# Machine Learning on Brain Activity Patterns in Older Adults

# Table of Contents

# 1. Injecting Data into PostgreSQL and Exporting it as CSV Files

This document provides a detailed explanation of the steps to load data from an `.sql` file into PostgreSQL, manage the database and tables, and export data to CSV files. The example assumes the use of the `psql` tool for interacting with the PostgreSQL database.

## 1.1. Connecting to the PostgreSQL Server

1. Open the terminal and run the following command to connect to the PostgreSQL server:

*psql -h localhost -U postgres -d postgres*

Replace `localhost`, `postgres`, and `postgres` with your server address, username, and database name, respectively.

2. You will be prompted to enter the password for the `postgres` user.

3. Once connected, you can check the existing databases:

*\l*

## 1.2. Creating a New Database

1. Create a new database to store the data:

*CREATE DATABASE testdb;*

2. Connect to the new database:

*\c testdb*

## 1.3. Loading Data from the `.sql` File

1. Import the `.sql` file into the `testdb` database using the following command:

*psql -U postgres -d testdb -f /path/to/file.sql*

Replace `/path/to/file.sql` with the actual path to your `.sql` file.

2. Verify the tables have been created:

*\dt*

## 1.4. Managing Tables

1. List all tables in the database:

*\dt*

2. Count the number of rows in each table:

*SELECT COUNT(*) FROM "table_name";*

Replace `table_name` with the name of the table.

3. Drop any table if needed:

*DROP TABLE IF EXISTS "table_name";*

## 1.5. Exporting Data to CSV Files

1. To export data from a table to a CSV file, use the `\copy` command in `psql`. For example:

*\copy "table_name" TO 'C:/path/to/output/file.csv' DELIMITER ',' CSV HEADER;*

Replace `table_name` with the name of the table and `C:/path/to/output/file.csv` with the desired file path.

2. Example commands for exporting all tables:

*\copy "one_back" TO 'C:/Users/Admin/Desktop/brain activity/one_back.csv' DELIMITER ',' CSV HEADER;*
*\copy "onset_nback" TO 'C:/Users/Admin/Desktop/brain activity/onset_nback.csv' DELIMITER ',' CSV HEADER;*
*\copy "onset_rest" TO 'C:/Users/Admin/Desktop/brain activity/onset_rest.csv' DELIMITER ',' CSV HEADER;*
*\copy "participant_data" TO 'C:/Users/Admin/Desktop/brain activity/participant_data.csv' DELIMITER ',' CSV HEADER;*
*\copy "three_back" TO 'C:/Users/Admin/Desktop/brain activity/three_back.csv' DELIMITER ',' CSV HEADER;*
*\copy "two_back" TO 'C:/Users/Admin/Desktop/brain activity/two_back.csv' DELIMITER ',' CSV HEADER;*

3. Ensure the file paths are accessible and writable from your system. If permission errors occur, verify the directory permissions or choose a different path.

1.6.

## 1.6. Reference Commands and Outputs

- Connection Information:

*Server [localhost]: localhost*
*Database [postgres]: postgres*
*Port [5432]: 5432*
*Username [postgres]: postgres*
*Password for user postgres:*

- Listing Databases:

*\l*

- Example Row Counts:

*SELECT COUNT(*) FROM "one_back";*
*count*
*-------*
*375742*

- Successful Data Export Confirmation:

*\copy "one_back" TO 'C:/Users/Admin/Desktop/brain activity/one_back.csv' DELIMITER ',' CSV HEADER;*
*COPY 375742*

# 2. Data Preparation Report and Brain Activity Dataset

The goal is to prepare the dataset for subsequent analysis and machine learning modeling. The data comes from various sources including brain activity data for different tasks, participant data, and onset times for n-back and rest tasks.

## 2.1. File Loading and Initial Inspection

The initial step in the data preparation process involved loading data from several CSV files. The following files were loaded to gather data regarding brain activity, participant information, and task-specific onsets:
- one_back.csv
- two_back.csv
- three_back.csv
- onset_nback.csv
- onset_rest.csv
- participant_data.csv

These files contain information regarding brain activity during different tasks, participant demographics, and n-back task onsets. Each file was read using the pandas library, and the first row of each file was printed to confirm the data structure.

## 2.2. Data Merging and Task Identification

Once the individual files were loaded, the brain activity data from the one-back, two-back, and three-back tasks were combined. A 'task' column was added to each dataset to specify which task the data corresponds to. The following steps were performed:
- A new column called 'task' was added to the one_back, two_back, and three_back data frames to label the task type.
- The datasets for each task were concatenated vertically (along axis=0) to combine them into a single DataFrame, `brain_activity`.

The combined dataset contained brain activity data for 21 channels (both oxygenated and deoxygenated) and other task-specific columns. This dataset was cleaned by removing duplicates to ensure each data point was unique.

## 2.3 Handling Missing Data

It was important to ensure that the dataset was free of missing values before proceeding with analysis. The following steps were taken to handle missing data:
- First, missing values in the combined brain activity dataset were checked using the `.isnull().sum()` function.
- Missing data was not present in the brain activity dataset after removing duplicates, but the participant data contained missing values for participant's MoCA, RBANS, and CRIQ scores.

To fill the missing values, the most frequent value (mode) of each column was used to impute the

missing data for MoCA, RBANS, and CRIQ scores. This ensured that the participant dataset was complete for further processing.

### 2.4. Outlier Detection and Z-Score Normalization

In order to ensure that extreme values (outliers) did not affect subsequent analyses, the Z-score normalization method was applied to the oxygenated and deoxygenated columns.

The following steps were performed to detect and remove outliers:
- The columns corresponding to oxygenated ('oxy') and deoxygenated ('deoxy') brain activity for all 21 channels were identified.
- The Z-scores for each of these columns were calculated using `scipy.stats.zscore`.
- Absolute Z-scores greater than 3 were considered outliers and removed from the dataset to ensure that extreme values did not affect the analysis.

### 2.5. Feature Engineering

As part of the feature engineering process, new columns were created to summarize the brain activity data.
Two new columns were added to the dataset to represent the average oxygenated and deoxygenated levels across all 21 brain channels:
- `avg_oxy`: The average oxygenated brain activity across all channels.
- `avg_deoxy`: The average deoxygenated brain activity across all channels.

Additionally, the variance of oxygenated and deoxygenated brain activity across all channels was calculated to understand the variability in brain activity for each participant. These features could be valuable for machine learning models to predict brain activity patterns.

### 2.6. Final Dataset and Saving Processed Data

Once all the necessary preprocessing steps were completed, the final dataset was saved to a CSV file for further use in analysis and machine learning modeling.
The cleaned and processed dataset includes the following data:
- Participant data (including demographics and cognitive scores)
- Brain activity data (including oxygenated and deoxygenated brain activity across 21 channels)
- Task-related data (including n-back task labels and average brain activity values)

The processed dataset was saved as `combined_brain_activity.csv` to ensure it could be reused for further analysis and modeling tasks.

## 3. Exploratory Data Analysis (EDA)

### 3.1. Introduction

This document provides a comprehensive explanation of the exploratory data analysis (EDA) conducted on datasets related to brain activity and participant demographics. The goal of this analysis is to uncover patterns, relationships, and distributions within the data, which can serve as a foundation for further modeling and hypothesis testing.

## 3.2. Data Loading and Initial Exploration

The following datasets were utilized in the analysis:
1. **`combined_brain_activity.csv`**: Contains brain activity data including oxy (oxygenated hemoglobin) and deoxy (deoxygenated hemoglobin) levels across multiple channels.
2. **`processed_participent_data.csv`**: Includes participant demographic information, cognitive scores, and group classifications.
3. **`onset_nback.csv`**: Records the onset times and task levels for n-back tasks.
4. **`onset_rest.csv`**: Records the onset times and task levels during rest tasks.

Descriptive statistics were computed for each dataset to summarize the central tendencies, variability, and general structure of the data.

## 3.3. Visualization and Distribution Analysis

### 3.3.1. Oxy and Deoxy Levels Distribution

Histograms were created to visualize the distributions of oxy and deoxy levels across all channels. Kernel Density Estimation (KDE) was used to overlay smooth probability density curves, providing a clear understanding of the data's density at various levels. Blue was used for oxy levels, while red was used for deoxy levels, ensuring clear visual distinction.

### 3.3.2. Correlation Matrix of Oxy and Deoxy Levels

A heatmap representing the correlation matrix for oxy and deoxy levels across channels was plotted. The heatmap employed the 'coolwarm' colormap to visualize both positive and negative correlations. Annotations within the heatmap display the correlation coefficients for a quantitative understanding. This analysis helps in identifying channels with strong dependencies or inverse relationships.

### 3.3.3. Boxplot for Oxy Levels

Boxplots were used to summarize the distribution of oxy levels across channels. The boxplots highlighted the median, interquartile range (IQR), and potential outliers for each channel. This visualization is particularly useful for detecting anomalies and understanding variability.

## 3.4. Temporal Analysis of Brain Activity

The oxy levels over time for a specific channel were visualized using a line plot. This chart provides insights into temporal trends, patterns, or anomalies that could be linked to specific tasks or events. The x-axis represents the row ID (as a proxy for time), while the y-axis indicates the oxy levels.

## 3.5. Task Onset Data Analysis

### 3.5.1. Distribution of Task Levels

Count plots were employed to analyze the distribution of task levels in both n-back and rest tasks. These plots display the frequency of each task level, providing insights into the task allocation and distribution.

### 3.5.2. Value Counts for Task Levels

The counts of task levels in the n-back dataset were tabulated to quantify the occurrences of each task level. This analysis aids in understanding the dataset's balance and prevalence of specific task conditions.

## 3.6. Participant Data Analysis

### 3.6.1. Demographic Analysis

Demographic data, including participant age, sex, and group classifications, were analyzed:
- **Age Distribution**: A histogram with KDE was plotted to visualize the age distribution, highlighting the central tendencies and spread.
- **Gender Distribution**: A pie chart represented the proportion of male and female participants, offering a quick view of gender balance in the dataset.

### 3.6.2. Cognitive Scores Analysis

The distributions of cognitive scores, including MOCA, RBANS, and CRIQ, were examined:
- **Boxplots**: These were used to summarize the central tendency and spread of cognitive scores, highlighting any outliers.
- **Histograms**: Separate histograms for each score provided detailed insights into their distribution, with KDE curves for additional context.

### 3.6.3. Relationships between Variables

Scatterplots were created to explore relationships between:
- **Age and MOCA Scores**: Grouped by participant categories, this scatterplot highlighted trends in cognitive performance with age.
- **Age and RBANS Scores**: Similar to the MOCA scatterplot, this examined the correlation between age and RBANS performance.

### 3.6.4. Group-wise Cognitive Score Analysis

Boxplots compared MOCA, RBANS, and CRIQ scores across participant groups. These visualizations reveal inter-group differences, helping to identify group-specific cognitive patterns or disparities.

# 4. Model Selection and Training for Predicting patients age using brain activity signals

## 4.1. Introduction

This report presents the analysis, approach, and methodology used for training multiple machine learning models to predict the age of participants based on various features. The dataset used in this analysis includes oxy and deoxy data channels, along with participant-related attributes such as participant group, MoCA score, RBANS score, and CRIQ score. The goal is to select the best-performing machine learning model for predicting participant age based on these input features.

The models chosen for this task are:

- Linear Regression Model
- RandomForest Regressor
- XGBoost Regressor with GPU support

## 4.2. Data Preprocessing

### 4.2.1. Loading the Data

The data from two CSV files was loaded into two separate DataFrames (`data1` and `data2`) using Pandas. `file1` contains the oxy and deoxy channels along with additional participant-related features. `file2` contains the same set of features linked to the participants.

### 4.2.2. Data Merging

The two datasets were merged on the `participant_number` column using the `merge()` function to create a unified dataset, which contains the oxy and deoxy data channels and the associated participant features.

### 4.2.3. Missing Data Handling

Any rows with missing values were dropped from the dataset using `dropna()` to ensure clean data for model training.

### 4.2.4. Feature Selection

The features selected for training the models include oxy and deoxy data channels (`oxy_channel_1` to `oxy_channel_21` and `deoxy_channel_1` to `deoxy_channel_21`), along with additional features like `avg_oxy`, `avg_deoxy`, `participant_group`, `participant_moca`, `participant_rbans`, and `participant_criq`. These features were used as input variables (X), and the target variable (y) was `participant_age`.

### 4.2.5. Data Normalization

The features were normalized using StandardScaler to standardize the data so that each feature has a mean of 0 and a standard deviation of 1. This is crucial for ensuring that all features contribute equally to the model's learning process.

### 4.2.6. Train-Test Split

The dataset was split into training and test sets using a 80-20 ratio, with 80% of the data used for training and 20% used for testing. This split ensures that the model can be evaluated on unseen data.

## 4.3. Machine Learning Models Used:

### 4.3.1. RandomForest Regressor (cuRF)

Random Forest is an ensemble method that aggregates predictions from multiple decision trees to produce a more accurate and stable prediction. It works well for both classification and regression tasks.

GPU Acceleration: The `cuml.ensemble.RandomForestRegressor` was used to take advantage of GPU acceleration, making training faster compared to the CPU version.

### 4.3.2. Linear Regression

Linear Regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a linear equation to the observed data. It aims to predict the value of the dependent variable based on the values of the independent variables.

In its simplest form, Linear Regression models the relationship between the dependent variable and a single independent variable. The model assumes that the relationship between these variables is linear, meaning the change in the dependent variable is proportional to the change in the independent variable.

#### 4.3.3. XGBoost Regressor (XGB)

XGBoost is a widely-used and powerful gradient boosting framework that has been optimized for speed and performance. It is known for its efficiency in handling large datasets and complex patterns. GPU Acceleration: The `XGBRegressor` from XGBoost was used with GPU support (`tree_method='gpu_hist'` and `predictor='gpu_predictor'`) to accelerate the training process on GPUs.

## 4.4. Model Training and Evaluation

### 4.4.1. Training the Model

Each model was trained on the training set (`X_train`, `y_train`) using the respective machine learning algorithm. During training, the `tqdm` library was used to show a progress bar, providing real-time updates on training progress.

### 4.4.2. Model Prediction

After training, predictions were made on the test set (`X_test`).

### 4.4.3. Model Evaluation

Each model's performance was evaluated using two common metrics:
- Mean Absolute Error (MAE): A lower MAE indicates that the model's predictions are closer to the true values.
- $R^2$ Score: This metric represents how well the model fits the data, with higher $R^2$ values indicating better performance.

## 4.5. Model Selection

After evaluating the models, the one with the lowest MAE or highest $R^2$ score was selected as the best model.
A dictionary `models` was created to store each model along with its MAE and $R^2$ values. The best model was selected based on the lowest MAE using the `min()` function.

The selected model was then saved using Joblib for later use. The file `best_model.joblib` contains the model's trained parameters, which can be used for making future predictions.

# 5. Conclusion:

The comprehensive analysis and modeling effort detailed in this report represents a significant step forward in leveraging machine learning to analyze brain activity patterns in older adults. Beginning with robust data ingestion, storage, and transformation processes in PostgreSQL, the data was meticulously prepared to ensure accuracy and reliability. Subsequent preprocessing steps, including handling missing values, removing outliers, and performing feature engineering, enriched the dataset to capture critical aspects of brain activity and participant demographics. Exploratory Data Analysis (EDA) provided valuable insights into the distributions, relationships, and trends within the data,

enabling an informed approach to feature selection and hypothesis formulation. Leveraging machine learning models such as Linear Regression, Random Forest Regressor, and XGBoost Regressor, the study aimed to predict participant age based on a wide array of brain activity signals and cognitive features. The inclusion of GPU-accelerated methods significantly improved the efficiency of model training, ensuring that computational resources were optimally utilized. Through careful evaluation and comparison of model performances, this work lays a strong foundation for understanding the predictive power of brain activity signals and their potential applications in cognitive health research. Ultimately, this report highlights the intersection of neuroscience, data science, and machine learning, demonstrating the transformative potential of these disciplines in studying and understanding brain function in older adults.