis of EEG time series.

3 Methodology

3.1 Dataset

The raw data consists of the results of an electroencephalography(EEG) examination in 127 patients. Figure 1 illustrates how the electrodes are placed on a patient's scalp for the examination. The dataset used contains five epochs of each patient and each epoch represents ten(10) seconds of the examination. Moreover, each epoch consists of twenty-one(21) time series where each time series reflects the difference in electric potential between an electrode and a reference electrode [3], in this case, a common reference is the electrode *Cz*. Furthermore, each patient with Parkinson's disease(PD) is labelled as having "good cognition" or "poor cognition". The raw data was obtained by the Leiden University Medical Center (LUMC).

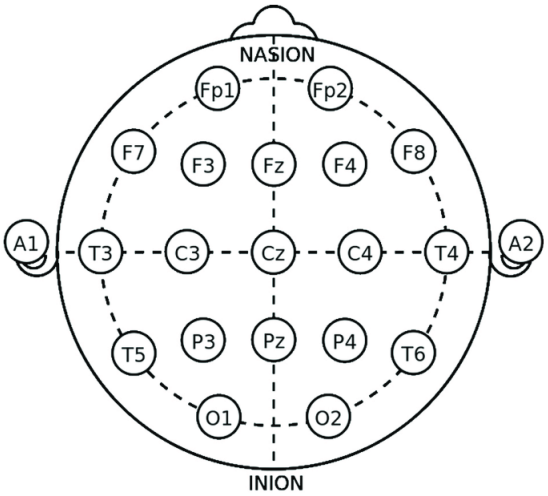


Figure 1: Imaging electrodes on the patient's head [15]

3.1.1 Pre-processing dataset

Missing values handling

The initial dataset as mentioned above consisted of the patient's epochs (i.e., patient id and patient time series) and the patient's labels (i.e., patient id and labels). At first, it was observed that the labels were referring to 60 patients while the total number of patients was 127. To arrive at the final dataset the intersection of these two subsets was taken based on patient id. The patient's epochs had 127 patients of which 80 patients were discarded because there were no labels for them. This results in 47 patient epochs (i.e., 127 minus 80 with no labels) and 47 patient labels (i.e., 60 minus 13 without patient epochs).

Moreover, the raw data consisted of five epochs for each patient. Each epoch contains the measurements of 21 time series which are the measurements of the electrodes that were used to perform the examination and the duration of each epoch is 10 seconds of examination.

Choice of epoch length

Table 1 displays the data format utilized in this project. The first data set consists of five 10-second epochs of the examination clinicians have assessed that contain fewer artifacts and it is the raw data. Each epoch consists of 21 time series, with time series containing 5000 data points, which serves as the length of the time series. Two additional subsets, named Part 1 and Part 2, were generated from the first dataset of 10 seconds by splitting the dataset in the middle, resulting in two 5-second subsets. Each epoch within these subsets also contains 21 time series but the length of the time series is 2500 data points for both subsets.

	EEG cases		**Labels**			
Name of set	**Patients**	**Epochs**	**Class 0**	**Class 1**	**Dimension**	**Length**
10 seconds	47	5	20	27	21	5000
5 seconds (Part 1 and Part 2)	47	5	20	27	21	2500

Table 1: Overview of data format used as input in feature engineering

Table 2 contains the four approaches taken into account to execute the experiments. These approaches were developed based on the data format in Table 1 which contains the seconds of each epoch of the EEG and the average of features of those epochs. More specifically, **Ind-10s approach** includes features of each individual epoch of 10 seconds from the EEG, while the average of features from five 10-second epochs is contained in **Avg-10s approach**. Features from each individual 5-second epoch of the examination, including both Part 1 and Part 2, are contained in **Ind-5s approach** and **Avg-5s approach** includes the average of features of five epochs of 5 seconds (Part 1 and Part 2). In Approaches Ind-5s and Avg-5s, the subsets of 5 seconds were utilised. These approaches were applied to execute the experiments by using two different feature engineering techniques to extract and select features, namely ts-fresh combined with boruta, and Catch22.

				Labels	
Approach	**Description**	**Patients**	**Epochs**	**Class 0**	**Class 1**
Ind-10s	Automated Features 10sec (each epoch)	47	5	20	27
Avg-10s	Automated Features 10sec (averaged)	47	Averaged Features of epochs	20	27
Ind-5s	Automated Features 5sec (each epoch)	47	5	20	27
Avg-5s	Automated Features 5sec (averaged)	47	Averaged features of epochs	20	27

Table 2: Overview of the Approaches for executing experiments with Automated Machine Learning Pipeline, namely Individual 10s Features (Ind-10s), Averaged 10s Features (Avg-10s), Individual 5s Features (Ind-5s), Averaged 5s Features (Avg-5s)

3.2 Machine learning pipeline

The pipeline used in this project was created for application in the automotive industry, for time series classification tasks with vehicle on-board data [11], [13]. Moreover, it has been used to predict the cognitive function of patients who suffer from PD by using EEG [12]. It has also been applied to EMG (electromyography) to limit arbitrary choices by using EMG time series from healthy people and patients with either myopathic or neuropathic diseases [10]. Figure 2 illustrates the steps of the automated machine learning pipeline used to execute the experiments.

The three main steps of the pipeline are:

1. Feature engineering (Extracted features and Selected features)
2. Modelling
3. Hyper-parameter optimization

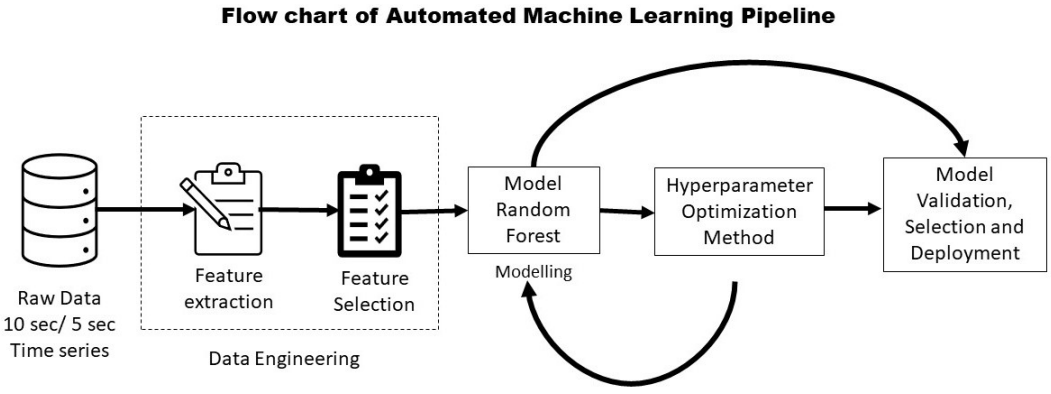


Figure 2: Illustration of the process of performing the experiments by using the Automated Machine Learning pipeline

3.2.1 Feature Engineering

Feature engineering contains the extraction of meaningful features from raw data and their transformation into formats optimized for utilization by machine learning models [20]. Feature engineering is the procedure of extracting and selecting features of each time series. Here, two different engineering features were used. The first is ts-fresh fused with Boruta, and the latter is Catch22. They are described in detail below.

Ts-fresh and Boruta

Feature extraction is an important procedure for time series analysis. Selecting and calculating features from time series is time-consuming and needs the expertise of specialists to extract meaningful features from time series. However, there is a python package, ts-fresh (Time Series FeatuRe Extraction based on Scalable Hypothesis tests) which is a faster method that uses 63

time series characterization methods [5] and extracts more than 750 features[2] for each time series.

Ts-fresh was applied to extract 16528 features from each time series of the patients. However, it was required to verify that the same features were present in all files. Therefore, a script was created to compare the features across the files and identify any discrepancies. After that, the data frame was checked for *nan* values and found that 85 features had missing values. It was decided to drop these features from the data frame. As a result, the total number of features was 16443, which means that 783 features were extracted from each time series. It is observed that the number of extracted features is different from the default number based on the research [5], i.e., 794. This can be due to factors such as the length of the time series or the characteristics of the data. However, the number of features was very high and a feature selection method was used to reduce the dimensionality of the dataframe.

Hence, those extracted features were utilised as input in the Borura algorithm to select the most important features. To perform features selection the library BorutaPy was used. Boruta is based on the Random Forest method. This method is quick and does not require tuning the parameters and it produces an estimation for the most important features. Boruta generates shadow features using the random shuffle method from the extracted features. It then compares the extracted features with the shadow features and checks if the extracted feature is better, in which case the algorithm keeps it [14].

Catch22

Canonical Time-series Characteristics (Catch22) is a method that generates twenty-two features from a time series. In this project, a Python package of pyCatch22 was used to provide a collection of time-series features. These features are designed to be useful for time series classification tasks.

The main idea is to extract features from a time series that capture different aspects of its shape, dynamics, and distribution. The chosen features are robust, interpretable, and computationally efficient[4]. The package contains a set of features that span through different aspects of time series, including their autocorrelation, distributions and outliers, complexity, fluctuation scaling aspects, and stationarity[4].

The features are computed using a combination of statistical and machine-learning techniques. These techniques evaluate the univariate performance, combine feature scores, and select a reduced set of important features which is useful in accordance with two steps of the filtering process, namely performance filtering and redundancy minimization [4]. A time series is provided as input in Catch22 and the Python package computes a feature vector, i.e., 22 features as output.

To extract features from each time series in this project, a function was applied to compute 22 different statistics for each time series. This resulted in a set of 21 rows which is the number of time series, each with 22 features, for each patient. However, this format was not suitable for the analysis, since a single row of 22 features for each patient is needed. Moreover, the

[2]https://nils-braun.github.io/tsfresh-on-cluster-1/

time series varied widely in their values and using the mean or median across them to reduce the dimensionality was not feasible.

For this reason, the time series from the different brain regions were employed, specifically the frontal, temporal, central, parietal, and occipital regions, which correspond to the electrode placements. The names of the electrodes for each region are listed in Table 3. Data originating from these electrodes were the time series that were used in this project to extract features. Each region provided a set of time series, with 22 features extracted for each time series. Then, the average of these features was calculated across all time series in each region to create a final set of 22 features. For instance, the occipital region consists of two electrodes, namely O1 and O2, the referred time series of these electrodes are "*O1-Cz*" and "*O2-Cz*". For each time series, 22 features were extracted, resulting in two sets of 22 features. After that, the average of each feature across the two time series was computed to create a single set of 22 features for the occipital region. This process was repeated for each region and then the resulting sets of features were concatenated into a single set of features for each patient, with 5 rows (one for each region of the brain) and 22 columns (one for each feature). As a result, the dimensionality of extracted features from Catch22 decreased. Table 4 shows the 22 extracted features of the Catch22 method [4] for each time series.

Regions	**Electrodes**
Frontal	F3, F4, F7, F8, Fz
Temporal	T3, T4, T5, T6
Central	C3, C4, Cz
Parietal	P3, P4, Pz
Occipital	O1, O2

Table 3: Five different regions of the brain and the electrodes of each region

Then, two different methods were employed to perform the experiments as they are illustrated in Table 5. The first method is Catch22-Comprehensive which consists of an array for each patient that contains all features from the five regions i.e., 110 (5x22)features. The second method is Catch22-Compact where the arrays of all patients of the previous steps were concatenated into a dataframe and a filtering method was employed to select the most important features. To perform the filtering method, from the library "sklearn.feature_selection", the function "selectKBest" was used. This function creates an object by using as a score function the "f_classification", this function calculates the ANOVA F-value for the features which is the degree of linear dependency between two features. Due to the fact that Catch22 generates 22 features, the 22 most important features were selected from the 110 features of the Catch22-Comprehensive method. After that, the selected features were used as input to Random Forest Classifier model to train the model.

No	Feature names
1	DN_HistogramMode_5
2	DN_HistogramMode_10
3	CO_f1ecac
4	CO_FirstMin_ac
5	CO_HistogramAMI_even_2_5
6	CO_trev_1_num
7	MD_hrv_classic_pnn40
8	SB_BinaryStats_mean_longstretch1
9	SB_TransitionMatrix_3ac_sumdiagcov
10	PD_PeriodicityWang_th0_01
11	CO_Embed2_Dist_tau_d_expfit_meandiff
12	IN_AutoMutualInfoStats_40_gaussian_fmmi
13	FC_LocalSimple_mean1_tauresrat
14	DN_OutlierInclude_p_001_mdrmd
15	DN_OutlierInclude_n_001_mdrmd
16	SP_Summaries_welch_rect_area_5_1
17	SB_BinaryStats_diff_longstretch0
18	SB_MotifThree_quantile_hh
19	SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1
20	SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1
21	SP_Summaries_welch_rect_centroid
22	FC_LocalSimple_mean3_stderr

Table 4: Overview of 22 extracted features of Catch22

Method	Description	Dimension	Length
Catch22-Comprehensive	Extracted features from 5 regions electrodes	110	47
Catch22-Compact	Extracted features using filter method	22	47

Table 5: Overview of the methods for the extracted features from the Catch22 method.

3.2.2 Random Forest classifier

Random forest is an ensemble learning method that uses many decision trees to do the prediction. Each decision tree is trained by using a subset of the training set and features. After that, the results of all decision trees are gathered and the class with the most votes is the result of the model [9]. This model is simple and efficient and is used in different domains while achieving good performance. This model also uses the hyperparameters of the Decision Tree classifier and Bagging Classifier [9].

Here, the selected features were used as input in the Random Forest model. This model was applied by using the function "RandomForestClassifier" from the scikit-learn (sklearn) library. The parameters that were selected to be used were *n_estimators*, *max_depth*, *bootstrap*, *max_features*, *min_samples_leaf*, *min_samples_split*. As mentioned above, the Random Forest

consists of several decision trees. Each decision tree is independent of the remaining ones. By using the Random Forest algorithm the overfitting is reduced by using the majority ranking of the decision trees [1].

For the experiments, the extracted features from five individual epochs of 10 seconds and 5 seconds were used, as well as the average of the features of all epochs for the seconds referred to earlier.

3.2.3 Hyperparameter Optimization

To achieve a good performance of the model, the best hyperparameters should be selected. There are several methods to optimize the parameters, such as Grid Search, Evolutionary Algorithm, and Bayesian Optimization [9]. Firstly, Table 6 indicates the names of the hyperparameters of the Random Forest model and the range of values examined to choose the best values for the model.

Parameter	**Values**
max_depth	none, 2, 4,...,100
n_estimators	1,2,....1000
min_samples_leaf	1,2,....10
min_samples_split	2,3...,20
bootstrap	True, False
max_features	auto, sqrt, log2

Table 6: The range of hyperparameters to optimize the Random Forest Classifier

The Bayesian Optimization algorithm was selected to optimize the model. This algorithm uses intelligence to select the next set of hyperparameters that will improve the model. This method is repeated until it converges to an optimum. The Bayesian Optimization is preferable to tune the hyperparameters of a well-performing model on the validation dataset.

However, in Table 6, parameters containing categorical and integer values exist. For this reason, Mixed-integer Parallel Efficient Global Optimization (MIP-EGO) [17], [18], is selected to handle mixed variables namely integer and categorical. This method is efficient in optimizing expensive problems. This algorithm proposes a candidate hyperparameter setting at each iteration. The number of iterations executed in the experiments is 200 iterations.

Subsequently, this setting is evaluated by measuring the performance of the model using the test set. In order to tune the hyperparameters, a k-fold cross-validation setup is used, where k represents the number of folds, here 10-fold. Cross-validation divides the data into ten folds. During this process, the model is trained and evaluated ten times, with each fold serving as a validation set while the remaining folds are utilized for training. Notably, the model's evaluation takes place when the hyperparameter optimization method is applied within each test set, representing a single fold of cross-validation.

3.2.4 Evaluation

Two metrics were used to evaluate the model, namely accuracy and F1-score. Accuracy is the number of correct predictive classes divided by all samples. The accuracy is computed by using the formula:

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

where

- TP is True Positive,
- TN is True Negative,
- FP is False Positive
- FN is False Negative

F1-score is the harmonic mean of the precision and recall. It is used in binary classification problems where the labels are "positive" and "negative" [1]. F1-score is computed by using the formula:

$$f1 = 2 \times \frac{precision \times recall}{precision + recall}$$

.
Precision measures how many of the predicted classes as positive are predicted correctly by the model while recall measures how many of the positive samples in the dataset are correctly captured by the model [1].

However, the accuracy works well when the dataset is class balanced i.e., the number of class 1 and class 2 are the same. Whilst, the F1-score takes into account Precision and Recall, it is more appropriate for imbalanced binary classification problems than accuracy.

4 Results and Discussions

This section consists of two subsections that provide a detailed description of the experiments and their corresponding results. Two distinct feature engineering techniques were applied, utilizing three different sets of data. The first dataset consists of five epochs of 10 seconds and each epoch is part of a complete EEG examination. Additionally, there are two subsets consisting of five epochs of 5 seconds that were derived by dividing the aforementioned initial dataset in half.

Consequently, a total of three sets of data were used for conducting the experiments. For each approach, namely Ind-10s, Avg-10s, Ind-5s, and Avg-5s, five independent runs have been conducted for each feature engineering technique. Below the experiments of each feature engineering technique are described in detail.

4.1 Using Machine Learning pipeline with ts-fresh and Boruta

Firstly, the Ind-10s approach consists of 10 seconds dataset and was used for feature extraction using ts-fresh. 16443 features were extracted and used as input to Boruta to select the most important features. Similarly, in the Ind-5s approach, the dataset was divided into 5 seconds subsets (Part 1 and Part 2) and the same feature extraction and selection process was repeated for each subset.

Subsequently, the experiments were performed in each epoch separately. The number of experiments performed was five (5) executions. Table 7, shows the features selected more times, i.e., more than 4 times out of 5 experiments for Approaches Ind-10s and Ind-5s.

No	Selected Feature names
1	P4-Cz__fft_coefficient__attr_""imag""__coeff_64
2	Pz-Cz__partial_autocorrelation__lag_8
3	T3-Cz__fft_coefficient__attr_""imag""__coeff_71
4	A1-Cz__fft_coefficient__attr_""abs""__coeff_65
5	F7-Cz__fft_coefficient__attr_""angle""__coeff_4
6	O2-Cz__fft_coefficient__attr_""angle""__coeff_25
7	O2-Cz__fft_coefficient__attr_""real""__coeff_53
8	P4-Cz__fft_coefficient__attr_""angle""__coeff_64
9	T6-Cz__fft_coefficient__attr_""angle""__coeff_88
10	T6-Cz__fft_coefficient__attr_""imag""__coeff_88
11	T3-Cz__fft_coefficient__attr_""imag""__coeff_39

Table 7: Overview of the features selected more times during the experiments for Approaches Ind-10s and Ind-5s by using ts-fresh combined with Boruta.

The above-mentioned features (Table 7) adhere to the following naming format:

$\{time\ series\ name\}\_\{feature\ name\}\_\{parameter\ 1\}\_\{value\ 1\}\_\{parameter\ 2\}\_\{value\ 2\}$

where "time series name" represents a different time series of the dataset, "feature name" represents a different ts-fresh feature [5] and parameters and values show the different choices

of the parameters [5]. For instance, feature "P4-Cz__fft_coefficient__attr_""imag""__coeff_64", refers to P4-Cz time series, the ts-fresh feature is "fft_coefficient", and the first parameter is "attr" and its value is "imag" while the second parameter is "coeff" with value "64". The majority of selected features were related to the FFT coefficient (Fast Fourier Transform coefficient). FFT coefficient algorithm computes the Fourier coefficient of the one-dimensional discrete Fourier Transform for real input [5]. The FFT coefficients are used as a set of features to represent the frequency content of the time-series signal [5].

The experiments of Approaches Avg-10s and Avg-5s were performed by averaging the features of all five epochs. The experiments of those approaches were also executed five times. Table 8 indicates the features selected more times, i.e., more than 2 times out of 5 experiments for Approaches Avg-10s and Avg-5s. The format of the extracted features is the same as previously described.

No	Selected Feature names
1	P3-Cz__cwt_coefficients__coeff_0__w_2__widths_(2, 5, 10, 20)
2	T6-Cz__fft_coefficient__attr_""imag""__coeff_15
3	C3-Cz__fft_coefficient__attr_""abs""__coeff_99
4	T3-Cz__fft_coefficient__attr_""imag""__coeff_72
5	A1-Cz__fft_coefficient__attr_""real""__coeff_73
6	A2-Cz__ar_coefficient__coeff_7__k_10
7	F4-Cz__fft_coefficient__attr_""angle""__coeff_17
8	F8-Cz__ar_coefficient__coeff_8__k_10
9	F8-Cz__fft_coefficient__attr_""angle""__coeff_66
10	Fp1-Cz__cwt_coefficients__coeff_3__w_5__widths_(2, 5, 10, 20)
11	P4-Cz__fft_coefficient__attr_""imag""__coeff_97
12	Pz-Cz__fft_coefficient__attr_""real""__coeff_15
13	Pz-Cz__fft_coefficient__attr_""real""__coeff_89
14	T4-Cz__fft_coefficient__attr_""angle""__coeff_15
15	T5-Cz__index_mass_quantile__q_0.7
16	T5-Cz__index_mass_quantile__q_0.8

Table 8: Overview of the features selected more times during the experiments for Approaches Avg-10s and Avg-5s by using ts-fresh combined with Boruta.

Table 9 displays the results of feature selection for the 10-second dataset and the two 5-second subsets (referred to as Part 1 and Part 2). The feature selection was conducted using both the extracted features of individual epochs and the average of extracted features of all epochs to conduct the experiments. Table 9 illustrates the range of selected features obtained from the Boruta algorithm after five separate runs. Notably, the Boruta algorithm selected a different number of selected features in each execution. A noteworthy observation is that the 10-second dataset has selected more features in both individual epochs and the average of epochs. For instance, Epoch 3 has a range from 1 to 11 features during the execution of experiments which means that the Boruta algorithm selected one feature in one execution and 11 features in another execution out of the five runs. Moreover, the average of features of all epochs has a

Range of selected features			
Epochs	**10 sec**	**5 sec (Part 1)**	**5 sec (Part 2)**
1	1-7	5-9	1-3
2	1-6	2-7	4-10
3	1-11	4-10	4-13
4	2-6	1-6	1-6
5	2-4	2-5	2-10
averaged epochs	2-11	1-5	1-6

Table 9: The range of selected features by using the Boruta method for features of each epoch and the average of features of all epochs by using the datasets of EEG examination after 5 runs.

range from 2 to 11. Conversely, the 5-second subsets (Part 1 and Part 2) selected more features in the individual epochs than the average of epochs during the five execution of experiments. It is worth noting that out of five epochs, the two subsets selected more features in three of them. The range of selected features varied in Part 1, ranging from 5 to 9 for Epoch 1, ranging from 2 to 7 for Epoch 2, and ranging from 4 to 10 for Epoch 3, while Part 2 exhibited a range from 4 to 10 in Epoch 2, a range from 4 to 13 in Epoch 3, and a range from 2 to 10 in Epoch 5.

		Epochs				
Approach	**metric**	**1**	**2**	**3**	**4**	**5**
Ind-10s	Accuracy	86.8 ± 7.9	87.4 ± 7.9	91.9 ± 6.5	91.7 ± 7.9	87.8 ± 5.1
	F1-score	86 ± 8.4	86.5 ± 8.5	91.5 ± 6.8	91.3 ± 8.4	87.38 ± 5.3
Ind-5s (Part 1)	Accuracy	95.3 ± 2.7	90.8 ± 3.7	95.9 ± 4.5	86.1 ± 7.8	89.7 ± 5.2
	F1-score	95.1 ± 2.9	90.3 ± 3.9	95.8 ± 4.7	85 ± 8.4	89.2 ± 5.4
Ind-5s (Part 2)	Accuracy	84.9 ± 4.6	96.2 ± 3.1	96.6 ± 3.1	89.3 ± 5.2	91.8 ± 4.1
	F1-score	83.9 ± 5.3	96 ± 3.2	96.5 ± 3.2	88.7 ± 5.5	91.5 ± 4.3

Table 10: Performance scores of **Ind-10s approach** and **Ind-5s approach** for each epoch separately by using ts-fresh and boruta for features extraction and selection. The score of metrics is computed on the test set and averaged in a 10-fold cross-validation. The mean and standard deviation are the averages of the 5 executions of the experiments.

Table 10 and Table 11 present the results of the model evaluation metrics, specifically accuracy and F1-score. These tables showcase the average values of the mean and standard deviation across the five experiment executions. Table 10 demonstrates that the accuracy of the Ind-10s approach among the individual epochs fluctuates between $86,8\%$ with a standard deviation of 7.9% and 91.9% with a standard deviation of 6.5%. In comparison, the Avg-10s approach (Table 11) achieves an accuracy of 92.3% with a standard deviation of 8.1%. On the contrary, in Table 10, the Ind-5s approach has better accuracy among the individual epochs in relation to the Avg-5s approach (Table 11) which has lower accuracy for the average of epochs. Moreover, the Ind-5s approach shows larger accuracy in comparison to the Ind-10s approach

among the individual epochs, while the Avg-10s approach has better accuracy with regard to the Avg-5s approach. It is additionally worth noting that Part 2 of 5 seconds of data has greater accuracy in comparison to Part 1 of 5 seconds among the individual epochs and the average of epochs. It is interesting to observe that the individual epochs with a large accuracy have selected more features during the experiments. For instance, the Ind-10s approach has an accuracy of 91.9% in Epoch 3 which is the highest among the individual epochs and the range of selected features in this epoch fluctuates from 1 to 11 during the experiments which is also the highest amount of selected features.

Another metric which plays a significant role in the project is the F1-score which is the most appropriate metric for the imbalanced dataset used in this study. Table 10 indicates that there is a fluctuation among the individual epochs in Approaches Ind-10s and Ind-5s. F1-score fluctuates between 86% and 91.5% among the individual epochs of the Ind-10s approach, while the range of the F1-score is between 83.9% and 96.5% for the Ind-5s approach (Part 2). However, it is observed that Epoch 3 has a better performance in both approaches. Table 11 shows that the Avg-10s approach has better performance with a percentage of F1-score of 92% and a standard deviation of 8.5%, while the Avg-5s approach has a lower F1-score with 86.3% and 91.2% for Part 1 and Part 2 respectively. F1-score also has slightly better results in Part 2 of 5 seconds subset in relation to Part 1 of 5 seconds subset in each epoch and the average of epochs.

Approach	**metric**	**Averaged epochs**
Avg-10s	Accuracy	92.3 ± 8.1
	F1-score	92 ± 8.5
Avg-5s (Part 1)	Accuracy	86.8 ± 9
	F1-score	86.3 ± 9.3
Avg-5s (Part 2)	Accuracy	91.8 ± 7.2
	F1-score	91.2 ± 7.9

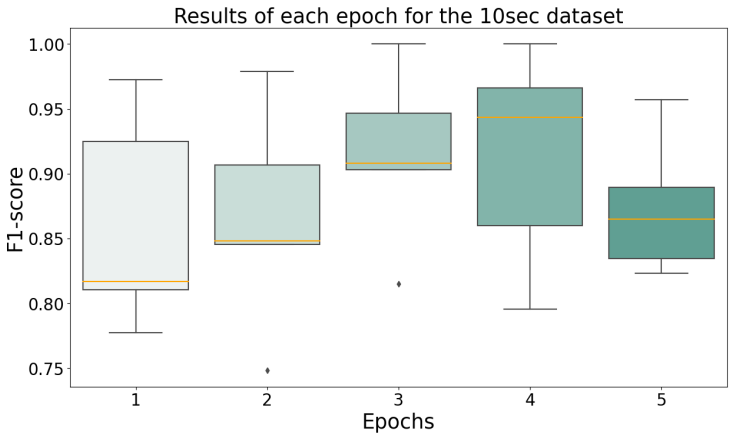
Table 11: Performance scores of **Avg-10s approach** and **Avg-5s approach** for the average of epochs by using ts-fresh combined with boruta for features extraction and selection. The score of metrics is computed on the test set and averaged in a 10-fold cross-validation. The mean and standard deviation are the averages of the 5 executions of the experiments.

Furthermore, the Ind-5s approach (Part 1 and Part 2) in Table 10 demonstrates superior F1-score results for each epoch. Part 1 has higher performance in epochs 1 and 3, surpassing the percentage of 95% with a standard deviation of 2.9 and 4.7 respectively. On the other hand, Part 2 achieves better results in epochs 2 and 3, exceeding the percentage of 96% with a standard deviation of 3.2% in both epochs. By comparing the range of extracted features from Table 9 with F1-scores in Table 10 and Table 11 it is noticed that epochs with a larger range of selected features have better F1-scores than the epochs with fewer selected features.

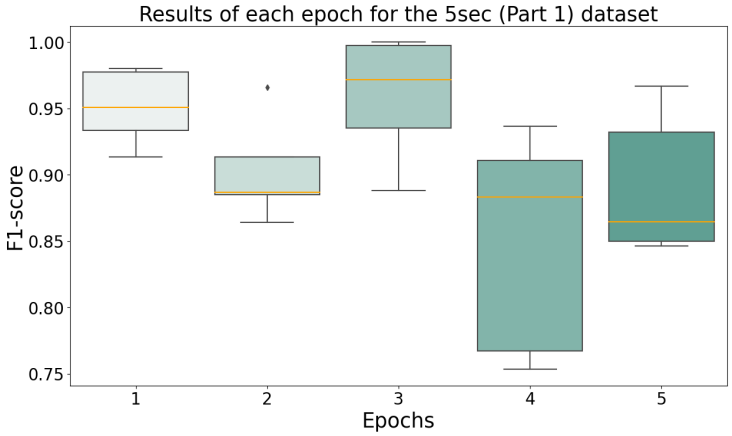
Figure 3 shows the visual representations of the distributions of the five execution of the experiments for the Approaches Ind-10s and Ind-5s (Part 1 and Part 2). The vertical axis represents the values of the F1-score, while the horizontal axis indicates the individual epochs. Figure 3a

shows five box plots where each box plot corresponds to a specific epoch, namely 1, 2, 3, 4, and 5. Epochs 1 and 4 have large distributions which means that the results of experiments are more dispersed. However, Epoch 4 performs better than Epoch 1 because the median of Epoch 4 is 91.3% while the median of Epoch 1 is almost 82. Moreover, the distribution of Epoch 3 has shorter distribution than the other epochs, however, the median is low in the inter-quartile range. Comparing Epoch 3 and Epoch 4 in the Ind-10s approach, it is observed that the distribution of Epoch 3 is slightly better than Epoch 4 where the 50% of the distribution is spread between almost 86% and 97%.

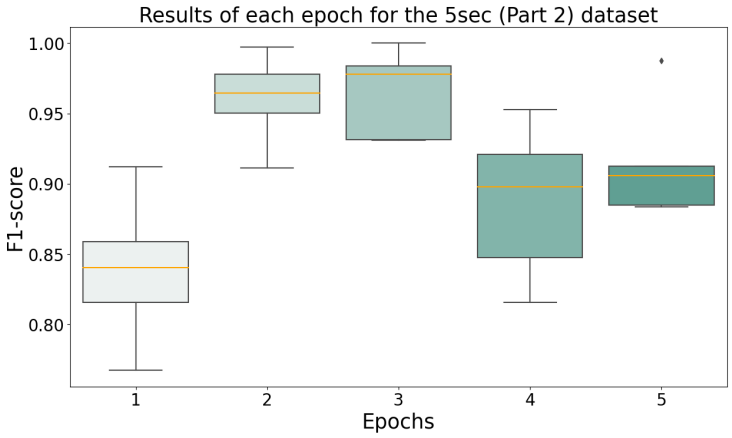
Then, by comparing Figure 3b and Figure 3c, the Ind-5s approach of Part 2 has better distributions among the individual epochs than the Ind-5s approach of Part 1. Figure 3b depicts that three out of five epochs have better distributions, namely epochs 1, 2 and 3, than epochs 4 and 5. On the other hand, Figure 3c shows better distributions for epochs 4 and 5 in relation to Figure 3b where the F1-score of the five experiments is more dispersed. However, both parts of the Ind-5s approach have a good distribution in Epoch 3, but the Ind-5s approach of Part 2 is better because the interquartile range is shorter and the median is close to the upper quartile. Moreover, Epoch 2 of Part 2 has a good distribution because the results of experiments are less dispersed and the median is in the middle of the box plot. However, the maximum and minimum values have a large range. Yet, Epoch 3 has a higher median than Epoch 2 in Part 2 of the Ind-5s approach and based on Table 10 the F1-score is slightly different between these two epochs.



(a) **Ind-10s approach:** F1-score for each epoch of the 10sec dataset



(b) **Ind-5s approach:** F1-score for each epoch of the 5sec data (Part 1)



(c) **Ind-5s approach:** F1-score for each epoch of the 5sec data(Part 2)

Figure 3: Visualization of the results of Approaches Ind-10s and Ind-5s using ts-fresh and Boruta

Figure 4 illustrates the results of experiments for Approaches Avg-10s and Avg-5s which are the averages of the features from five epochs for the dataset of 10 seconds and 5 seconds (Part 1 and Part 2) respectively. Comparing these results, it becomes evident that the Avg-10s approach outperforms the Avg-5s approach for both Part 1 and Part 2. The F1-score values obtained from the experiments exhibit less variability in the case of the Avg-10s approach. The box plot of the Avg-10s approach illustrates a compact distribution with the median positioned close to the centre, indicating relatively consistent performance. The range between the maximum and minimum values is also limited, further emphasizing the stability of the results. Conversely, the distribution of Avg-5s (Part 1) demonstrates a wider range, suggesting a more diverse spread of values. However, the distribution of Part 2 shows a narrower range, with the majority of values concentrated in the third quartile. This concentration is indicated by the median value being situated in this particular area of the box plot.

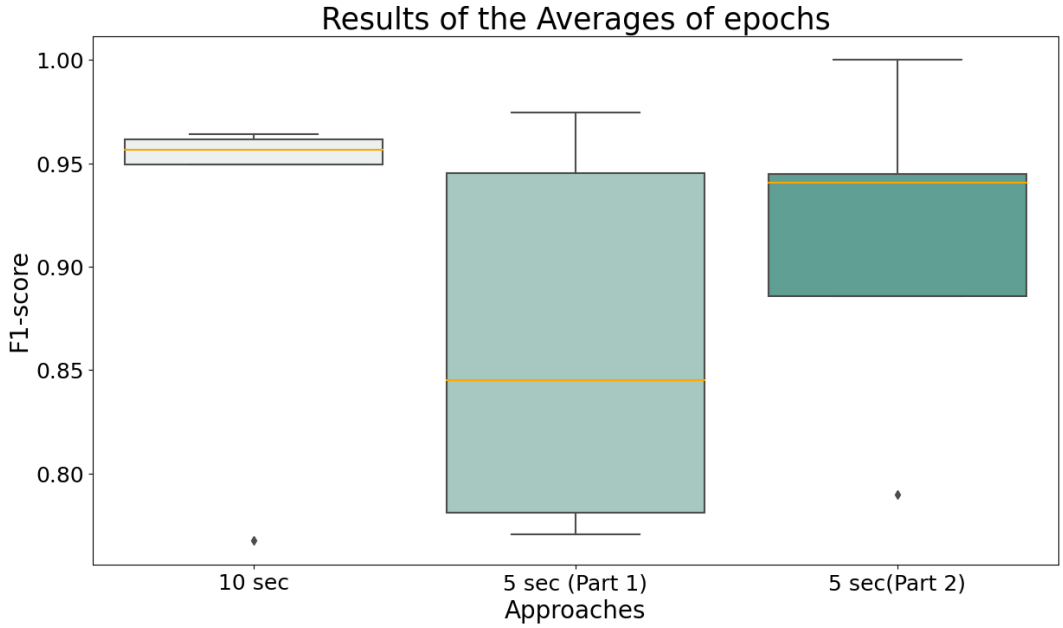


Figure 4: Visualization of the results of **Avg-10s approach** and **Avg-5s approach** using ts-fresh and Boruta

In conclusion, after the experiments, the most robust data set is observed in the Avg-10s approach. The average of features for all epochs gives better results in both accuracy and F1-score with values of 92.3% and 92% respectively for the dataset of 10 seconds. The difference in metrics is small.

Subsequently, it is observed that the best results of the Ind-5s approach for the data of 5 seconds, namely Part 1 and Part 2, were given by Part 2. In Part 2, the individual epoch distributions have a smaller range compared to Part 1. Averaging the epochs in Figure 4 also indicates a good distribution in Part 2 compared to Part 1 where the distribution was spread over a larger range.

4.2 Using Machine Learning pipeline with Catch22

As before, the same approaches were employed to execute the experiments using a different feature engineering technique. The data of 10 seconds and 5 seconds (Part 1 and Part 2) were examined to observe how the model performs. Catch22 is a technique that extracts and selects 22 features from each time series. In this case, the experiments were performed separately in two methods as indicated in Table 5.

4.2.1 Catch22-Comprehensive method

The dataset of 10 seconds and the two subsets of 5 seconds of 47 patients with PD was utilised to extract 22 features from each time series. To distinguish the time series, the electrodes from each side of the brain were utilised. The 22 features were obtained by averaging the values of the electrodes within the same brain region, ensuring that each feature represented a specific region of the brain (Table 3). For each epoch, a set with 5 rows, which are the five different regions of the brain, and 22 columns, which are the extracted features, was generated. Then, for each patient, an array (1x110), is created by concatenating all values of the previous set. Finally, a dataframe was created by concatenating the arrays of all patients (i.e., 47x110) and was used as input in the model.

		Epochs				
Approach	**metric**	**1**	**2**	**3**	**4**	**5**
Ind-10s	Accuracy	96.4 ± 8	93.5 ± 8.5	96.1 ± 7.4	95.8 ± 9.3	99.2 ± 1.8
	F1-score	95.8 ± 9.3	92.6 ± 9.8	95.5 ± 8.5	95.4 ± 10.2	99.1 ± 2
Ind-5s (Part 1)	Accuracy	91.5 ± 5.3	97.2 ± 4	86.8 ± 7.6	96.6 ± 3.3	89.7 ± 6.6
	F1-score	90.6 ± 6	97 ± 4.4	85.2 ± 9.4	96.4 ± 3.6	88.7 ± 7.4
Ind-5s (Part 2)	Accuracy	88.8 ± 12	89.3 ± 7.8	97.7 ± 1.6	99.7 ± 0.5	96.4 ± 4.2
	F1-score	87.5 ± 13.2	87.8 ± 9.3	97.5 ± 1.7	99.7 ± 0.5	96.1 ± 4.6

Table 12: Performance scores of **Ind-10s approach** and **Ind-5s approach** for each epoch separately by using 110 selected features. The score of metrics is computed on the test set and averaged in a 10-fold cross-validation. The mean and standard deviation are the averages of the 5 executions of the experiments.

Table 12 and Table 13 present the performance of the pipeline for different approaches using 110 features. The performance was measured by two metrics: Accuracy and F1-score. The F1-score of all approaches is slightly lower than accuracy but follows a similar trend. As shown in Table 12, Ind-10s approach achieves higher accuracy rates with more than 93% for all individual epochs than the Ind-5s approach (Part 1 and Part 2). The subset of 5 seconds (Part 2) of the Ind-5s approach ranges between 88.8% and 99.7%, while Part 1 of 5 seconds has slightly lower accuracy than Part 2. However, it is worth noting that the Ind-5s approach in Part 2 of 5 seconds performs better in epochs 3 and 4 with F1-scores 97.5% and 99.7% and low values of the standard deviation of 1.7% and 0.5% respectively. The F1-score of the Ind-10s approach fluctuates between 92.6% with a standard deviation of 9.8% and 99.1% with a standard deviation of 2%, across the individual epochs. The Ind-10s approach achieves the highest F1-score

in Epoch 5. On the other hand, the Ind-5s approach has an F1-score from 87.5% to 99.7% among epochs for the subset (Part 2) of 5 seconds. Considering the number of features (i.e., 110), the large percentages of the F1-score of individual epochs and the standard deviation in these epochs show that the model may overfit.

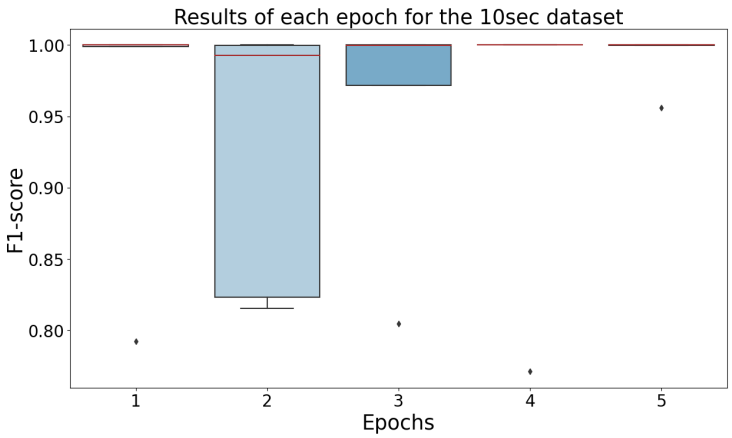
Table 13 indicates the results of Approaches Avg-10s and Avg-5s using the 110 extracted features of Catch22 feature engineering. To run the experiments of these approaches, the average of features for all epochs was used to train the model, and the results were from the test set. It is observed that the Avg-10s approach performs better as the accuracy and F1-score are 94.8% and 94.1% respectively than the Avg-5s approach. On the contrary, the results of the Avg-5s approach for the two subsets of 5 seconds are slightly low, Part 1 has an accuracy of 92.1% and F1-score 90.7%, while Part 2 has an accuracy of 87.1% and F1-score 85%. The results of the two approaches, i.e., the Avg-10s approach and the Avg-5s approach are compared in terms of the mean and the standard deviation of the averaged epochs. The Avg-10s approach has a higher mean and a lower standard deviation than the Avg-5s approach, indicating that it is more consistent and reliable. The Avg-5s approach has two parts, but both have larger standard deviations than the Avg-10s approach, meaning that they have more variation and uncertainty. The data of the Avg-10s approach are also closer to the mean than those of the Avg-5s approach, which shows that it had fewer outliers.

Based on the results of Table 12 and Table 13, it is noticed that in the Ind-5s approach, Part 2 of 5 seconds performed better in individual epochs and mostly in epochs 3 and 4 than Part 1. On the other hand, in the Avg-5s approach, Part 1 of 5 seconds has better results in the average of epochs.

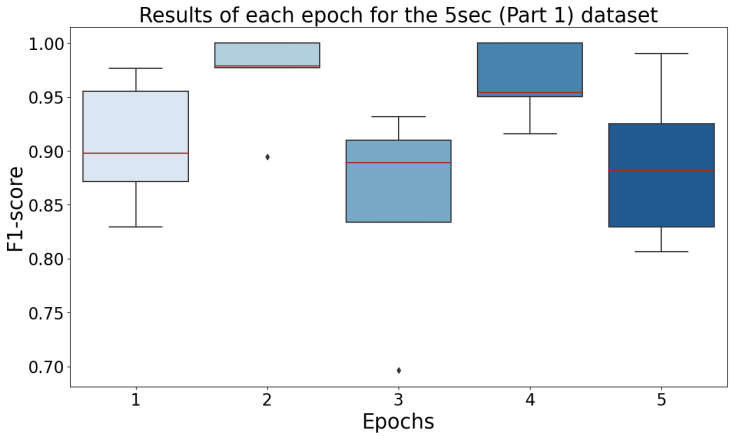
Approach	**metric**	**Averaged epochs**
Avg-10s	Accuracy	94.8 ± 7.4
	F1-score	94.1 ± 8.3
Avg-5s (Part 1)	Accuracy	92.1 ± 10.5
	F1-score	90.7 ± 12.5
Avg-5s (Part 2)	Accuracy	87.1 ± 8.3
	F1-score	85 ± 10

Table 13: Performance scores of **Avg-10s approach** and **Avg-5s approach** for the average of epochs by using 110 selected features. The score of metrics is computed on the test set and averaged in a 10-fold cross-validation. The mean and standard deviation are the averages of the 5 executions of the experiments.

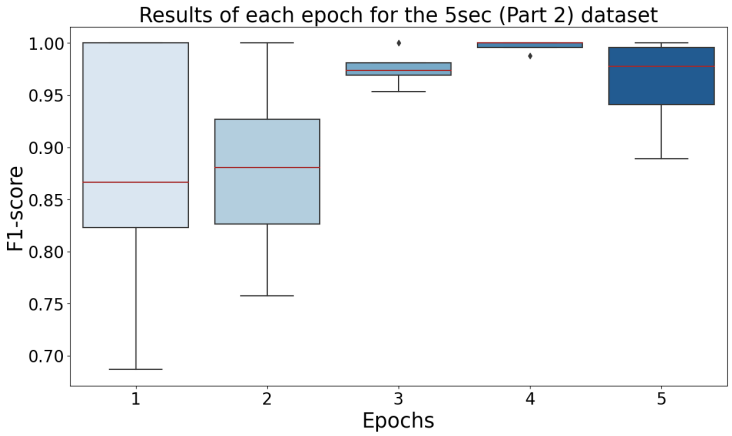
Figure 5 illustrates the results of the F1-score metric for Approaches Ind-10s and Ind-5s. Figure 5a shows the results of the Ind-10s approach for each epoch of the five executions of experiments. It is observed that there is an overfitting of the model because epochs 1, 4 and 5 are almost 100%. This may happen because of the large number of features, i.e., 110. However, the range of values of the F1-score in Epoch 2 is large which means that each iteration of the experiments gave a different F1-score which fluctuated from 100% to below 83%. Whilst the median of Epoch 2 is almost 99% which means that most values of the F1-score are found close to the third quartile of the box plot.



(a) **Ind-10s approach:** F1-score for each epoch of the 10sec dataset by using 110 features



(b) **Ind-5s approach:** F1-score for each epoch of the 5sec dataset (Part 1) by using 110 features



(c) **Ind-5s approach:** F1-score for each epoch of the 5sec dataset (Part 2) by using 110 features

Figure 5: Visualization of the results of Approaches Ind-10s and Ind-5s using Catch22 feature engineering and considering all features extracted for each brain region.

Figure 5b and Figure 5c indicate the results of the Ind-5s approach for each epoch after five runs of experiments for Part 1 and Part 2 respectively. In Figure 5b it is noticed that Epochs 1, 3, 4 and 5 have a large range of distribution in relation to Epoch 2 where its values span across a short range. The same happens in Figure 5c for epochs 1, 2 and 5 where the distribution of the results is large, while Epoch 3 has less dispersed results of F1-score and the median is almost 97%. On the other hand, Epoch 4 is almost 100% which means that the model may overfit.

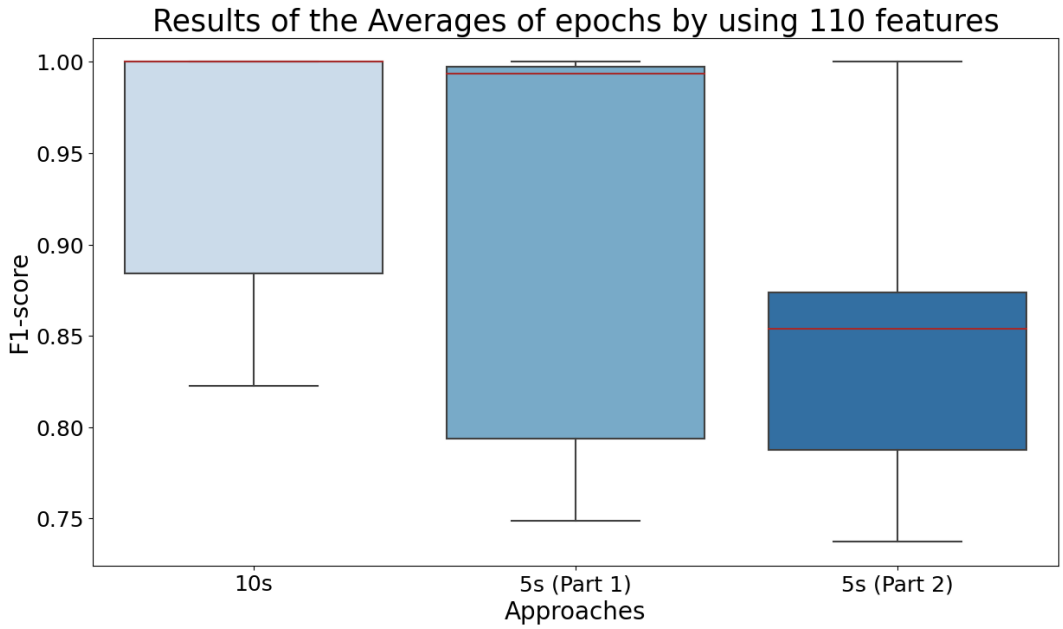


Figure 6: Visualization of the results of **Avg-10s approach** and **Avg-5s approach** using Catch22 feature engineering and taking into account the 110 features extracted from each brain region.

Approaches Avg-10s and Avg-5s are illustrated in Figure 6 where the average of features for all epochs was used to execute the experiments. Similarly, five runs of experiments were performed and Figure 6 indicates the distribution of results. It is observed that the Avg-10s approach has better results than the Avg-5s approach because the Avg-10s approach has a large distribution but the median was found at 100%. However, the Avg-5s approach has also larger distribution for Part 1 of 5 seconds than the Avg-10s approach but the median is lower than the dataset of 10 seconds. Comparing Part 2 of Avg-5s and the other two approaches, Part 2 of the Avg-5s approach has reduced dispersion compared to the other two. However, the median of the Avg-5s approach (Part 2) is slightly more than 85%, while in the other two approaches, the medians are over 98%.

4.2.2 Catch22-Compact method

The original 110 features extracted by Catch22-Comprehensive were used to select 22 features. A filtering method was applied to rank the most important features. Then, the model was trained using only the 22 selected features for each of the four approaches listed in Table 2. The purpose of this experiment is to test how well the model can classify the data using a fixed number of features, which is the main idea of the Catch22 technique and to evaluate the performance of Catch22. Table 14 shows the 22 features that were selected most frequently across all approaches.

No	Selected Feature names
1	CO_trev_1_num_F-region
2	SP_Summaries_welch_rect_area_5_1_P-region
3	CO_trev_1_num_C-region
4	DN_OutlierInclude_p_001_mdrmd_T-region
5	SP_Summaries_welch_rect_area_5_1_O-region
6	SP_Summaries_welch_rect_centroid_C-region
7	DN_HistogramMode_5_P-region
8	DN_OutlierInclude_n_001_mdrmd_F-region
9	DN_OutlierInclude_n_001_mdrmd_T-region
10	DN_OutlierInclude_p_001_mdrmd_F-region
11	DN_OutlierInclude_p_001_mdrmd_O-region
12	CO_f1ecac_C-region
13	CO_FirstMin_ac_C-region
14	CO_HistogramAMI_even_2_5_O-region
15	DN_HistogramMode_10_O-region
16	DN_OutlierInclude_n_001_mdrmd_C-region
17	DN_OutlierInclude_n_001_mdrmd_O-region
18	DN_OutlierInclude_p_001_mdrmd_P-region
19	SB_BinaryStats_mean_longstretch1_P-region
20	SB_TransitionMatrix_3ac_sumdiagcov_T-region
21	SC_FluctAnal_2_dfa_50_1_2_logi_prop_r1_P-region
22	SC_FluctAnal_2_rsrangefit_50_1_logi_prop_r1_F-region

Table 14: Overview of 22 extracted features of Catch22 by using filtering method

The above-mentioned features adhere to the following naming format: {name of the feature}_{region of the brain} where "name of the feature" represents a different Catch22 feature [4] and "region of the brain" refers to a specific region of the brain as shown in Table 3 with each letter representing the initial letter of each region. For example, feature "CO_trev_1_num_F-region", refers to Catch22 feature CO_trev_1_num, describing Time-reversibility statistic $\langle (x_{t+1} - x_t)^3 \rangle_t$ [4] and F-region is the Frontal side of brain (Table 3).

The performance of the two approaches, Ind-10s and Ind-5s, is compared using accuracy and F1-score metrics in Table 15. The table shows the average of the mean and the standard devi-

ation of the outcomes from five repeated experiments. The accuracy metric is slightly higher than the F1-score metric for both approaches. For the Ind-10s approach, the accuracy values range from above 82% to 97.7% with a standard deviation between 3.4% and 7.3% across the individual epochs. For the Ind-5s approach which used two subsets of 5 seconds, the accuracy values of Part 1 vary from above 88% to above 98%, and for Part 2 from 87.9% to 99.3% across the individual epochs. Based on the results of two subsets (i.e., Part 1 and Part 2), it is observed that Part 2 of the Ind-5s approach has better accuracy values in epochs 3 and 4 with a low standard deviation, 2.1% and 1.4% respectively. On the contrary, Part 1 of the Ind-5s approach has the best results in Epoch 4 with an accuracy of 98.9% and a standard deviation of 2.5%. Yet, the F1-score is a more reliable metric than the accuracy for evaluating the performance of the approaches since the class labels are not balanced.

Table 15 shows that the F1-score of the Ind-10s approach fluctuates from 80.3% to more than 97% among the different individual epochs, while the F1-score of the Ind-5s approach (Part 1 and Part 2) ranges from 86.9% to 98.8% for Part 1 and from 87% to 99.3% for Part 2 among the epochs. Comparing Part 1 and Part 2 of 5 seconds subsets, it is noticed that the automated machine learning pipeline achieves higher results of F1-score for epochs 3, 4 and 5 in Part 2 than in Part 1. However, it is also important to note that the F1-score of the Ind-10s approach exceeds 94% in 4 out of 5 epochs, whereas the F1-score of Part 2 of the Ind-5s approach surpasses 97% in only 3 out of 5 epochs.

		Epochs				
Approach	**metric**	**1**	**2**	**3**	**4**	**5**
Ind-10s	Accuracy	97.7 ± 3.4	95.1 ± 6.2	95 ± 3.7	82.7 ± 7.3	97.6 ± 3.6
	F1-score	97.5 ± 3.6	94.8 ± 6.7	94.6 ± 4	80.3 ± 8.9	97.5 ± 3.9
Ind-5s (Part 1)	Accuracy	89.4 ± 6.7	97 ± 2.5	93.5 ± 5.6	98.9 ± 2.5	88.5 ± 7.6
	F1-score	88.2 ± 8	96.9 ± 2.7	93 ± 6.2	98.8 ± 2.6	86.9 ± 9.1
Ind-5s (Part 2)	Accuracy	88.3 ± 4.6	87.9 ± 2.2	98.6 ± 2.1	99.3 ± 1.4	97.4 ± 2.4
	F1-score	87 ± 5.5	87 ± 2.4	98.5 ± 2.2	99.3 ± 1.5	97.2 ± 2.6

Table 15: Performance scores of **Ind-10s approach** and **Ind-5s approach**, for each epoch separately by using 22 selected features. The score of metrics is computed on the test set and averaged in a 10-fold cross-validation. The mean and standard deviation are the averages of the 5 executions of the experiments.

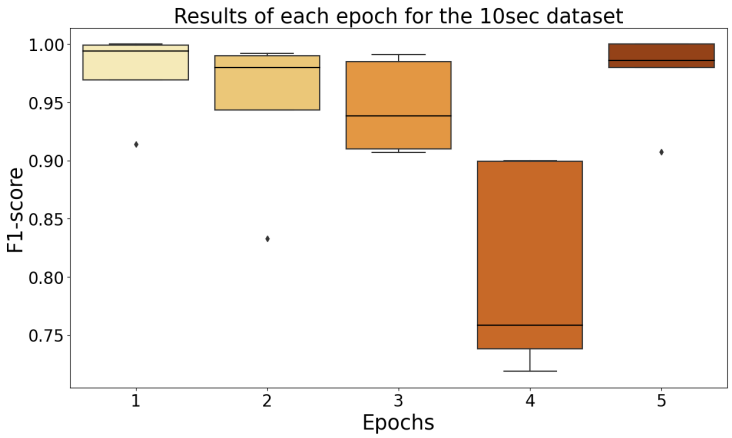
Approach	metric	Averaged epochs
Avg-10s	Accuracy	94.1 ± 6.3
	F1-score	93.4 ± 7
Avg-5s (Part 1)	Accuracy	86.9 ± 6.7
	F1-score	85.5 ± 7.8
Avg-5s (Part 2)	Accuracy	96.7 ± 3.6
	F1-score	96.5 ± 3.9

Table 16: Performance scores of **Avg-10s approach** and **Avg-5s approach**, the average of epochs by using 22 selected features. The score of metrics is computed on the test set and averaged in a 10-fold cross-validation. The mean and standard deviation are the averages of the 5 executions of the experiments.

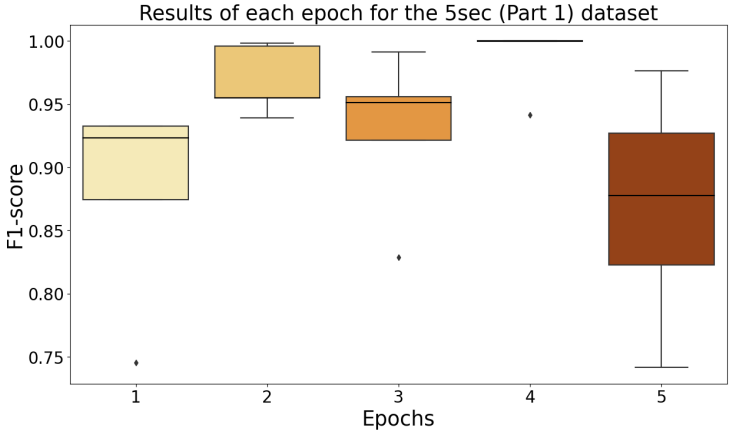
Additionally, Table 16 shows the result of the accuracy and F1-score of Approaches Avg-10s and Avg-5s. It is observed that the Avg-10s approach has a slightly higher accuracy (94.1%) than F1-score (93.4%). The same trend is observed for the Avg-5s approach with both subsets of 5 seconds. However, Part 2 of the Avg-5s approach outperforms Part 1, achieving over 96% in both accuracy and F1-score, while Part 1 only reaches close to 85% in both metrics.

The graph representations of F1-scores for each epoch for the Ind-10s approach and the Ind-5s approach are illustrated in Figure 7, which shows the performance of the Catch22 technique by using 22 selected features. Figure 7a shows the results of the Ind-10s approach among the individual epochs for five executions of experiments. It is observed that the Ind-10s approach performed better in epochs 1 and 5 which have a better distribution of F1-scores during the five executions of experiments. Epoch 5 has a small range of distribution but a lower median than Epoch 1, which has a slightly large range of distribution and a high median. This suggests that the 22 features were more consistent but less effective in Epoch 5 than in Epoch 1 in the Ind-10s approach.

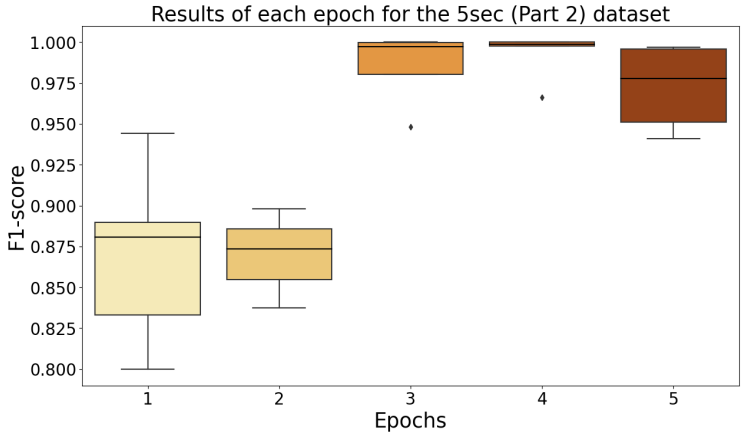
The F1-scores of the Ind-5s approach, namely the two subsets of 5 seconds, are shown in Figure 7b and Figure 7c. Epoch 4 stands out with an F1-score of nearly 100% in both parts, indicating a good fit of the features to the model. This result was achieved by using the cross-validation method to prevent overfitting. However, Epoch 2 has a short distribution and the median is almost 0.95 in Figure 7b while the other epochs have much lower medians. Figure 7c depicts a good distribution except in Epoch 4, in Epoch 3, where the results of the experiments are less scattered than epochs 1, 2 and 5, and its median is close to 100%.



(a) **Ind-10s approach:** F1-score for each epoch of the 10sec dataset by using 22 features



(b) **Ind-5s approach:** F1-score for each epoch of the 5sec dataset (Part 1) by using 22 features



(c) **Ind-5s approach:** F1-score for each epoch of the 5sec dataset (Part 2) by using 22 features

Figure 7: Visualization of the results of **Ind-10s approach** and **Ind-5s approach** using Catch22 feature engineering and considering the most important 22 features extracted by using the filtering method.

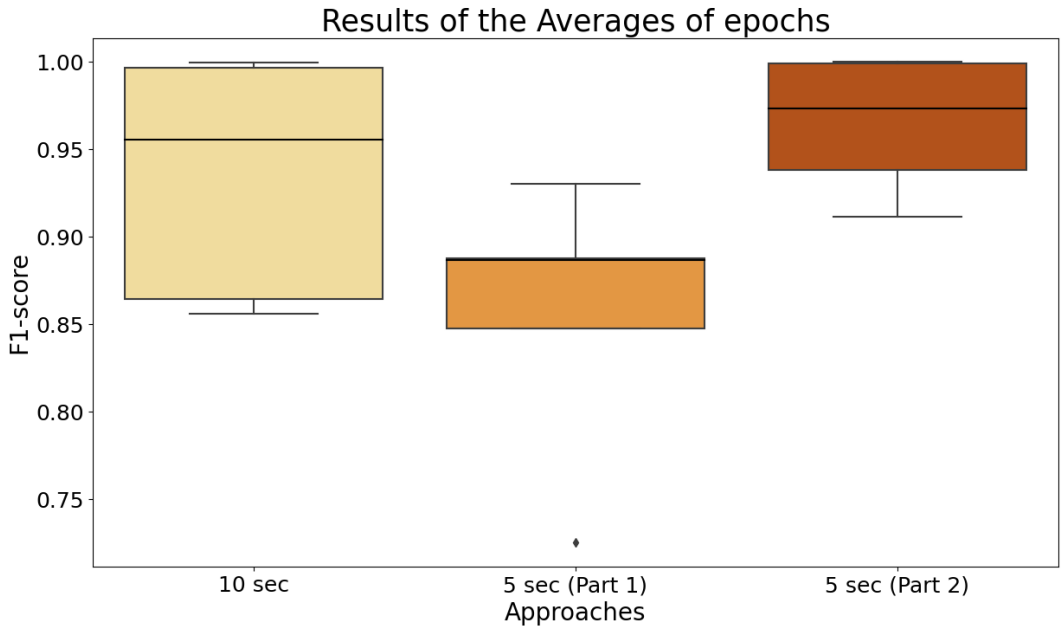


Figure 8: Visualization of the results of **Avg-10s approach** and **Avg-5s approach** using Catch22 feature engineering and taking into account the most important 22 features extracted by using the filtering method.

Figure 8 illustrates the F1-score of the experiments, after five runs, of Approaches Avg-10s and Avg-5s. It is noticed that the Avg-5s approach, and specifically Part 2 of 5 seconds, achieves the highest results in relation to the results of the Avg-10s approach and Part 1 of the Avg-5s approach. Furthermore, the box plot of Part 2 of the Avg-5s approach indicates a larger spread of values than Part 1 of the Avg-5s approach, and a median that is close to the middle of the box, implying that the middle 50% of F1-scores are equally distributed in the interquartile range. On the other hand, the results of the Avg-10s approach are more dispersed than Part 1 and Part 2 of the Avg-5s approach, and the median is close to 95%. Whereas, the Avg-5s approach of Part 1 exhibited a slight distribution of values between almost 0.89 and 0.85, with a higher density of values near 0.89. In addition, Part 2 of the Avg-5s approach has a good distribution and the median is 96.5% which means the average of features of all epochs performs better than the Avg-10s approach and Part 1 of the Avg-5s approach.

5 Conclusion

The main objectives of this study were to identify a reliable dataset that could assist clinicians in diagnosing cognitive function and to compare the performance of two feature engineering techniques. To achieve these objectives, a dataset of 10 seconds and two subsets of 5 seconds were utilised, each containing five epochs for each patient and 21 time series for each epoch. An automated Machine Learning pipeline was applied to execute the experiments using two feature engineering techniques: ts-fresh combined with Boruta, and Catch22. The experiments revealed that the features selected by ts-fresh combined with boruta varied for each experiment, as well as the number of selected features. On the other hand, Catch22 extracted 22 features for each time series and two methods were created, namely Catch22-Comprehensive and Catch22-Compact.

Based on the results of the data, the 10 seconds dataset showed better performance in averaged epochs with an F1-score of 91.99%, while the other two subsets showed better results among the individual epochs by using the ts-fresh combined with boruta. It was noticed that the average of epochs of 10 seconds dataset was robust and consistent as it had observed in [12]. On the other hand, using the 10 seconds dataset and the two subsets of 5 seconds (Part 1 and Part 2), Catch22 was applied to extract and select 22 features from each time series. This led to two different methods of conducting the experiments, namely Catch22-Comprehensive and Catch22-Compact. The first one used all the features from each brain region, resulting in 110 features in total. The second one used only the 22 most important features by using a filtering method. The Catch22-Comprehensive method produced a model that was overfitted due to the high number of features and a relatively small dataset. The Catch22-Compact method performed better for the Ind-5s approach and specifically Part 2 of 5 seconds subset, achieving the highest F1-score.

In conclusion, the Ind-10s approach and Ind-5s approach had better results by using the Catch22 technique with 22 selected features than by using ts-fresh combined with Boruta by the mean of the F1-score from the five runs of experiments as presented in Table 15. Based on the approaches that used the different lengths of data, i.e., the Ind-10s approach used 10 seconds and the Ind-5s approach used 5 seconds, the five epochs of 5 seconds can be used instead of the five epochs of 10 seconds because both approaches yielded similar results in Catch22-Compact method. However, the Ind-5s approach consisted of two subsets of 5 seconds, which were obtained by dividing the dataset of 10 seconds into two parts, which was a random procedure. If the 5 seconds subset had been selected by the clinicians, using their expertise, then the results might be more reliable.

Furthermore, comparing the two feature engineering techniques, Catch22 achieved higher F1-scores in all approaches and the standard deviation was very low in the average of five execution of experiments. However, the cross-validation method has been used in Catch22 but there is a possibility of overfitting due to the relatively small data. In contrast to Catch22, which always selects the same 22 features for each time series analysis, ts-fresh combined with Boruta chooses a different number of selected features during the experiments. This leads to different performance outcomes for the two techniques. For example, ts-fresh combined with Boruta tends to achieve higher F1-scores when it selects a large number of features, while Catch22 has more consistent and robust results across different approaches. The reason for Catch22's

best performance is that it includes a diverse set of features that capture various aspects of time series dynamics, such as autocorrelation, distribution, outliers and scaling properties [4]. However, further research is required to avoid overfitting in the model.

6 Future work

This project has provided valuable insights into the use of the dataset of 10 seconds and the two subsets of 5 seconds for the diagnosis of cognition of patients with Parkinson's Disease and comparing two feature engineering techniques. However, there are several topics for further exploration and improvement:

Towards the same direction, alternative feature engineering techniques, such as tsflex [16] and tsfe [2], could be assessed based on their performance within an automated machine learning pipeline, providing insights into their effectiveness in capturing relevant patterns and improving predictive models.

One potential direction is to investigate whether epoch selection can be entirely avoided, exploring alternative approaches to handle the data more effectively.

In addition, it would be worthwhile for future work to investigate the performance of the model using datasets comprising varying sizes of epochs directly sourced from clinicians. This will allow us to assess the impact of different epoch lengths on diagnostic accuracy and provide a more comprehensive understanding of the system's robustness.

An intriguing approach for further research could involve evaluating the performance of various machine learning models, including Support Vector Machines or deep learning architectures.

Exploring these directions can enhance our understanding of the optimal handling of time series data in Parkinson's Disease diagnosis and develop the feature engineering techniques within the context of Automated Machine Learning. Such investigations have the potential to improve the accuracy and efficiency of automated systems.

References

[1] Sarah Guido Andreas C. Müller. *Introduction to Machine Learning with Python*. O'Reilly Media, Inc., September 2016.

[2] Marília Barandas, Duarte Folgado, Letícia Fernandes, Sara Santos, Mariana Abreu, Patrícia Bota, Hui Liu, Tanja Schultz, and Hugo Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[3] Jeffrey W Britton, Lauren C Frey, Jennifer L Hopp, Pearce Korb, Mohamad Z Koubeissi, William E Lievens, Elia M Pestana-Knight, and EK St Louis. Electroencephalography (eeg): An introductory text and atlas of normal and abnormal findings in adults, children, and infants. 2016.

[4] Philip Knaute Simon R. Schultz Ben D. Fulcher Nick S. Jones Carl H. Lubba, Sarab S. Sethi. catch22: Canonical time-series characteristics. data min knowl disc. page 1821–1852, 08 2019.

[5] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307:72–77, 2018.

[6] Victor J. Geraedts, Lennard I. Boon, Johan Marinus, Alida A. Gouw, Jacobus J. van Hilten, Cornelis J. Stam, Martijn R. Tannemaat, and Maria Fiorella Contarino. Clinical correlates of quantitative eeg in parkinson disease. *Neurology*, 91(19):871–883, 2018.

[7] Victor J Geraedts, Lennard I Boon, Johan Marinus, Alida A Gouw, Jacobus J van Hilten, Cornelis J Stam, Martijn R Tannemaat, and Maria Fiorella Contarino. Clinical correlates of quantitative eeg in parkinson disease: A systematic review. *Neurology*, 91(19):871–883, 2018.

[8] V.J. Geraedts, M. Koch, M.F. Contarino, H.A.M. Middelkoop, H. Wang, J.J. van Hilten, T.H.W. Bäck, and M.R. Tannemaat. Machine learning for automated eeg-based biomarkers of cognitive impairment during deep brain stimulation screening in patients with parkinson's disease. *Clinical Neurophysiology*, 132(5):1041–1048, 2021.

[9] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc., September 2019.

[10] Marios Kefalas, Milan Koch, Victor Geraedts, Hao Wang, Martijn Tannemaat, and Thomas Bäck. Automated machine learning for the classification of normal and abnormal electromyography data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1176–1185, 2020.

[11] Milan Koch and Thomas Bäck. Machine learning for predicting the impact point of a low speed vehicle crash. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1432–1437, 2018.

[12] Milan Koch, Victor Geraedts, Hao Wang, Martijn Tannemaat, and Thomas Bäck. Automated machine learning for eeg-based classification of parkinson's disease patients. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4845–4852, 2019.

[13] Milan Koch, Hao Wang, and Thomas Bäck. Machine learning for predicting the damaged parts of a low speed vehicle crash. In *2018 Thirteenth International Conference on Digital Information Management (ICDIM)*, pages 179–184, 2018.

[14] Miron B. Kursa and Witold R. Rudnicki. Feature selection with the boruta package. *Journal of Statistical Software*, 36(11):1–13, 2010.

[15] Gonzalo Rojas, Carolina Alvarez, Carlos Montoya Moya, Maria de la Iglesia Vaya, Jaime Cisternas, and Marcelo Gálvez. Study of resting-state functional connectivity networks using eeg electrodes position as seed. *Frontiers in Neuroscience*, 12, 03 2018.

[16] Jonas Van Der Donckt, Jeroen Van Der Donckt, Emiel Deprost, and Sofie Van Hoecke. tsflex: Flexible time series processing feature extraction. *SoftwareX*, 17:100971, 2022.

[17] Hao Wang, Michael Emmerich, and Thomas Bäck. Cooling strategies for the moment-generating function in bayesian global optimization. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2018.

[18] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Back. A new acquisition function for bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512, 2017.

[19] Jonathan Waring, Charlotta Lindvall, and Renato Umeton. Automated machine learning: Review of the state-of-the-art and opportunities for healthcare. *Artificial intelligence in medicine*, 104:101822, 2020.

[20] Alice Zheng and Amanda Casari. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.", 2018.