|  |  |
| --- | --- |
| Thesis  DATA8006: HDip Data Science Analytics Project | Description  Investigation into Using Motor Activity as well as Computerised Tests to Build Binary Classification Models to Predict ADHD.  JOHN VERLING  Data Science and Analytics |

# Abstract

An investigation was conducted into various approaches to binary classification between people with, and without, ADHD using motor activity gathered from wearable devices and results of computerised tests. Initially, the project involved reproducing an existing study [1] using motor activity. On completion of this, other approaches to modelling using motor activity were explored. Finally, results from the computerised test provided the best performing model with an accuracy of 76%. An analysis of this model revealed insights that could lead to future work.

# Introduction

## Attention Deficit/Hyperactivity Disorder (ADHD)

ADHD in adults is a neurological condition that presents through symptoms of inattentiveness, hyperactivity, and impulsivity. It is considered to have three different presentations.

* Inattentiveness (sometimes referred to as ADD)
* Hyperactivity and Impulsivity
* Combined (both together)

To be diagnosed, symptoms must be judged to be long-term, and they must be causing significant impairment in a person’s social, academic, or occupational functioning [2]. According to a global systematic review [3], an estimated 2.58% of adults have ADHD symptoms from childhood into adulthood and 6.76% of adults show symptoms.

## Diagnosis

ADHD is currently diagnosed through subjective evaluation by a qualified professional. Given that the diagnosis is based on impairment, it seems unlikely that a Machine Learning or a Statistical model will replace this. However, the ability to detect ADHD through widely available and relatively cheap technology such as wearable devices and computerised tests could be valuable as a screening tool or as a pathway to deepen our understanding of the condition.

## Data

The data used for this study is taken from the Hyperaktiv [1] study and the Psykose [4] datasets. The studies and analyses of these datasets are summarised in the literature review. The Hyperaktiv dataset, which is publicly available at <https://osf.io/3agwr/> , contains anonymised data from 103 participants. It includes motor activity, heart rate over time, categorical data related for diagnoses of various conditions, including ADHD, along with results of a Conners Continuous Performance Test (CPT-II). In addition to the Hyperaktiv dataset, motor activity from the 32 healthy controls were taken from the Psykose study. For the healthy controls, patient information about age and gender are provided. A detailed breakdown of the dataset is included in the Dataset section below.

### Motor Activity

The motor activity was measured for all study participants using wrist-worn actigraph devices. The details of the measuring device are given in the dataset subsection.

## Aims of the Project

The aims of this project were:

* Investigate and reproduce the binary classification model produced in the Hyperaktiv study [1].
* Take and apply lessons from this investigation to produce a better model.
* Use computerised test data (CPT-II) to form a binary classification model and gain insights into future feature engineering.

## Structure of Project

The project had 3 phases. Each one is outlined below.

### Phase 1: Reproducing the Original Study

In phase 1, the goal was to reproduce the binary classification model provided by the Hyperaktiv publication [1] and, ideally, improve it through feature engineering. An analysis of the features used for modelling in the original study was conducted. Insights gained from this analysis were used to improve data preprocessing and modelling in phase 2.

### Phase 2: Motor Activity from the Hyperaktiv dataset

In phase 2, a more sophisticated approach to preprocessing the motor activity data was used. Solutions were found to overcome issues related to overfitting and data-leakage. It was decided in phase 2 that only the Hyperaktiv dataset, i.e. excluding the healthy controls, would be used in modelling.

### Phase 3: Binary Classification using CPT-II data

In phase 3, a ML model was produced using the results of the Conners Continuous Progression Test (CPT-II). The results were then analysed using other patient data.

# Literature Review

## Hyperaktiv: Original Study

A dataset, Hyperaktiv [1], was published in 2021 through the Open Science Framework (OSF) website. The study recruited 103 participants who were referred for diagnoses to a psychiatric outpatient clinic for ADHD, mood, or anxiety disorders. Participants wore actigraph devices to measure motor activity. The diagnoses subsequently made, along with other patient information was gathered, including data from a Conners Continuous Performance Test (CPT-II). Of the 103 participants, 85 wore actigraphs.

Of these 85 only 84 were selected. It is not clear why a participant was dropped but it may be that one of the ADHD participants was on stimulants. This assumption is based on a related study [5]. In the study, the patient receiving stimulants as medication was dropped.

For each participant 788 features were extracted using the tsfresh [6] open-source package in Python. Some examples of the features produced are:

* Basic statistics
  + Mean activity, median activity, skewness, variance etc.
* Coefficients and summary statistics from Fast Fourier Transforms, this accounts for more than 400 features.
* Statistics relating to Auto Regression with various lags.

A complete list of the features is available in the tsfresh documentation [6].

Using these features, the study examined several ML models using Hyperaktiv dataset along with activity data from healthy controls from the Psykose study [4]. A 10-Fold Cross-validation (CV) procedure was used. Logistic Regression and Random Forest, depending on metrics prioritised, showed the most promising performance. The Logistic Regression model produced a recall of 82% with a precision of 58% while the Random Forest produced a recall of 60% and a precision of 76%.

## Related Studies

A study [7] that used only the Hyperaktiv dataset without the healthy controls provided a model with an accuracy of 98.43% along with many other excellent performance scores. They also used tsfresh to extract the 788 features. They used a Principal Component Analysis to reduce the dimensionality of the dataset. Several ML models were tested. Support Vector Machine (SVM) performed the best. The SVM used a regularisation hyperparameter to avoid overfitting. It provided a recall of 98.33% and a precision of 98.56%. A 10-fold cross validation was performed to decrease the chances of overly optimistic results. A Random Forest classifier produced a similar performance.

The Psykose dataset [4], which looked at motor activity data in participants with schizophrenia was published along with an analysis in 2020. Activity data was gathered from 22 psychotic hospitalised patients in a long-term ward. Data was also gathered from 32 healthy controls which were reused in the Hyperaktiv study being reproduced for this project. Various other data was presented in the study for each participant relating to age, gender, type and diagnostics of schizophrenia, and medication. More details on the control group are given below in the Data section. The features used were mean activity, standard deviation, and proportion of zero values. These were calculated for each patient for each day. The features were then used to produce baseline ML performance metrics. 10-Fold Cross Validation was used as well as Leave-one-out (LOO). For 10-Fold CV, Logistic regression produced an average precision of 0.88 and an area-under-the-curve (AUC) of 0.92 with only 10% of the data used for training. In the LOO analysis, Random Forest had a weighted average of 0.84 for recall, precision, and F-score.

Another study [8] looked at motor activity, measures of rotational movement, and other data, for children with and without ADHD. Children with other conditions were not included and all participants with ADHD had the hyperactive presentation. Approx. 2/3 of the features that were selected for modelling were from motor activity. Gender and age were selected for some models. High-Resolution Histograms (HRH) provided approx. half of the selected features. Basic statistics and frequency domain characteristics were selected as features regularly. The data that led to the best model was gathered while the participants were performing a computerised test (CPT-II). The CPT-II is discussed in the dataset subsection. A linear Support Vector Machine (SVM) classification model showed excellent performance metrics.

Another study [9] used 24-hour actigraph readings with non-linear signal processing analysis to achieve an extremely promising classification of 6-year-olds with the combined form of ADHD and healthy controls. Their analysis involved signal decimation (reducing the sample rate) by various amounts to observe different behaviours. Measures of central tendency for signals, as well as symbolic dynamics, which involves calculating occurrence probabilities of certain activity states (low, medium, and high) for the signals. The motor activity data used was in three-dimensions. All three dimensions were analysed separately. Principal Component analysis was a central part of the feature selection process. Also, listed in this study, were features used in previous analyses. Among them are High-Resolution-Histograms, and basic statistics.

A study [5] in 2015 analysed a very similar dataset to the Hyperaktiv one. It is not clear from reading the paper, but it seems likely to be a subset of the Hyperaktiv dataset. They used statistical analysis rather than machine learning. The participants were made up of people who were referred for diagnosis for either ADHD, mood disorders, or anxiety disorders. They carefully selected the first 300 minutes of waking time of all participants and created features using Fourier analysis. They also used data gathered over a six-day period with activity rates aggregated over half hour periods. The Fourier analysis was able to detect a difference between ADHD and non-ADHD cohorts.

# Dataset

## Motor Activity

The Hyperaktiv dataset [1] contains motor activity data for 85 adults. The motor activity was measured using a wrist worn actigraph (Actiwatch, Cambridge Neurotechnology Ltd, England, model AW4) [10]. The device creates voltages that are proportional to the motor activity. They were sensitive to accelerations of 0.05g m/s2, orabove, and made 32 recordings per second. The values were measured in 3 dimensions. The actigraph measures the integration of the intensity, amount, and duration of movement in all 3 dimensions [10]. The voltages were converted into integers that were proportional to the intensity of the movement. The sum of the integers for each minute over the time that the device was worn is what is seen in the raw csv files.

Of the 85 adults, 45 were diagnosed with ADHD and 40 were not. Data was gathered over a mean of 6.6 days (standard deviation of 1.3 days) for the group with ADHD and, for the 40 adults without ADHD, 7.2 days (standard deviation of 0.9 days). As mentioned in the Literature review, all the 85 participants were referred for diagnoses to the clinic for either ADHD, Anxiety Disorder, or Mood Disorder.

Activity data from 32 healthy controls, originally gathered as part of the Depresjon dataset [11], was used. They were used again in the Psykose study [4]. The average number of days used from the healthy controls was 12.6 days with a standard deviation of 2.3 days. According to the Psykose paper [4], the healthy controls consist of 23 hospital staff members, 5 students nurses and 4 healthy participants gathered by a general practitioner. However, in the Depresjon paper [11], the 4 healthy participants are described as former patients without current psychiatric symptoms. Concerns about the nature of the control group are discussed in the discussion section.

## Additional Data

Detailed additional data is available for the 85 adults in the Hyperaktiv dataset. This will be discussed in the Patient Information section below. It should be noted that the patient information for the 32 healthy controls is limited to age category, gender and number of days wearing the device. None of the controls have a history of psychotic or affective disorder, which allows for certain assumptions around diagnoses and medication [11]. As there is much more additional data available for participants in the Hyperaktiv dataset than the Psykose dataset, using the healthy controls from the Psykose dataset means not being able to use a lot of data in the Hyperaktiv dataset. For example, the CPT-II data, discussed below, is provided for the Hyperaktiv dataset. This data is not available in the healthy controls. The use of the additional data was a factor in deciding to exclude the healthy controls from phase 2 of the project.

## Conner’s Continuous Performance Test II (CPT-II)

A total of 99 participants completed the CPT-II test. This is a computerised assessment that measures impulsivity and sustained attention. The test consists of 360 trials and the results for each one are given. Several summary statistics and index values are provided for each patient. These summary statistics formed the features for phase 3 of the project.

## Patient Information

For all 103 participants in the Hyperaktiv dataset, categorical data is provided and described below.

### Age

There are four age categories denoted by integers 1 to 4. The categories are given below:

* 1: 17-29 years
* 2: 30-39 years
* 3: 40-49 years
* 4: 50-67 years

### Gender

A value of 0 means female and a value of 1 means male.

### Diagnosed Conditions

For all participants, diagnosis are provided for ADHD, ADD (a term often used for the inattentive presentation of ADHD), bipolar disorder, unipolar depression, anxiety disorder, substance abuse along with an “other” column for other psychiatric disorders.

### Diagnostic Metrics

The results of several diagnostic tests are provided also. A detailed description of their medical meaning is provided in the dataset section of the Hyperaktiv study [1]. None of them were used in modelling or analysis, therefore individual descriptions are not included here.

### Medication

The dataset included whether, or not, participants are using various medications. Listed in the file are antidepressants, mood stabilisers, anti-psychotics, anxiety, sleep, opioids and stimulants (commonly prescribed to people with ADHD). There is also a summary column which simply states whether a patient is medicated or not. Data is denoted as follows:

* 1: patient has been prescribed medication
* 0: patient has not been prescribed medication.
* 9: not known

### Heart Rate Data

Heart rate data was not used for this study therefore it is not discussed further.

# Methodology

## Exploratory Data Analysis

### Data Quality

In order to check the quality of the data, all the time series data was sliced into days where incomplete days were discarded. This resulted in 585 motor activity timeseries from the Hyperaktiv dataset. Each one is 24 hours (1440 minutes) long. These timeseres were plotted with motor activity on the y-axis and time on the x-axis. Several plots showed data where it is clear that the device was removed for extended periods of time. These rows of data were removed. Below are examples of a normal day plot, a day plot which was rejected and one that was kept despite missing some data.

#### Fig 1: Normal day plot for a patient with ADHD

A graph of a patient activity

Description automatically generated

#### Fig 2: A plot that was rejected as the participant clearly took off the device for an extended time

A graph of a patient activity

Description automatically generated

#### Fig 3: Plot that was kept despite some missing data

A graph of blue lines

Description automatically generated

In phase 2 of the project, 449 out of the 585 days were kept, meaning 23% of the data was removed. The same data cleaning methodology was applied to the healthy controls from the Psykose dataset. However, they were not used in the final methodology so are not discussed further.

### Phase 1: Reproducing the Original Study

For the purpose of reproducing the baseline presented in the Hyperaktiv study [1], mean plots, broken down by relevant categorical data were produced.

#### Cohorts

The three cohorts consist of participants that have ADHD, participants with other diagonoses (non-ADHD) and healthy controls from the Psykose study.

##### Fig 4: Mean Average Daily Activity by cohort

A graph of a graph

Description automatically generated with medium confidence

Fig 4 suggests that the healthy control cohort appear to move more on average especially in the late morning and afternoon. The non-ADHD cohort appear to move less than the non-ADHD cohort in the morning. Given that the healthy controls appear to move more, and the non-ADHD cohort move less, one might expect that combining the healthy controls and non-ADHD would make the differences over the days smaller.

##### Fig 5: Average Daily Activity by ADHD

A graph of a graph showing adhd and healthy

Description automatically generated

As expected, fig 5, shows that the differences, on average, are more subtle when categorised as having ADHD or not. There appears to be a small difference in the early morning (5.40 – 7.10) and late morning/afternoon (10.00 – 16.40).

##### Fig 6: Average Daily Activity by ADHD for Males

A graph of a graph showing adhd

Description automatically generated

There is no clear distinction between ADHD and non-ADHD in Males except for the peak at 1100 – 1200 minutes (18.30h – 20.00h).

##### Fig 7: Average Daily Activity by ADHD for Females

A graph of adhd and adhd

Description automatically generated

Females with ADHD may have lower activity during late morning to afternoon. Interestingly, the spike between 18.30h – 20.00h is not seen in the females.

## Phase 2: Motor Activity from the Hyperaktiv dataset

The following are plots using the Hyperaktiv dataset only i.e. The healthy controls are not part of it.

##### Fig 8: Average Daily Activity by ADHD

A graph of a graph showing different types of numbers

Description automatically generated with medium confidence

ADHD cohort seem more active in the morning. A brief peak in the ADHD cohort is seen in the evening from approx. 18.30 to 19.30.

##### Fig 9: Average Daily Activity by ADHD for Males

A graph of a number of people with adhd

Description automatically generated

Fig 9 shows a significant gap between cohorts for males between 400 – 650 minutes (6.40 a.m. – 8.50 p.m.).

##### Fig 10: Average Daily Activity by ADHD for Females

A graph of a graph showing different types of numbers

Description automatically generated with medium confidence

Fig 10 and fig 11 show that the spike in the ADHD cohort from approx. 18.30h – 20.00h is not there for female participants or men between 30 and 39 years of age. It is present in all the other age categories for men implying that it is not the result of a small amount of outliers.

#### Fig 11: Average Daily Activity by ADHD ages 30 - 39

A graph of a graph showing different types of numbers

Description automatically generated with medium confidence

The age group between 30 and 39 years, which consists of 26 participants, may be problematic as their mean plots are almost identical.

### Patient Information

As described in the Dataset section, a large amount of information, and potential features, are given for each participant in the Hyperaktiv study. Among those variables were summary statics for the CPT-II computerised test. Many of these looked promising from their boxplots. For brevity, only one is shown. During the EDA, seven more were found and are included in the appendix in the graph subsection. These seven, as well as the feature shown in fig 13 were eventually chosen for modelling.

##### Fig 13: Boxplot of one of the Summary Statistics Provided with The Conner’s Continuous Performance Test by ADHD

A graph with blue and green squares

Description automatically generated

Fig 13 shows one of many promising variables for distinguishing between cohorts from the CPT-II summary statistics.

##### Fig 14: Count Table for Substance abuse broken down by ADHD

A screenshot of a computer

Description automatically generated

Categorical variables were examined using tables. The most notable one, fig 14, shows the counts for substance abuse among ADHD and Non-ADHD participants. 31% of the ADHD cohort have a substance abuse issue compared to 16% of the Non-ADHD.

### Tsfresh Features

For both the original study, and subsequent work on the motor activity data, a python package tsfresh [6] was used to extract 788 timeseries features for each participant. More details on the feature extraction are discussed in the data preprocessing section below.

Boxplots of all continuous variables (737 features), grouped by cohort were inspected. Fig 15 below shows an example. Of all the plots, this was the one judged to distinguish the two cohorts the best. The boxplots only illustrate how likely a feature is to predict well from a univariate perspective. The fact that no large difference was observed does not mean that ML models will not perform well. It is possible that combinations of different features could distinguish the labels in ways that are not obvious from this analysis.

#### Fig 15: Boxplot of Fast Fourier Transform: Angle coefficient 33

A graph with lines and text

Description automatically generated with medium confidence

## Data Preprocessing

### Phase 1: Reproducing the Original Study

In phase 1 of the project, a reproduction of the original study was completed. The features used in the study were provided with the dataset and the code [12]. Hence it was not necessary to extract the features from the motor activity again.

### Phase 2: Motor Activity from the Hyperaktiv dataset

#### Dataset Structure

For the rest of the study, it was decided to preprocess the data by days. For each participant, every full day of motor activity was saved as an individual timeseries. Incomplete days were discarded. This means that several “days” could belong to the same participant which increases the number of rows from 85 to 449 after cleaning. These 449 rows of data, containing motor activity were used to produce the mean plots for the EDA.

As discussed in the literature review, the Psykose [4] study succesfully used a similar strategy where they had more than 600 rows with only 52 participants as each row was a day. Also discussed in the literature review, a statistical analysis [7] found differences in the Fourier Transform coefficients. They compared the first 5 waking hours of each participant’s motor activity. By cutting the data into days, or comparable 5 hour periods, both studies kept the lengths of the timeseries constant and compared “like-with-like”. Several of the features produced by tsfresh are sensitive to these differences. This influenced the decision to adopt the “by day” structure.

#### Feature Extraction

##### Tsfresh

Tsfresh [6] was used to extract 788 timeseries features for each row (day). Some of these are described in the literature review of the original study. This resulted in a features dataset of 449 rows. belonging to 84 participants, with 788 features.

##### Missing values

Among the 788 features produced, 6 of the columns contained missing values.

These columns were examined through the tsfresh documentation [6].

* query\_similarity\_count: For this column all the values were missing. It can be used to count the number of datapoints in a timeseries that are within a certain Euclidean distance range from a given point (query).
* max\_langevin\_fixed\_point: All but one row of the data contained missing values. The reason for this is unclear.
* friedrich\_coefficients: 4 diiferent coefficients ascociated with the Langevin Model. Similar to the above, only one row contained a numerical value and all others were missing. The reason/s for this are unclear. Due to time constraints, these potential features were not investigated and hence discarded.

##### Columns with zero variance

The VarianceThreshold function from scikit learn [13] was used to discard columns that had all the same values i.e. no variance. 48 columns were removed in this process.

This resulted in a dataset of 449 rows and 734 features.

##### Multicollinearity

While not used in the final methodology, a function was written that looked at the correlation matrix of the data-frame and discarded 1 column where they had a correlation coefficient above a certain threshold. Setting this threshold to 0.8, reduces the dimensionality to 497 features. As discussed below in the Feature Selection subsection, The high number of features was a challenge throughout the modelling process.

##### Grouping the Data

While making each day of data into a row increased the number of rows, it introduced a challenge during modelling (see Modelling section). Given that several rows could belong to one participant, it was important that a participant in a testing group did not have any of their rows in the training set as the algorithm may just be remembering the features of that participant. This would result in overly optimistic performance metrics. Because of this and the small number of participants (84), the final methodology used was “Leave-One-Group-Out Cross Validation” (LOGO CV). This is discussed in more detail later in the modelling section. To enable this, a Pandas series was created with a row for each patient containing a group number. This allowed all the days associated with a participant to be in one group. Hence, when testing models, none of the participants had rows in both testing and training datasets.

##### Standardisation

Over the study, the StandardScaler function from Scikit Learn [13] was used to convert all values into units of the standard deviation of their column. The values were also centred so that mean values had were replaced with 0. This prevents the scale of each column from affecting performance. This is important for algorithms that use distance metrics. Also, for Logistic Regression Regularisation, the type of standardisation is important for the units of the penalty to the cost function. Regularisation is discussed in more detail in the Modelling section. Ultimately, due to the choice of Random Forest Classifier (RFC), standardisation was not used in the final methodology. RFCs are not affected by the scale of the units. RFC and Decision Trees Classifiers are explained in detail in the modelling section.

## Modelling

### Algorithms Used

An explanation is given below for the main algorithms used throughout modelling. Only the models important for understanding the results are explained in detail. All the modelling was in done in Python using Scikit Learn [13].

#### Binary Classification ML Models

In general, classification in ML involves taking a dataset in the form of a table where each row represents a sample, in this case a participant, and each column represents a piece of information about that sample. For each sample, the variable to be predicted is referred to as the target class or the label.

The rows are divided into a training dataset and a held-out test dataset. The various algorithms have different methods of mapping the representations to the target classes. They then predict the labels of the held-out test dataset based on their features. These predicted labels are then compared to the actual labels to assess the performance of the model.

Throughout the project, the variable to be predicted is whether a participant had ADHD or not hence, the models are binary classifiers.

#### Decision Tree Classifiers (DTC)

A DTC uses the features in the training data to find decision criteria for classifying the samples. The basic functioning is best described using a plot.

##### Fig 16 Plot of a Decision Tree with a depth of 2

A diagram of a number of samples

Description automatically generated with medium confidence

To understand fig 16, we must understand the terms. X[2] refers to the third feature/column, in this case the “Raw Score VarSE” from the CPT-II dataset. Based on the training data, the DTC has learned that dividing up the samples with X[2] less than or equal to 6.386, or above is a good criteria for predicting if someone has ADHD. This can be thought of as a “split”. Samples with X[2] less than this number are put into the “box” on the left and above this number are put in the “box” on the right. When a “box” is split, it is called a decision node and a “box” that isn’t split is called a leaf node. The very first node is called the “root node”.

The “gini” number in the nodes in fig 16 , the gini index, is a measure that the algorithm uses to decide what the best splitting critical point, in this case 6.386 is, as well as what feature is best to split. There are other metrics for doing this as well. For brevity only one is explained. They are all ways to try to assess how well a split creates homogenous nodes (pure). An ideal split would create 2 leaf nodes where all the samples in the leaves are of one label, e.g. ADHD/non-ADHD. These leaf nodes would be said to be “pure”. When the DTC considers a feature for forming a node, it assesses how much that split would reduce the weighted average impurity of the resulting leaves. For each feature it considers, it must find the optimal splitting point (if the variables are continuous, as they are in this example).

A gini value is at its maximum at 0.5 (1/2 the labels in the node are of each label) and its minimum at 0 (the node only contains one label). The “value” in the root/decision nodes above show that the node has an even amount of both labels (49), hence the gini index is 0.5. The best feature, and its critical point, is the one that lowers the weighted average of the gini indices the most.

In the first split above, setting the critical value for X[2] to be 6.386 puts 43 samples in one node and 55 in the other. The above process is then repeated for each node until a stopping criteria is reached.

##### Stopping Criteria

The DTC above could keep splitting nodes until all of them are pure. This can result in overfitting to the training data. For example, if the tree becomes too complex, i.e. too many splits, it could have a leaf node for each sample in the training data. In this situation, it is simply memorising the training data and highly unlikely to perform well on the held-out dataset. There are several ways to manage how the tree splits using hyperparameters provided by Scikit Learn [13].

###### Cost Complexity Pruning (CCP)

CCP is a form of regularisation. Regularisation is a way of preventing overfitting to training data by penalising classifiers for becoming too complex. In the case of DTC, becoming more complex means adding more decision nodes and leaves.

When the DTC is deciding to split a node or not, it looks at the reduction in impurity. CCP adds a set value to the resulting impurity decrease. This CCP value is a hyperparameter in Scikit Learn. This means that if a split reduces impurity by a small amount, the CCP hyperparameter could make it so that there is no reduction of impurity, or even an increase. In this way it provides a “penalty” to splits that increase the complexity for small decreases in impurity. This prevents the DTC from overfitting due to “over-splitting”.

Several other hyperparameters can provide stopping criteria but CCP was found to be most effective over the project.

#### Random Forest Classifier (RFC)

When several classifiers are used together to form one ML model, this is known as an ensemble. RFC is an ensemble of DTCs. In other words, it creates several DTCs for a dataset and each DTC then votes for what the predicted label should be. The label voted for by the most DTCs is the predicted label.

The differences, and advantages, of RFC is in how it creates the DTCs.

###### Maximum Features

When RFCs are building DTCs, every time it looks for the next feature to split, it doesn’t look at all the features available, although it can if required. It selects a random subset of features and selects the best one from that subset. The number of features that it “looks at” for each split can be set as a hyperparameter. This random component means that the DTCs created are all different.

Overfitting due to Large Number of Features

For phase 2 of the project, the number of features being used for modelling was 736. As the number of features increases relative to the number of samples, the probability that a feature will predict the label well just by random chance goes up. This is another form of overfitting where the model is likely to perform well on training data but poorly on unseen data. By restricting the number of features being looked at for each split, The probability that the training labels will be predicted well by random chance reduces dramatically.

###### Sampling

As highlighted in the overfitting subsection in phase 2, another potential form of overfitting was due to features from the same participant being in multiple rows. This introduces a risk that the RFC may overfit to training data because rows are similar due to being from the same participant. Sampling was used to mitigate this issue. For each DTC created by the RFC, only a random subsection of the samples was used to build the DTC. The fraction of the samples that were selected randomly is a hyperparameter. As the number of samples was relatively small, bootstrapping was used. This meant that when a subsample of rows was selected for a DTC, it was “replaced” to be used again in the next DTC.

#### Principal Component Analysis (PCA)

When a dataset has a lot of features relative to the number of samples many challenges arise. PCA is an unsupervised method of reducing the number of features with minimised loss of information.

PCA, as used on Scikit Learn [13], works by creating a matrix with all the covariances between each feature in the dataset [14]. It then uses linear algebra and calculus to arrive at a single column of numbers that explains as much of the variance/information as possible. The variables whose variance is explained by the first iteration of this are then removed and this single column is the first principal component. The covariance matrix is then calculated again based on the left-over features. This is repeated multiple times. Each time adds a new component that represents a smaller amount of variance than the one before it. It was seen in phase 2 that 99.9999% of the variance in the dataset, with 736 features, could be explained in 31 components.

This process has the effect of drastically reducing the number of dimensions with minimal loss of information.

While it was used in the project it was not a part of the final models produced. It was decided not to adopt it due to difficulty in interpreting the features. Feature engineering was a primary goal in the project. However, it proved useful in understanding the dataset in phase 2.

### Phase 1: Reproducing the Original Study

For reproducing the original Hyperaktiv study [1], the features used along with the Python code were provided on Github [12]. Running the code on the features produces comparable results.

While stated in the study that the train/test split was 80%/20%, it is assumed based on the confusion matrix provided that the split was 70%/30%. A 10-Fold Cross Validation was performed on the training data and the fitted model for each fold was used to predict for the held-out testing data. Fig 17 shows the results produced by the code. The Cross-Validation refers to the average performance of the model across the 10 folds. The Test Performance refers to the average performance of the 10 models on the held-out testing data.

##### Fig 17 Shows the output from the original code showing relevant performance metrics

A screenshot of a graph

Description automatically generated

#### Feature Selection

The feature selection for the experiment was performed using the tsfresh function “select\_features” [6]. This function performs a univariate analysis on each feature in the dataset and produces a p-value denoting its relevance as a predictor on the target variable. By default, for binary predictors, the Fisher’s Test was used to assess whether variables were independent or not with respect to the labels. For continuous features, the Mann-Whitney U test was used. Importantly, this step was performed before the test/train split. Given the high dimensionality of the dataset, 788 features and 116 rows, the likelihood of variables predicting the label by random chance/noise is high. This is a form of data leakage, known as, Feature Selection Leakage [15] and can lead to overly optimistic performance metrics.

Concerns around the lengths of the timeseries from which the features were extracted were investigated. The healthy control cohort, 32 samples, had much higher number of days on average to the ADHD and non-ADHD cohorts. The mean number of days for the healthy controls was 12.6 days while the ADHD group mean was 6.6 days, and the non-ADHD group had a mean of 7.2 days. Of the 52 features used in modelling, 17 were judged, by inspection, to be related to the length of the timeseries. Some examples are given below.

* Length: Contains the number of points in each timeseries from which the features were extracted
* Count\_above\_mean: Contains the number of values above the mean value.
* Range\_count\_max\_1000000000000.0\_min\_0: Contains number of points with values in the range between 1x1012 and 0.

To assess other less obviously biased features a function was written that found the absolute value of the Pearson’s Correlation Coefficient between each of the variables and the length feature. Features with a correlation of 0.7 or over were discarded. This reduces the number of features to 29.

The remaining variables were then examined using scatter plots against the length column. Fig 20 shows the Augmented Dickey Fuller (ADF) test statistic which did not have a correlation above 0.7 but is clearly inversely related to the length of the timeseries.

##### Augmented Dickey-Fuller (ADF)

###### Test Statistic

The ADF test predicts whether a timeseries is stationary over time or not through a hypothesis test. The ADF test statistic is calculated for a timeseries and compared to a critical value in order reject or fail to reject the null hypothesis. The test statistic is sensitive to the number of points in the timeseries. To illustrate this, the ADF test statistic was calculated for random integers between 0 and 1000 for different numbers of points. The random seed was set to 0 so results are reproducible. Fig 18 shows that there is an obvious inverse relationship. This led to excluding this feature.

###### P-Value

Fig 19 below shows that the p-values for the ADF are not independent of sample size. This feature was also excluded based on fig 19. Similarly to fig 18, the p-values were calculated for random integers with a random seed of 0. The sample size was varied from 100 to 290. This feature was excluded.

##### Fig 18: Effect of Length of Timeseries on ADF Test Statistic

A graph with blue dots

Description automatically generated

##### Fig 19: Effect of Length of Timeseries on ADF P-Values

A graph with blue dots

Description automatically generated

##### Fig 20 Scatter plot of Dickey Fuller Test Statistic versus Length of Timeseries

A diagram of blue dots

Description automatically generated

Fig 21 shows a variable that is strongly related but not linearly. It returns the number of unique values as a fraction of the total values.

##### Fig 21: Scatter plot of number of unique values as a fraction of the total values against length of timeseries.

A graph with blue dots

Description automatically generated

##### Fig 22 Scatter plot of Location of the first minimum value as a fraction of the length of the timeseries.

A graph of a number of blue dots

Description automatically generated

One plot appeared not to be related to the length column. The first location of the minimum value as a fraction of the total number of points, see fig 22. For all participants In the Hyperaktiv dataset the value is zero, which means, presumably, that they did not move in the first minute of recording data. Only 7 of the healthy controls have zero values. This appears to be related to how the two experiments were conducted and is highly unlikely to be an indicator of ADHD.

The other 28 variables with correlation coefficient below 0.7 were inspected and found not be independent of length of timeseries by visual inspection or analysis.

Following this analysis, all other methodologies were designed to process the data so that every row represented a timeseries of equal length with each row beginning and ending at the same time of day.

#### Phase 2: Motor Activity from the Hyperaktiv dataset

##### Dataset Structure for Motor Activity

As discussed in the data preprocessing section the final dataset structure for modelling involved finding all the complete days from each patient after cleaning out unreliable data. Features were extracted for each day. This resulted in a dataset with 449 rows, having days from 84 participants. 734 features remained after taking out missing value columns and columns with no variance.

##### Dropping The Healthy Controls

Had the study looked at all the three cohorts, i.e. including the healthy controls, there would have been approximately 900 rows. Over the course of the study, it emerged that the work was unlikely to produce impressive performance metrics. It was decided at that point to focus on gaining insights into why certain participants were being well predicted or not. Given the well-controlled nature of the Hyperaktiv dataset and the rich information available on each patient, it was decided to use only that dataset. This facilitated a much deeper analysis of the models produced.

##### Leave One Group Out Cross Validation (LOGO-CV)

Most participants in the dataset had multiple rows associated with them. Therefore, it was essential that no participant had rows in both the training and testing dataset. Also, the relatively small number of participants meant that a simple train/test split would not be suitable as the results would vary significantly with respect to the random split.

Having experimented with k-Fold Cross-Validation (CV) using various numbers of folds, eventually a LOGO-CV strategy was implemented. LOGO is a form of Cross Validation where each participant/group in turn has all their rows taken out of the dataset. Then the rest of the dataset is used to fit a model and then predicts labels on the held-out participant. This is repeated for each participant i.e. 84 times. The results consist of a prediction for each row. These predicted values are then compared to the true values to produce performance metrics. The results can be calculated by row or aggregated using the mean value of the predictions for each participant. This methodology was implemented in place of a held-out testing dataset strategy. All the data was used for the LOGO CV.

While this method was computationally expensive, it had considerable advantages. Among them were that there was no random aspect to analysing the results. Analysing 10 Fold CV required performing experiments across several random states. Much time and effort was taken up with finding a pattern in one random state and, subsequently, not finding it in the next. The only randomness in the LOGO CV experiments were internal to the Classifiers. Also, this method allowed for using almost all the data (all the clean and complete days) while avoiding data leakage and comparing “like-with-like”. All features were comparable and meaningful.

##### Post-Processing

The resulting predictions from the LOGO-CV can be assessed by row or by participant. While both approaches were looked at during experiments, it was deemed more relevant to predict by participant. This was achieved by finding the mean value of the predictions, which were in the form 0/1, for each participant. If the mean value was more than 0.5, which meant that more than half of that participants days were deemed to be “ADHD”, the participant was predicted to have ADHD.

##### Overfitting

This methodology introduced some challenges with overfitting on the training data at the expense of the performance on the unseen data.

##### Multiple Rows for Participants

While there was no data leakage in the testing set, in the training dataset several rows often belonged to the same participant. This was a concern as the Classifier may have been achieving overly optimistic results by remembering patterns associated with the individual participants rather than overall differences between classes. This was mitigated using the sampling hyperparameter discussed in the explanation of the Random Forest Classifier algorithm.

##### High Number of Features

As discussed in the Model Selection subsection the high dimensionality of the dataset also created challenges around overfitting. These challenges were met using the fact that the random forest selects the best features for splits from random subsections of the features. This hyperparameter is explained above in the algorithms used subsection.

#### Investigation into Model Performance

The poor performance of the model was investigated. A Decision Tree Classifier was fitted with LOGO-CV but with only the median activity as a feature, which is not a good predictor. The model scores 100% in all metrics in the training data. The plot of the resulting tree indicates that the model is simply remembering training data by splitting the same variable repeatedly until there is approximately one leaf per row. When an informative feature is used, e.g. Raw Score VarSE from the CPT-II data, a much simpler tree is produced. As discussed in the EDA, all the tsfresh continuous variables (728 of the 734) were inspected through boxplots broken down by label. From a univariate perspective, none of the variable showed any promise. The 6 binary variables were inspected using Mathew’s Correlation Coefficient (MCC) and Bar plots. The strongest MCC was -0.03.

An experiment was performed using Principal Component Analysis to project the features down to 31 components which contain 99.9999% of the variance according to the “explained\_variance\_ratio” provided by Scikit Learn [13]. The features from the resulting dataset were also boxplotted and grouped by label. None of them appeared to predict the label well. Applying the RFC with LOGO CV to the transformed data performed poorly.

The probabilities of the RFC were also analysed and showed that there was very high uncertainty in the model with most of the probablities close to 50%. This is illustrated in fig 23 , and explained in the results section.

This may explain why the model still performed poorly after mitigating the other issues around overfitting through the LOGO-CV and the RFC hyperparameters. When the regularisation and sampling hyperparameters were tuned to find realistic performance metrics on the training data, the models still performed poorly on the held-out groups.

#### Feature Engineering

After performing many experiments on the tsfresh features, without finding a baseline model, two other approaches to feature engineering were attempted. One approach was based on the EDA and involved computing basic statistics for certain time windows in the motor activity values. The other approach, based on the Literature Review, involved producing High Resolution Histograms.

##### Time-Based Features

Based on the plots shown in figure 8 and 9, differences in mean activity were seen between 6.40 – 10.00 in the morning and dramatic spike in the ADHD Male sub cohort was seen between 18.40 – 20.00 in the evening. The mean, standard deviation, skewness, and kurtosis were calculated for each participant for both time-windows. The features were inspected by boxplots grouped by label. None showed a significant difference in the groups. The features were also used to train a model using the LOGO CV with the RFC. The predictions were similar to random guessing.

The same basic statistics were calculated for every hour of the day also. The same analysis was conducted on these variables with similar results.

##### High-Resolution Histogram (HRH)

In a study [9], discussed in the Literature Review, High-Resolution Histograms (HRHs) were a very common source of features for successful classification of ADHD. This method of feature engineering involves sorting the motor activity values for each participant into bins and computing the mean and the variance within each bin. While the data used in the study has many differences to the Hyperaktiv dataset, it did work in 24-hour long timeseries. This prompted the use of HRH with 24 bins. The mean, variance, skewness and kurtosis were calculated for each bin in the sorted data. None of variables appeared to predict univariately when inspected through boxplots grouped by label. The LOGO CV with RFC was used on the features and produced performance metrics close to random guessing. Reducing the number of bins to 12 resulted in the same.

### Phase 3: Binary Classification using CPT-II data

As mentioned in the EDA, many variables given relating to the participants outside of motor activity appear to be excellent predictors of ADHD. Several models were built using different combinations of these variables which all provided metrics of precision, recall and accuracy above 75%. These metrics were provided through a Leave-One-Out CV, explained below, with the RFC. Not restricting the dataset to people who wore the actigraphs increases the number of participants to 99. Also, issues in the LOGO-CV where the training data often had multiple rows from the same patient were not relevant to this methodology.

#### Leave One Out Cross Validation (LOO-CV)

LOO-CV is very similar to the LOGO-CV. It only differs in that it does not have any groups in it. This form of Cross Validation takes one sample/participant out of the dataset, then fits the classifier to the rest of the data. This trained model is then used to predict the held-out sample. It repeats this process for each row. This leaves a prediction for each sample in the dataset from which performance metrics are calculated.

#### Feature Selection and Regularisation

It was decided that the most informative dataset would have features only from the CPT-II test as this is the most objective data. The 8 features selected in the EDA, were used with a RFC with LOO-CV to produce a model with accuracy of 76%. The CCP hyperparameter, ccp\_alpha, was set 0.1. Other patient information was used to analyse the model later in the experiments.

## Results

### Phase 1: Reproducing the Original Study

The first phase of this project was to reproduce the baseline ML models presented in the Hyperaktiv study [1]. As discussed in the modelling section, an analysis of the selected features revealed that all but one were sensitive to the length of the timeseries. The lengths of the timeseries for participants from the healthy controls were much longer on average than the other two cohorts from the Hyperaktiv dataset. The lengths of the non-ADHD cohort timeseries from the Hyperaktiv dataset were also slightly longer on average. The length of time a device was worn for is a property of how the data was gathered, not ADHD. Therefore, the results produced in the original study were deemed unreliable. As mentioned before, this demonstrated the importance of pre-processing the data in a way that compares “like-with-like” i.e., features were extracted from motor activity timeseries of equal length and beginning and ending at the same time of day.

### Phase 2: Motor Activity from the Hyperaktiv dataset

As described in the data preprocessing subsection, the final dataset used tsfresh to extract 788 features for each complete day of data. The tsfresh features for each day formed a row in the modelling dataset.

Over the course of the experiments, it was decided to restrict the modelling to just the Hyperaktiv dataset due to concerns about the control group and the additional information that was available for participants in the Hyperaktiv dataset.

The final methodology involved using a LOGO-CV, with a RFC, on the whole dataset. Concerns around overfitting to the training data were mitigated through a Cost-Complexity Pruning (CCP) regularisation hyperparameter and the use of sampling, with boostrapping, to restrict each DTC to a random subset of the rows while reusing the rows for subsequent DTCs. As discussed in modelling, using random subsamples of rows was intended to mitigate the false signals created by having multiple rows from the same participant in the training data. Despite all of this, no model was produced that had workable performance metrics.

The best result produced was when the top 6 features were picked out based on visual inspection of boxplots broken down by the label. The RFC was used with CCP hyperparameter set to 0.01, sampling set to 0.3 (random 30% of the rows was used for each tree). The number of DTCs was set to 1000 to make the output less sensitive to random fluctuations between DTCs. Bootstrapping was applied by default. All other parameters were left at default values. The model produced a precision of 57% and recall of 67% when predictions were aggregated by participant. How these metrics are calculated is explained in detail below in the Interpreting Results subsection.

##### Fig 23: Histogram of Probabilities from Random Forest Classifier in phase 2

A blue graph with black text

Description automatically generated

Probability values were inspected for the resulting predictions. In the case of Random Forest Classifiers, this is the portion of the trees that voted for a positive (ADHD) outcome. Approximately 65 of the probabilities were above 65% and around 85 were below 40%. All other rows/days were between these figures. The distribution of the probabilities is illustrated in fig 23.

At this point in the project a decision was made to not explore motor activity data further. The reasons were as follows:

* The performance metrics and probabilities produced by the modelling after dealing with all the challenges with cross validation and overfitting were disappointing.
* Exploring other features such as High-Resolution Histograms and Time-Based features did not improve the model performance.
* Reducing the dimensionality of the dataset to 31 features using PCA did not improve the modelling process either.

An alternative option at this point was to reproduce the model that produced a 98% accuracy in the Literature Review [7]. It was judged, at this stage of project, that there was not enough time to complete this. Hence, it was not attempted.

### Phase 3: Binary Classification using CPT-II data

In the final phase of the project, the data provided about each patient was explored. As discussed in the Modelling section, 8 features were selected based on the EDA from the Conner’s Continuous Progression Test (CPT-II) data. Due to time constraints the modelling was not rigorously optimised. A RFC was used with the CCP parameter set to 0.1, the proportion of samples used for each DTC was set to 0.5 (random 50% of rows were used for each tree). The number of DTCs was set to 1000. LOO-CV was used. There were 99 rows, one per participant.

The model had a precision of 74.5%, a recall of 78% and an accuracy of 74.75%. The labels were very well balanced with 50 ADHD participants and 49 non-ADHD.

It is highly likely that this could be improved relatively easily by trying several ML models and tuning the hyperparameters.

#### Interpreting Results

##### Fig 24 Confusion Matrix from Phase 3 of the project using the CPT-II data.

A chart of a graph

Description automatically generated with medium confidence

The confusion matrix in, fig 24, illustrates the performance metrics. Each quadrant is explained below:

* True Positives (TP)
  + The bottom right quadrant shows the number of people predicted to have ADHD (38) that really had ADHD. These are referred to as True Positives (TP)
* True Negatives (TN)
  + The top left quadrant shows the number of people predicted not to have ADHD (36) that really did not have ADHD. These are referred to as True Negatives (TN)
* False Positives (FP)
  + The top right quadrant shows the number of people predicted to have ADHD (13) that did not have ADHD. These are referred to as False Positives (TP)
* False Negatives (FN)
  + The bottom left quadrant shows the number of people predicted to not have ADHD that really did have ADHD (12). These are referred to as False Negatives (FN)

###### Accuracy

The accuracy is a measure of how often the predictions are right. In this case that means adding up the “Trues” (the number of predictions that were correct) and dividing by the total number of predictions. The sum of TPs and TNs amount to 74. The total number of predictions is 99. Therefore, the accuracy is given as 74.75%. Accuracy can be misleading as a metric when there is a lot more of one class/label than another in the dataset. In this case the accuracy, due to the balanced data, is an excellent metric but using multiple metrics and interpreting them carefully is important in analysis.

###### Precision

The precision is a measure of the portion of the positive predictions that are correct. If the model predicts that someone has ADHD, the precision indicates how likely it is that they really have ADHD. This is found by expressing the number of TPs as a fraction of all positive predictions (TPs + FPs). From the confusion matrix we get 38 as a fraction of 51 (38 + 13). This gives a precision of 74.5%.

###### Recall

The recall is a measure of the likelihood that if someone has ADHD, how likely is the model to correctly detect it. This is found by expressing the TPs as a fraction of the total number of people who really had ADHD. The model correctly predicted that 38 people had ADHD out of a total of 50 (38 + 12) people who really had ADHD. This gives a recall of 76%.

##### Analysis of Probabilities

As with the Motor Activity Data, the probabilities associated with each predicted value were investigated using other information available on the patients. Fig 25 shows the distribution of probabilities for the model. In contrast to fig 23, the proportion of small and large probabilities is much higher.

###### Fig 25 Probabilities of Random Forest Model using CPT-II Features

A graph with blue squares

Description automatically generated

For analysis, the probabilities were broken down into 3 categories.

##### Strong False Negatives

The first category investigated were the false negatives that had low probabilities (<0.25). This means that of the 1000 decision trees created for the model, less than 250 predicted that the participants had ADHD. By definition, all participants in this group had ADHD.

4 participants were in this group. All 4 were diagnosed with the inattentive presentation of ADHD (ADD). No other obvious pattern was detected other than that none of them were deemed to have a Substance Abuse issue. The possibility that the CPT-II scores work less well for certain cohorts could be something to investigate in future work.

##### Strong False Positives

This group had participants who were wrongly predicted to have ADHD with a probability greater than 0.75 (i.e. more than 750 out of the 1000 decision trees predicted that they had ADHD).

4 participants met these criteria. All 4 suffered from an anxiety disorder, 3 out of the 4 suffered from bipolar disorder, none had unipolar depression. This suggests that the CPT-II test may be detecting anxiety (or depression) as well as ADHD. Again, this could be fruitful future work.

##### Weak Probabilities

Members of this group had probabilities between 60% and 40%. Signifying that the classifier is highly uncertain. Both correct and incorrect predictions were considered. These are the cohort where the distinction is least clear.

17 participants met the criteria. Interestingly, the only participant who was taking stimulant medication was falsely predicted to not have ADHD. No other patterns were detected in this group.

## Discussion

### Conclusions

#### Phase 1: Reproducing the Original Study

In phase 1 of the study, the baseline results presented with the Hyperaktiv dataset was reproduced and analysed. An analysis of the methodology revealed significant flaws.

##### Supervised Feature Selection

The dataset had 116 rows and 788 features. The feature selection process was done before the test/train split and Cross Validation so the likelihood of overly optimistic results due to noise is very high.

##### Data Leakage

The features selected were analysed in detail in the modelling and results section. Only one of them was found to be independent to of the length feature. None of the features were likely to be predicting ADHD.

In addition to this, there were concerns over the healthy control cohort. These are discussed in detail below.

#### Phase 2: Motor Activity from the Hyperaktiv dataset

In phase 2, preprocessing the data so that every complete day was used to extract all 788 features meant all features were meaningful comparisons and expanded the number of rows to 449. This method allowed removing days with large amounts of missing data without affecting other data from the same participant. The LOGO-CV, along with a RFC produced a robust methodology. However, a workable model was not arrived at. After dealing with concerns around overfitting to training data due to high dimensionality and presence of multiple rows from the same patient, a reliable baseline still was not produced.

#### Phase 3: Binary Classification using CPT-II data

In phase 3, summary statistics from the Conner’s Continuous Progression Test (CPT-II) were used to produce a dataset with 99 rows (a row per participant that completed the test). A RFC was able to produce a model with accuracy of 76% using LOO-CV. Due to time constraints this was not optimised. The probabilities for each prediction were analysed and discussed in detail in the results section.

### Healthy Controls

As mentioned in the dataset section, the healthy controls were taken from a different study [4],[11]. They are made up of 23 hospital workers, 5 student nurses and 4 others. It is not clear what kinds of roles the 23 hospital workers have but if they are “patient-facing”, one might imagine that they would move more than most people in other professions, i.e. office workers. Student nurses would almost certainly move more than most randomly selected healthy people. The mean plots in the EDA certainly show higher movement on average across waking hours in the healthy control cohort. This raises a question over how well the healthy controls represent the wider population as opposed to hospital staff populations.

These concerns were a factor in deciding to exclude them from the second and third phase of the study. In contrast, the gathering of the data for the Hyperaktiv dataset was very well controlled. All participants were referred for either ADHD, Mood Disorders, or Anxiety Disorder. Detailed information was recorded for each patient.

### Future Work

Future work on this dataset could go in many directions:

* A reproduction of the study that achieved 98% accuracy that was described in the Literature Review [11].
* Multiclass classification models could be applied to the datasets for all the phases of the experiment.
* The motor activity EDA could be repeated with an emphasis on the participants that were not predicted well by CPT-II features. Motor activity features engineered based on this EDA could be used with the CPT-II data.

# References

[1] S A. Hicks, A. Stautland, O. Bernt Fasmer, W. Førland, H. L. Hammer, P. Halvorsen, K. Mjeldheim, K. J. Oedegaard, B. Osnes, V. E. G. Syrstad, M. A. Riegler, and P. Jakobsen. (2021, June). HYPERAKTIV: An Activity Dataset from Patients with Attention-Deficit/Hyperactivity Disorder (ADHD). In Proceedings of the 12th ACM Multimedia Systems Conference (MMSys '21). Association for Computing Machinery, New York, NY, USA, 314–319. Available: <https://dl.acm.org/doi/10.1145/3458305.3478454>

[2] B. Klein, G. Damiani-Taraba, A. Koster, J. Campbell, and C. Scholz, ‘Diagnosing attention-deficit hyperactivity disorder (ADHD) in children involved with child protection services: are current diagnostic guidelines acceptable for vulnerable populations?’, *Child: Care, Health and Development*, vol. 41, no. 2, pp. 178–185, 2015, doi: [10.1111/cch.12168](https://doi.org/10.1111/cch.12168).

[3] P. Song, M. Zha, Q. Yang, Y. Zhang, X. Li, and I. Rudan, ‘The prevalence of adult attention-deficit hyperactivity disorder: A global systematic review and meta-analysis’, *J Glob Health*, vol. 11, p. 04009, doi: [10.7189/jogh.11.04009](https://doi.org/10.7189/jogh.11.04009).

**[10] W.-S. Choi, C. equally to this work with: W.-S. Choi, Y. S. W. Y. S. Woo, Y. S. W. S.-M. Wang, H. K. Lim, and W.-M. Bahk, ‘The prevalence of psychiatric comorbidities in adult ADHD compared with non-ADHD populations: A systematic literature review’, *PLoS One*, vol. 17, no. 11, p. e0277175, Nov. 2022, doi: [10.1371/journal.pone.0277175](https://doi.org/10.1371/journal.pone.0277175).**

[4] P. Jakobsen et al. (2020, July), PSYKOSE: A Motor Activity Database of Patients with Schizophrenia, 2020 IEEE 33rd International Symposium on Computer-Based Medical Systems (CBMS), Rochester, MN, USA, 2020, pp. 303-308, Available: <https://ieeexplore.ieee.org/abstract/document/9182896>

[5] Ole Bernt Fasmer, Kristin Mjeldheim, Wenche Førland, Anita L Hansen, Steven Dilsaver, Ketil J Oedegaard, and Jan Øystein Berle. 2015. Motor activity in adult patients with attention deficit hyperactivity disorder. *Psychiatry investigation* 12, 4 (2015), 474.

[6] tsfresh. tsfresh.feature\_selection Package [Online]. Available [https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature\_selection.html#](https://tsfresh.readthedocs.io/en/latest/api/tsfresh.feature_selection.html)

[7] A. Kaur and K. S. Kahlon, ‘Accurate Identification of ADHD among Adults Using Real-Time Activity Data’, *Brain Sci*, vol. 12, no. 7, p. 831, Jun. 2022, doi: [10.3390/brainsci12070831](https://doi.org/10.3390/brainsci12070831).

[8] Niamh O’Mahony, Blanca Florentino-Liano, Juan J Carballo, Enrique Baca-García, and Antonio Artés Rodríguez. 2014. Objective diagnosis of ADHD using IMUs. *Medical engineering & physics* 36, 7 (2014), 922–926.

[9] D. Martín-Martínez, P. Casaseca-de-la-Higuera, S. Alberola-López, J. Andrés-de-Llano, J.A. López-Villalobos, J. Ardura-Fernández, C. Alberola-López, Nonlinear analysis of actigraphic signals for the assessment of the attention-deficit/hyperactivity disorder (ADHD), Medical Engineering & Physics, Volume 34, Issue 9, 2012, Pages 1317-1329, ISSN 1350-4533, <https://doi.org/10.1016/j.medengphy.2011.12.023>. (<https://www.sciencedirect.com/science/article/pii/S135045331100350X> )

[10] O. B. Fasmer *et al.*, ‘Diurnal variation of motor activity in adult ADHD patients analyzed with methods from graph theory’, *PLoS One*, vol. 15, no. 11, p. e0241991, Nov. 2020, doi: [10.1371/journal.pone.0241991](https://doi.org/10.1371/journal.pone.0241991).

[11] E. Garcia-Ceja *et al.*, ‘Depresjon: a motor activity database of depression episodes in unipolar and bipolar patients’, in *Proceedings of the 9th ACM Multimedia Systems Conference*, in MMSys ’18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 472–477. doi: [10.1145/3204949.3208125](https://doi.org/10.1145/3204949.3208125).

[12] ‘simula/hyperaktiv’. Simula, Feb. 29, 2024. Accessed: Apr. 03, 2024. [Online]. Available: <https://github.com/simula/hyperaktiv>

[13] scikit-learn: machine learning in Python — scikit-learn 1.4.1 documentation’. Accessed: Apr. 02, 2024. [Online]. Available: <https://scikit-learn.org/stable/>

[14] I. T. Jolliffe and J. Cadima, ‘Principal component analysis: a review and recent developments’, *Philos Trans A Math Phys Eng Sci*, vol. 374, no. 2065, p. 20150202, Apr. 2016, doi: [10.1098/rsta.2015.0202](https://doi.org/10.1098/rsta.2015.0202).

[15] M. A. Bouke and A. Abdullah, ‘An empirical study of pattern leakage impact during data preprocessing on machine learning-based intrusion detection models reliability’, *Expert Systems with Applications*, vol. 230, p. 120715, Nov. 2023, doi: [10.1016/j.eswa.2023.120715](https://doi.org/10.1016/j.eswa.2023.120715).

# Appendix

## Section 1: Code

### Phase 1: Analysis of variables from Hyperaktiv paper

#####################################################################

# This code is adapted from the original code used in the Hyperaktiv

# experimemts. It imports functions from their utils script.

# This contains the code used to create a large number of plots in

# order to investigate data leakage.

# It also has some especially written functions for analysing

# correlations between certain variables

######################################################################

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from lightgbm import LGBMClassifier

import numpy as np

import pandas as pd

from sklearn.dummy import DummyClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from tsfresh.feature\_selection.selection import select\_features

import re

from utils import k\_fold\_model\_evaluation

def reduce\_dimensions(dataX, thres):

correlations = dataX.corr().abs()

mask = np.triu(np.ones\_like(correlations, dtype=bool))

tri\_df = correlations.mask(mask)

correlations = correlations.reset\_index()

to\_drop = [c for c in tri\_df.columns if any(tri\_df[c]>thres)]

dataX = dataX.drop(to\_drop, axis=1)

return dataX

def clean\_length\_cols(dataX, thresh):

to\_drop = []

per\_cor = []

for col in dataX.columns:

cor = dataX[col].corr(dataX["ACC\_\_length"])

if cor != 1 and cor >=thresh:

to\_drop.append(col)

if cor ==1:

per\_cor.append(col)

dataX = dataX.drop(to\_drop, axis=1)

dataX = dataX.drop(per\_cor, axis=1)

return dataX

# setting random seed for random state to 0

random\_seed = 0

# setting testing percentage of data

test\_ratio = 0.3

# Setting number of splits for k fold crossvalidataion

k\_folds = 10

\_PARAMS\_LORGREG = {

"penalty": "l2", "C": 1.0, "class\_weight": "balanced",

"random\_state": 0, "solver": "liblinear", "n\_jobs": 1

}

\_PARAMS\_RFC = {

"n\_estimators": 1000,

"max\_features": None, "max\_depth": None,

"min\_samples\_split": 2, "min\_samples\_leaf": 1,

"min\_weight\_fraction\_leaf": 0.0,

"max\_leaf\_nodes": None, "bootstrap": True,

"oob\_score": False, "n\_jobs": -1, "random\_state": 0,

"class\_weight": "balanced"

}

\_PARAMS\_XGB = {

"random\_state": random\_seed, "verbosity": 0,

'objective':'binary:logistic'

}

\_PARAMS\_LIGHTGB = {

"random\_state": random\_seed, "verbosity": 0,

"objective": "binary"

}

data = pd.read\_csv("D:\Data Science MTU\Final Project\hyperaktiv\Current\Provided data\orig\_features.csv", index\_col="ID")

data = data.drop("col1", axis=1)

data = data.rename(columns = lambda x:re.sub('"', '', x))

data = data.rename(columns = lambda x:re.sub(',', '', x))

dataX = data.drop("ADHD", axis=1)

dataX = dataX.fillna(0)

dataY = data["ADHD"]

#na\_sum = dataX.isnull().sum(axis=0) # looking for columns with na values

#dataX.columns

# Dropping columns that have na values

# for column in dataX.columns:

# if dataX[column].isnull().sum(axis=0) > 0:

# dataX = dataX.drop([column], axis=1)

# na\_sum = dataX.isnull().sum(axis=0)

# Find relevant features using tsfresh

dataX = dataX.drop(['ACC\_\_first\_location\_of\_minimum', "ACC\_\_augmented\_dickey\_fuller\_\_attr\_pvalue\_\_autolag\_AIC", ], axis=1)

dataX = reduce\_dimensions(dataX, 0.7)

dataX = select\_features(dataX, dataY)

length = dataX["ACC\_\_length"]

dataX = clean\_length\_cols(dataX,0.7)

dataX = dataX.join(length)

c = dataX.columns

dataX.plot(kind="scatter" , y="ACC\_\_length",

x="ACC\_\_augmented\_dickey\_fuller\_\_attr\_teststat\_\_autolag\_AIC",

title="Plot of Augmented Dickey Fuller Test Statistic versus Length of timeseries")

dataX.plot(kind="scatter" , y="ACC\_\_length",

x="ACC\_\_ratio\_value\_number\_to\_time\_series\_length",

title="Plot Ratio value Number versus Length of timeseries")

dataX.plot(kind="scatter" , y="ACC\_\_length",

x="ACC\_\_first\_location\_of\_minimum",

title="Plot of First Location of Minimum versus Length of timeseries")

for col in c:

dataX.plot(kind="scatter" , y="ACC\_\_length",

x=col,

title=col)

#relevant = calculate\_relevance\_table(dataX, dataY)

#dimensions = pd.concat([dataX,dataY],axis=1)

#dimensions.to\_csv("D:\Data Science MTU\Final Project\hyperaktiv\Dataset\dimensions.csv")

#dataX = dataX.drop("ACC\_\_augmented\_dickey\_fuller\_\_attr\_pvalue\_\_autolag\_AIC", axis=1)

dataX.plot(x="ACC\_\_augmented\_dickey\_fuller\_\_attr\_pvalue\_\_autolag\_AIC", y="ACC\_\_augmented\_dickey\_fuller\_\_attr\_teststat\_\_autolag\_AIC", style='o')

#dataX.columns

data.plot(y='ACC\_\_augmented\_dickey\_fuller\_\_attr\_teststat\_\_autolag\_AIC', x="ACC\_\_length", style='o')

data.plot(y='ACC\_\_fft\_coefficient\_\_attr\_abs\_\_coeff\_97', x="ACC\_\_length", style='o')

dataX = dataX[['ACC\_\_augmented\_dickey\_fuller\_\_attr\_teststat\_\_autolag\_AIC','ACC\_\_fft\_coefficient\_\_attr\_abs\_\_coeff\_97']]

scaler = StandardScaler(copy=True)

dataX.loc[:, dataX.columns] = scaler.fit\_transform(dataX[dataX.columns])

X\_TRAIN, X\_TEST, Y\_TRAIN, Y\_TEST = train\_test\_split(

dataX,

dataY,

test\_size=test\_ratio,

random\_state=random\_seed,

stratify=dataY)

metric\_names = ["ACC", "PREC", "REC", "F1", "MCC"]

stratified\_train\_eval, stratified\_test\_eval = k\_fold\_model\_evaluation(DummyClassifier, { "strategy": "stratified", "random\_state": 0 },

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits= k\_folds, random\_state=random\_seed)

most\_frequent\_train\_eval, most\_frequent\_test\_eval = k\_fold\_model\_evaluation(DummyClassifier, { "strategy": "most\_frequent", "random\_state": 0 },

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

prior\_train\_eval, prior\_test\_eval = k\_fold\_model\_evaluation(DummyClassifier, { "strategy": "prior", "random\_state": 0 },

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

minor\_train\_eval, minor\_test\_eval = k\_fold\_model\_evaluation(DummyClassifier, { "strategy": "constant", "random\_state": 0, "constant": 1 },

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

major\_train\_eval, major\_test\_eval = k\_fold\_model\_evaluation(DummyClassifier, { "strategy": "constant", "random\_state": 0, "constant": 0 },

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

random\_train\_eval, random\_test\_eval = k\_fold\_model\_evaluation(DummyClassifier, { "strategy": "uniform", "random\_state": 0 },

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

logreg\_train\_eval, logreg\_test\_eval = k\_fold\_model\_evaluation(LogisticRegression, \_PARAMS\_LORGREG,

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

rfc\_train\_eval, rfc\_test\_eval = k\_fold\_model\_evaluation(RandomForestClassifier, \_PARAMS\_RFC,

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

xgb\_train\_eval, xgb\_test\_eval = k\_fold\_model\_evaluation(XGBClassifier, \_PARAMS\_XGB,

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

gbm\_train\_eval, gbm\_test\_eval = k\_fold\_model\_evaluation(LGBMClassifier, \_PARAMS\_LIGHTGB,

X\_TRAIN, Y\_TRAIN, X\_TEST, Y\_TEST, n\_splits=k\_folds, random\_state=random\_seed)

print("\*\*\*CROSS-VALIDATION PERFORMANCE\*\*\*")

print("MODEL\t" + "\t ".join(metric\_names))

print("Rand\t" + "\t ".join([ "%.2f" % (np.mean(random\_train\_eval[name])) for name in metric\_names ]))

print("Strat\t" + "\t ".join([ "%.2f" % (np.mean(stratified\_train\_eval[name])) for name in metric\_names ]))

print("Minor\t" + "\t ".join([ "%.2f" % (np.mean(minor\_train\_eval[name])) for name in metric\_names ]))

print("Major\t" + "\t ".join([ "%.2f" % (np.mean(major\_train\_eval[name])) for name in metric\_names ]))

print("Prior\t" + "\t ".join([ "%.2f" % (np.mean(prior\_train\_eval[name])) for name in metric\_names ]))

print("LogReg\t" + "\t ".join([ "%.2f" % (np.mean(logreg\_train\_eval[name])) for name in metric\_names ]))

print("RFC\t\t" + "\t ".join([ "%.2f" % (np.mean(rfc\_train\_eval[name])) for name in metric\_names ]))

print("XGB\t\t" + "\t ".join([ "%.2f" % (np.mean(xgb\_train\_eval[name])) for name in metric\_names ]))

print("GBM\t\t" + "\t ".join([ "%.2f" % (np.mean(gbm\_train\_eval[name])) for name in metric\_names ]))

print("\n")

print("\*\*\*TEST PERFORMANCE\*\*\*")

print("MODEL\t" + "\t ".join(metric\_names))

print("Rand\t" + "\t ".join([ "%.2f" % (np.mean(random\_test\_eval[name])) for name in metric\_names ]))

print("Strat\t" + "\t ".join([ "%.2f" % (np.mean(stratified\_test\_eval[name])) for name in metric\_names ]))

print("Minor\t" + "\t ".join([ "%.2f" % (np.mean(minor\_test\_eval[name])) for name in metric\_names ]))

print("Major\t" + "\t ".join([ "%.2f" % (np.mean(major\_test\_eval[name])) for name in metric\_names ]))

print("Prior\t" + "\t ".join([ "%.2f" % (np.mean(prior\_test\_eval[name])) for name in metric\_names ]))

print("LogReg\t" + "\t ".join([ "%.2f" % (np.mean(logreg\_test\_eval[name])) for name in metric\_names ]))

print("RFC\t\t" + "\t ".join([ "%.2f" % (np.mean(rfc\_test\_eval[name])) for name in metric\_names ]))

print("XGB\t\t" + "\t ".join([ "%.2f" % (np.mean(xgb\_test\_eval[name])) for name in metric\_names ]))

print("GBM\t\t" + "\t ".join([ "%.2f" % (np.mean(gbm\_test\_eval[name])) for name in metric\_names ]))

with open("performance\_results", "w") as f:

f.write("\*\*\*CROSS-VALIDATION PERFORMANCE\*\*\*\n")

f.write("MODEL\t" + "\t ".join(metric\_names) + "\n")

f.write("Rand\t" + "\t ".join([ "%.2f" % (np.mean(random\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Strat\t" + "\t ".join([ "%.2f" % (np.mean(stratified\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Minor\t" + "\t ".join([ "%.2f" % (np.mean(minor\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Major\t" + "\t ".join([ "%.2f" % (np.mean(major\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Prior\t" + "\t ".join([ "%.2f" % (np.mean(prior\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("LogReg\t" + "\t ".join([ "%.2f" % (np.mean(logreg\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("RFC\t" + "\t ".join([ "%.2f" % (np.mean(rfc\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("XGB\t" + "\t ".join([ "%.2f" % (np.mean(xgb\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("GBM\t" + "\t ".join([ "%.2f" % (np.mean(gbm\_train\_eval[name])) for name in metric\_names ]) + "\n")

f.write("\n")

f.write("\*\*\*TEST PERFORMANCE\*\*\*\n")

f.write("MODEL\t" + "\t ".join(metric\_names) + "\n")

f.write("Rand\t" + "\t ".join([ "%.2f" % (np.mean(random\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Strat\t" + "\t ".join([ "%.2f" % (np.mean(stratified\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Minor\t" + "\t ".join([ "%.2f" % (np.mean(minor\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Major\t" + "\t ".join([ "%.2f" % (np.mean(major\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("Prior\t" + "\t ".join([ "%.2f" % (np.mean(prior\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("LogReg\t" + "\t ".join([ "%.2f" % (np.mean(logreg\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("RFC\t" + "\t ".join([ "%.2f" % (np.mean(rfc\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("XGB\t" + "\t ".join([ "%.2f" % (np.mean(xgb\_test\_eval[name])) for name in metric\_names ]) + "\n")

f.write("GBM\t" + "\t ".join([ "%.2f" % (np.mean(gbm\_test\_eval[name])) for name in metric\_names ]) + "\n")

### 1.2.0 Phase 2: Extraction of tsfresh features with every full day forming a row

# -\*- coding: utf-8 -\*-

"""

Created on Fri Jan 26 09:33:17 2024

@author: jverl

"""

##############################################################################################

# This code takes in the motor activity timeseries csv files for both the Hyperaktiv dataset

# and the Healthy Controls from the Psykose/Depresjon Dataset

# It also, retrieves information for each patient on their age, sex, medication, and whether

# have ADHD. These were used later in the EDA.

# For each timeseries, each day is extracted seperately into a dataframe. Incomplete days

# are discarded.

# Tsfresh extracts its features for every full day and these form the rows of the dataset.

##############################################################################################

import os

import glob

import pandas as pd

from tsfresh import extract\_features

# Get CSV files list from a hyperaktiv dataset

path\_hyper = "D:\\Data Science MTU\\Final Project\\hyperaktiv\\Dataset\\activity\_data"

csv\_files\_hyper = glob.glob(path\_hyper + "/\*.csv")

# Get CSV files list from psykose dataset

path\_psyk = "D:\\Data Science MTU\\Final Project\\Psykose\\control"

csv\_files\_psyk = glob.glob(path\_psyk + "/\*.csv")

# loading patient\_info to a dataframe

patient\_info\_hyper = pd.read\_csv("D:\\Data Science MTU\\Final Project\\hyperaktiv\\Dataset\\patient\_info.csv",delimiter=";" )

# filtering for people who wore actigraph

patient\_info\_hyper\_ACC = patient\_info\_hyper[patient\_info\_hyper["ACC"]==1]

#patient\_info\_hyper\_ACC["ACC"]

#patient\_info\_hyper\_ACC.columns

patient\_info\_depresjon = pd.read\_csv("D:\\Data Science MTU\\Final Project\\Depresjon\\scores.csv")

patient\_info\_control = patient\_info\_depresjon[23:56]

# converting age intervals to age codes in line with hyperaktiv study

patient\_info\_control["age"][patient\_info\_control["age"] == "20-24"] = 1 # hyperaktiv study starts from 17

patient\_info\_control["age"][patient\_info\_control["age"] == "25-29"] = 1

patient\_info\_control["age"][patient\_info\_control["age"] == "30-34"] = 2

patient\_info\_control["age"][patient\_info\_control["age"] == "35-39"] = 2

patient\_info\_control["age"][patient\_info\_control["age"] == "40-44"] = 3

patient\_info\_control["age"][patient\_info\_control["age"] == "45-49"] = 3

patient\_info\_control["age"][patient\_info\_control["age"] == "50-54"] = 4

patient\_info\_control["age"][patient\_info\_control["age"] == "55-59"] = 4

patient\_info\_control["age"][patient\_info\_control["age"] == "60-64"] = 4

patient\_info\_control["age"][patient\_info\_control["age"] == "65-69"] = 4 # outside of range from hyperaktiv study "50-67"

# cpnverting gender code in control to allign with hyperaktiv

patient\_info\_control["gender"][patient\_info\_control["gender"] == 1] = 0

patient\_info\_control["gender"][patient\_info\_control["gender"] == 2] = 1

all\_days = []

features\_by\_day = pd.DataFrame()

adhd\_list = []

adhd\_df = pd.DataFrame()

age\_list = []

age\_list\_df = pd.DataFrame()

gender\_list = []

gender\_list\_df = pd.DataFrame()

med\_list = []

med\_list\_df = pd.DataFrame()

########################################################################################

# extracting info from hyperaktiv dataset

########################################################################################

# given a filepath and a patient id, returns the timeseries as a dataframe

def read\_activity\_file(filepath, patient\_id):

if patient\_id.split("\_")[0] == "patient":

delimiter= ";"

if patient\_id.split("\_")[0] == "control":

delimiter = ","

data = pd.read\_csv(filepath, delimiter=delimiter)

col\_to\_drop = "date"

if col\_to\_drop in data.columns:

data = data.drop(columns = col\_to\_drop)

data.columns=["timestamp", "activity"]

data["timestamp"]=pd.to\_datetime(data["timestamp"]) # convert time strings to pandas datetime

data["ID"] = patient\_id # adding patient ID for traceability during testing

return data

def get\_day\_frames(data):

data = data.set\_index("timestamp")

day\_frames = [group[1] for group in data.groupby(data.index.date)] # https://stackoverflow.com/questions/21605491/how-to-split-a-pandas-dataframe-or-series-by-day-possibly-using-an-iterator

full\_day\_frames = []

for day in day\_frames:

if day.shape[0] >= 1440:

full\_day\_frames.append(day)

return full\_day\_frames

# Looping through all 85 csv files in hyperaktiv dataset

for filepath in csv\_files\_hyper:

print("reading: ", filepath)

patient\_id = os.path.splitext(os.path.basename(filepath))[0] # extracts patient id from filepath

# extract id number from filepath

# to be used to retrieve patient info from patient\_info\_hyper\_ACC

record\_id = int(patient\_id.split("\_")[2])

data = read\_activity\_file(filepath, patient\_id) # saving timeseries

patient\_days = get\_day\_frames(data)

for day in patient\_days:

day["ID"] = patient\_id

data\_features = extract\_features(day, column\_id="ID", column\_kind=None, column\_value="activity")

all\_days.append(data\_features)

###############################################################################################

# extracting info from psykose dataset

###############################################################################################

for filepath in csv\_files\_psyk:

print("reading: ", filepath)

patient\_id = os.path.splitext(os.path.basename(filepath))[0]

data = read\_activity\_file(filepath, patient\_id) # saving timeseries

data = read\_activity\_file(filepath, patient\_id) # saving timeseries

patient\_days = get\_day\_frames(data)

for day in patient\_days:

day["ID"] = patient\_id

data\_features = extract\_features(day, column\_id="ID", column\_kind=None, column\_value="activity")

all\_days.append(data\_features)

features = pd.concat(all\_days)

features.to\_csv("D:\\Data Science MTU\\Final Project\\hyperaktiv\\Dataset\\features\_by\_day.csv")

###############################################################################################

# Creating dataframe with target (adhd/no adhd)

###############################################################################################

# Looping through all 85 csv files in hyperaktiv dataset

for filepath in csv\_files\_hyper:

print("reading: ", filepath)

patient\_id = os.path.splitext(os.path.basename(filepath))[0] # extracts patient id from filepath

# extract id number from filepath

# to be used to retrieve patient info from patient\_info\_hyper\_ACC

record\_id = int(patient\_id.split("\_")[2])

# finding patient info

# ADHD

get\_adhd = patient\_info\_hyper\_ACC["ADHD"][patient\_info\_hyper\_ACC["ID"]==record\_id].values[0]

# Age

get\_age = patient\_info\_hyper\_ACC["AGE"][patient\_info\_hyper\_ACC["ID"]==record\_id].values[0]

# gender

get\_gender = patient\_info\_hyper\_ACC["SEX"][patient\_info\_hyper\_ACC["ID"]==record\_id].values[0]

# medication

get\_med = patient\_info\_hyper\_ACC["MED"][patient\_info\_hyper\_ACC["ID"]==record\_id].values[0]

adhd\_list.append([patient\_id, get\_adhd])

age\_list.append([patient\_id, get\_age])

gender\_list.append([patient\_id, get\_gender])

med\_list.append([patient\_id, get\_med])

###############################################################################################

# extracting info from psykose dataset

###############################################################################################

for filepath in csv\_files\_psyk:

print("reading: ", filepath)

patient\_id = os.path.splitext(os.path.basename(filepath))[0]

get\_age = patient\_info\_control["age"][patient\_info\_control["number"]==patient\_id].values[0]

get\_gender = patient\_info\_control["gender"][patient\_info\_control["number"]==patient\_id].values[0]

adhd\_list.append([patient\_id, 0]) # don't have adhd

age\_list.append([patient\_id, get\_age])

gender\_list.append([patient\_id, get\_gender])

med\_list.append([patient\_id, 0]) # not on medication

age\_list\_df = pd.DataFrame(age\_list, columns=["ID", "AGE"])

gender\_list\_df = pd.DataFrame(gender\_list, columns=["ID", "SEX"])

med\_list\_df = pd.DataFrame(med\_list, columns=["ID", "MED"])

adhd\_df = pd.DataFrame(adhd\_list , columns=["ID", "ADHD"])

features = pd.read\_csv("D:\\Data Science MTU\\Final Project\\hyperaktiv\\Dataset\\features\_by\_day.csv")

predict\_dataset\_2 = features.merge(right=age\_list\_df, on="ID")

predict\_dataset\_2 = predict\_dataset\_2.merge(right=gender\_list\_df , on="ID")

predict\_dataset\_2 = predict\_dataset\_2.merge(right=med\_list\_df , on="ID")

predict\_dataset\_2 = predict\_dataset\_2.merge(right=adhd\_df, on="ID")

predict\_dataset\_2.to\_csv("D:\\Data Science MTU\\Final Project\\hyperaktiv\\Dataset\\predict\_by\_day.csv")

### 1.2.1 Phase 2: Modelling Experiments for tsfresh dataset

###############################################################################################################

# This code reads in the cleaned tsfresh features and the labels. It removes columns with all

# missing values and columns with no variance. It assigns a group number for each participant

# then uses these to set up the Leave-One-Group-Out Cross Validation along with a Random Forest Classifier.

# In this version, the 6 best features based on EDA are selected.

# The code can, and has been, easily adapted to perform various other experiments.

# The results are then aggregated by participant

###############################################################################################################

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.linear\_model import LogisticRegression

from sklearn.feature\_selection import VarianceThreshold

from sklearn.model\_selection import LeaveOneGroupOut, RandomizedSearchCV, GridSearchCV

import matplotlib.pyplot as plt

import pandas as pd

import re

from sklearn.metrics import precision\_recall\_fscore\_support, accuracy\_score

from sklearn.preprocessing import StandardScaler

def cut\_rows(data\_by\_day, patient\_info, max\_days):

patient\_id = list(patient\_info.index)

day\_list = []

for patient in patient\_id:

day\_list.append(data\_by\_day[data\_by\_day.index==patient])

cut\_day\_list = []

for df in day\_list:

if df.shape[0] >=max\_days:

df\_cut = df[:max\_days]

cut\_day\_list.append(df\_cut)

else:

cut\_day\_list.append(df)

cut\_data\_by\_day = pd.concat(cut\_day\_list, axis=0)

return cut\_data\_by\_day

# Dropping columns that have na values

def clean\_missing\_columns(data\_by\_day):

for column in data\_by\_day.columns:

if data\_by\_day[column].isnull().sum(axis=0) > 0:

data\_by\_day = data\_by\_day.drop([column], axis=1)

return data\_by\_day

def variance\_thresh(data\_by\_day, thresh):

var\_thr = VarianceThreshold(threshold=thresh)

data\_by\_day\_X = data\_by\_day.drop("ADHD", axis=1)

data\_by\_day\_Y = data\_by\_day["ADHD"]

var\_thr.fit(data\_by\_day\_X)

selected = data\_by\_day\_X.columns[var\_thr.get\_support()]

data\_by\_day\_X = data\_by\_day[selected]

data\_by\_day = pd.concat([data\_by\_day\_X, data\_by\_day\_Y], axis=1)

return data\_by\_day

def cohort\_filter(data\_by\_day, unwanted):

drops = []

for name in data\_by\_day.index:

cohort = name.split("\_")[0]

if cohort == unwanted:

drops.append(name)

data\_by\_day = data\_by\_day.drop(drops, axis=0)

return data\_by\_day

def grouping(data\_by\_day, patient\_info):

patient\_info["group"] = -1

i = 0

for index in patient\_info.index:

patient\_info.at[index, "group"] = i

i+=1

data\_by\_day = data\_by\_day.join(patient\_info["group"])

return data\_by\_day

def aggregate\_score(results, thresh):

predicted = results["Predicted"]

true = results["True"]

predicted\_df = pd.DataFrame(predicted)

predicted\_df = predicted\_df.set\_index(results\_df.index)

predicted\_df = predicted\_df.groupby(predicted\_df.index).mean(numeric\_only=True)

for index in predicted\_df.index:

if predicted\_df["Predicted"][index] >= thresh:

predicted\_df["Predicted"][index] = 1

else:

predicted\_df["Predicted"][index]= 0

true\_grouped = true.groupby(true.index).median(numeric\_only=True)

metrics = precision\_recall\_fscore\_support(true\_grouped,

predicted\_df,

pos\_label=1,

average="binary")

return metrics

def reduce\_dimensions\_2(data, thres):

correlations = data.corr().abs()

mask = np.triu(np.ones\_like(correlations, dtype=bool))

tri\_df = correlations.mask(mask)

correlations = correlations.reset\_index()

to\_drop = [c for c in tri\_df.columns if any(tri\_df[c]>thres)]

data = data.drop(to\_drop, axis=1)

return data

data\_by\_day = pd.read\_csv("D:\Data Science MTU\Final Project\hyperaktiv\Current\Feature Extraction\Hyperaktiv\hyper\_features\_clean.csv", index\_col="ID")

data\_by\_day = data\_by\_day.drop("col1", axis=1)

# These are the 6 selected features from EDA

selected = ['activity\_\_fft\_coefficient\_\_attr\_"abs"\_\_coeff\_36',

'activity\_\_fft\_coefficient\_\_attr\_"abs"\_\_coeff\_35',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_33',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_15',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_38',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_53'

]

data\_by\_day = data\_by\_day[selected]

patient\_info = pd.read\_csv("D:\Data Science MTU\Final Project\hyperaktiv\Current\Provided data\patient\_info.csv", index\_col="ID", delimiter=";")

patient\_info = patient\_info[patient\_info["ACC"]==1]

labels = patient\_info["ADHD"]

data\_by\_day = data\_by\_day.join(labels)

data\_by\_day = grouping(data\_by\_day, patient\_info)

data\_by\_day = clean\_missing\_columns(data\_by\_day)

data\_by\_day = variance\_thresh(data\_by\_day, 0)

data\_by\_day = data\_by\_day.rename(columns = lambda x:re.sub('"', '', x))

data\_by\_day = data\_by\_day.rename(columns = lambda x:re.sub(',', '', x))

#data\_by\_day = cut\_rows(data\_by\_day, patient\_info, 1)

groups = data\_by\_day["group"]

data\_by\_day = data\_by\_day.drop(["group"], axis=1)

dataX = data\_by\_day.drop(["ADHD"], axis=1)

dataY = data\_by\_day["ADHD"]

logo = LeaveOneGroupOut()

clf = RandomForestClassifier(n\_estimators=1000, ccp\_alpha=0.01, max\_samples=0.3)

# 0.53

features = []

pred\_list = []

true\_list = []

prob\_list = []

indexes = []

# Perform cross-validation

for train\_index, test\_index in logo.split(dataX, dataY, groups=groups):

X\_train, X\_test = dataX.iloc[train\_index], dataX.iloc[test\_index]

y\_train, y\_test = dataY.iloc[train\_index], dataY.iloc[test\_index]

# Fit model on training data

model = clf.fit(X\_train, y\_train)

prob = model.predict\_proba(X\_test)[:,1]

#x = data\_aggX.iloc[data\_aggX.index ==X\_test.index[0]]

#pred = model.predict(x)

#prob = model.predict\_proba(x)[:,1]

pred = model.predict(X\_test)

print("="\*20)

print("patient " + str(y\_test.index[0]))

print("testing:")

print(pred)

#print(data\_aggY.iloc[data\_aggY.index == y\_test.index[0]].values)

print(y\_test.values)

print("="\*10)

print("training accuracy")

print(accuracy\_score(y\_train, model.predict(X\_train)))

print("training precision, recall, F")

print(precision\_recall\_fscore\_support(y\_train, model.predict(X\_train),

pos\_label=1 ,

average="binary"))

length = X\_test.shape[0]

for i in range(length):

pred\_list.append(pred[i])

prob\_list.append(prob[i])

true\_list.append(y\_test.values[i])

indexes.append(X\_test.index[i])

#features.append(model.feature\_importances\_)

#plot\_tree(clf)

results\_dict = {"Predicted":pred\_list,

"True": true\_list,

"Prob": prob\_list}

results\_df = pd.DataFrame(results\_dict, index=indexes)

aggregate\_score(results\_df, 0.5)

precision\_recall\_fscore\_support(results\_df["True"] , results\_df["Predicted"])

f = []

for i in range(len(features)):

y = pd.Series(features[i], index=dataX.columns)

f.append(y)

### 1.3.0 Phase 3: Feature extraction of CPT-II and LOGO-CV Modelling

###############################################################################################################

# This code reads in the cleaned tsfresh features and the labels. It removes columns with all

# missing values and columns with no variance. It assigns a group number for each participant

# then uses these to set up the Leave-One-Group-Out Cross Validation along with a Random Forest Classifier.

# In this version, the 6 best features based on EDA are selected.

# The code can, and has been, easily adapted to perform various other experiments.

# The results are then aggregated by participant

###############################################################################################################

import numpy as np

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier, plot\_tree

from sklearn.linear\_model import LogisticRegression

from sklearn.feature\_selection import VarianceThreshold

from sklearn.model\_selection import LeaveOneGroupOut, RandomizedSearchCV, GridSearchCV

import matplotlib.pyplot as plt

import pandas as pd

import re

from sklearn.metrics import precision\_recall\_fscore\_support, accuracy\_score

from sklearn.preprocessing import StandardScaler

def cut\_rows(data\_by\_day, patient\_info, max\_days):

patient\_id = list(patient\_info.index)

day\_list = []

for patient in patient\_id:

day\_list.append(data\_by\_day[data\_by\_day.index==patient])

cut\_day\_list = []

for df in day\_list:

if df.shape[0] >=max\_days:

df\_cut = df[:max\_days]

cut\_day\_list.append(df\_cut)

else:

cut\_day\_list.append(df)

cut\_data\_by\_day = pd.concat(cut\_day\_list, axis=0)

return cut\_data\_by\_day

# Dropping columns that have na values

def clean\_missing\_columns(data\_by\_day):

for column in data\_by\_day.columns:

if data\_by\_day[column].isnull().sum(axis=0) > 0:

data\_by\_day = data\_by\_day.drop([column], axis=1)

return data\_by\_day

def variance\_thresh(data\_by\_day, thresh):

var\_thr = VarianceThreshold(threshold=thresh)

data\_by\_day\_X = data\_by\_day.drop("ADHD", axis=1)

data\_by\_day\_Y = data\_by\_day["ADHD"]

var\_thr.fit(data\_by\_day\_X)

selected = data\_by\_day\_X.columns[var\_thr.get\_support()]

data\_by\_day\_X = data\_by\_day[selected]

data\_by\_day = pd.concat([data\_by\_day\_X, data\_by\_day\_Y], axis=1)

return data\_by\_day

def cohort\_filter(data\_by\_day, unwanted):

drops = []

for name in data\_by\_day.index:

cohort = name.split("\_")[0]

if cohort == unwanted:

drops.append(name)

data\_by\_day = data\_by\_day.drop(drops, axis=0)

return data\_by\_day

def grouping(data\_by\_day, patient\_info):

patient\_info["group"] = -1

i = 0

for index in patient\_info.index:

patient\_info.at[index, "group"] = i

i+=1

data\_by\_day = data\_by\_day.join(patient\_info["group"])

return data\_by\_day

def aggregate\_score(results, thresh):

predicted = results["Predicted"]

true = results["True"]

predicted\_df = pd.DataFrame(predicted)

predicted\_df = predicted\_df.set\_index(results\_df.index)

predicted\_df = predicted\_df.groupby(predicted\_df.index).mean(numeric\_only=True)

for index in predicted\_df.index:

if predicted\_df["Predicted"][index] >= thresh:

predicted\_df["Predicted"][index] = 1

else:

predicted\_df["Predicted"][index]= 0

true\_grouped = true.groupby(true.index).median(numeric\_only=True)

metrics = precision\_recall\_fscore\_support(true\_grouped,

predicted\_df,

pos\_label=1,

average="binary")

return metrics

def reduce\_dimensions\_2(data, thres):

correlations = data.corr().abs()

mask = np.triu(np.ones\_like(correlations, dtype=bool))

tri\_df = correlations.mask(mask)

correlations = correlations.reset\_index()

to\_drop = [c for c in tri\_df.columns if any(tri\_df[c]>thres)]

data = data.drop(to\_drop, axis=1)

return data

data\_by\_day = pd.read\_csv("D:\Data Science MTU\Final Project\hyperaktiv\Current\Feature Extraction\Hyperaktiv\hyper\_features\_clean.csv", index\_col="ID")

data\_by\_day = data\_by\_day.drop("col1", axis=1)

# These are the 6 selected features from EDA

selected = ['activity\_\_fft\_coefficient\_\_attr\_"abs"\_\_coeff\_36',

'activity\_\_fft\_coefficient\_\_attr\_"abs"\_\_coeff\_35',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_33',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_15',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_38',

'activity\_\_fft\_coefficient\_\_attr\_"angle"\_\_coeff\_53'

]

data\_by\_day = data\_by\_day[selected]

patient\_info = pd.read\_csv("D:\Data Science MTU\Final Project\hyperaktiv\Current\Provided data\patient\_info.csv", index\_col="ID", delimiter=";")

patient\_info = patient\_info[patient\_info["ACC"]==1]

labels = patient\_info["ADHD"]

data\_by\_day = data\_by\_day.join(labels)

data\_by\_day = grouping(data\_by\_day, patient\_info)

data\_by\_day = clean\_missing\_columns(data\_by\_day)

data\_by\_day = variance\_thresh(data\_by\_day, 0)

data\_by\_day = data\_by\_day.rename(columns = lambda x:re.sub('"', '', x))

data\_by\_day = data\_by\_day.rename(columns = lambda x:re.sub(',', '', x))

#data\_by\_day = cut\_rows(data\_by\_day, patient\_info, 1)

groups = data\_by\_day["group"]

data\_by\_day = data\_by\_day.drop(["group"], axis=1)

dataX = data\_by\_day.drop(["ADHD"], axis=1)

dataY = data\_by\_day["ADHD"]

logo = LeaveOneGroupOut()

clf = RandomForestClassifier(n\_estimators=1000, ccp\_alpha=0.01, max\_samples=0.3)

# 0.53

features = []

pred\_list = []

true\_list = []

prob\_list = []

indexes = []

# Perform cross-validation

for train\_index, test\_index in logo.split(dataX, dataY, groups=groups):

X\_train, X\_test = dataX.iloc[train\_index], dataX.iloc[test\_index]

y\_train, y\_test = dataY.iloc[train\_index], dataY.iloc[test\_index]

# Fit model on training data

model = clf.fit(X\_train, y\_train)

prob = model.predict\_proba(X\_test)[:,1]

#x = data\_aggX.iloc[data\_aggX.index ==X\_test.index[0]]

#pred = model.predict(x)

#prob = model.predict\_proba(x)[:,1]

pred = model.predict(X\_test)

print("="\*20)

print("patient " + str(y\_test.index[0]))

print("testing:")

print(pred)

#print(data\_aggY.iloc[data\_aggY.index == y\_test.index[0]].values)

print(y\_test.values)

print("="\*10)

print("training accuracy")

print(accuracy\_score(y\_train, model.predict(X\_train)))

print("training precision, recall, F")

print(precision\_recall\_fscore\_support(y\_train, model.predict(X\_train),

pos\_label=1 ,

average="binary"))

length = X\_test.shape[0]

for i in range(length):

pred\_list.append(pred[i])

prob\_list.append(prob[i])

true\_list.append(y\_test.values[i])

indexes.append(X\_test.index[i])

#features.append(model.feature\_importances\_)

#plot\_tree(clf)

results\_dict = {"Predicted":pred\_list,

"True": true\_list,

"Prob": prob\_list}

results\_df = pd.DataFrame(results\_dict, index=indexes)

aggregate\_score(results\_df, 0.5)

precision\_recall\_fscore\_support(results\_df["True"] , results\_df["Predicted"])

f = []

for i in range(len(features)):

y = pd.Series(features[i], index=dataX.columns)

f.append(y)

## Section 2: Graphs

### 2.1: Graphs of boxplots grouped by ADHD or not from EDA on Patient Information and CPT-II features

#### Fig 2.1.1

A graph with blue and green lines

Description automatically generated

#### Fig 2.1.2

A graph with blue and green squares

Description automatically generated

#### Fig 2.1.3

A graph with blue and green lines and numbers

Description automatically generated

#### Fig 2.1.4

A graph with blue and green squares and dots

Description automatically generated

#### Fig 2.1.5

A graph with numbers and lines

Description automatically generated

#### Fig 2.1.6

A graph with lines and dots

Description automatically generated

#### Fig 2.1.7

A diagram of a graph

Description automatically generated with medium confidence

#### Fig 2.1.8

A graph with lines and numbers

Description automatically generated with medium confidence

#### Fig 2.1.9

A graph with blue and green lines

Description automatically generated

#### Fig 2.1.10

A graph with blue and green lines

Description automatically generated