RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

RAJALAKSHMI ENGINEERING COLLEGE
(AUTONOMOUS)

THANDALAM

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY IN

ARTIFICAL INTELLIGENCE AND MACHINE LEARNING

A MINI PROJECT
REPORT ON
NEURO SENSORY ANALYSIS (NEUROSIGN)

Submitted by

SURWEESH SP 231501165

SUJITH P 231501163

SWATHI E 231501169

BONAFIDE CERTIFICATE

Certified that this project report "**NEURO INSIGHT SENSORY ANALYSIS**" is the bonafide work of **"SURWEESH S P (231501163), SUJITH (231501163), SWATHI E (231501169)"**

who carried out the project work under my supervision.

Submitted for the Practical Examination held on ----------

SIGNATURE **SIGNATURE**

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Problem Statement

With mental health conditions like depression and ADHD on the rise, there is a pressing need for non-invasive, data-driven tools that can assist in early diagnosis and monitoring. Traditional methods for diagnosing these disorders often rely on subjective assessments and can take time to produce accurate results. The project aims to address these limitations by utilizing EEG data and ma

Introduction

The project focuses on analyzing and predicting neural health conditions, specifically exploring stroke risks and related neurological issues using EEG data. Electroencephalography (EEG) is a widely used technique for monitoring brain activity through electrical signals captured by electrodes placed on the scalp. By leveraging advanced data processing and machine learning techniques, the project aims to identify patterns or anomalies in EEG data that correlate with conditions like Attention Deficit Hyperactivity Disorder (ADHD) or stroke risks. This project can provide critical insights into brain health, paving the way for early detection and better management of neurological disorders.

Novelty

Cluster-Based Approach: Unlike existing models, this solution clusters unlabeled EEG data for depression and ADHD risk without pre-labelled datasets.

Real-Time Validation: An iterative process optimizes clusters until a validation score of 0.75+ is achieved, ensuring high reliability. Post-clustering, a Random Forest algorithm is applied to predict new inputs' mental health states with high accuracy, chain learning to predict mental health conditions based on cognitive load patterns.

Key Features

- EEG Signal Processing: Extraction of key signals from the frontal regions, crucial for identifying attention and emotional imbalances.

- Unsupervised Learning: Clustering the data to separate the mental states (healthy vs. affected by depression/ADHD) without explicit labels.

- Validation-Driven Clustering: Adjusts the clusters based on silhouette validation, ensuring the most accurate and meaningful groups.

- Predictive Analytics: Uses a trained Random Forest model to predict the future likelihood of depression or ADHD from unseen EEG data.

Impact

Provides an efficient method for detecting early signs of ADHD and depression, potentially aiding clinicians and users in monitoring mental health conditions in real-time.

Facilitates more proactive mental health interventions based on objective EEG signal analysis.

Potential Future Applications

This system can be expanded to detect other cognitive and neurological conditions such as anxiety, epilepsy, or even general cognitive load analysis for educational or performance monitoring applications.

By introducing unsupervised learning techniques with rigorous validation, this project offers a unique and innovative tool for mental health monitoring based on EEG data, providing a cost-effective solution that can be integrated into wearable devices for continuous tracking.

Objective

The primary objective of this project is to analyse EEG data to detect and predict neurological conditions, focusing on stroke risk assessment. The goals are:

1. To cluster EEG data and identify patterns linked to neurological anomalies using unsupervised learning.
2. To provide interpretable insights into brain activity based on features like Attention, Meditation, and frequency bands.
3. To develop an automated system for detecting stroke risks and related conditions, with potential applications in early diagnosis and patient monitoring.
4. To enhance our understanding of EEG-based brain activity analysis, facilitating further research in neuroscience and healthcare technology.

SURVEY OF TECHNOLOGY

2.1 SOFTWARE DESCRIPTION

Visual Studio Code

Visual Studio Code (VS Code) is a widely-used, open-source code editor developed by Microsoft that caters to a diverse range of development needs. Known for its user-friendly interface, VS Code combines the simplicity of a text editor with robust developer tools, making it an excellent choice for programming in various languages and frameworks. With features that enhance productivity and streamline workflows, VS Code has become a favoured choice among developers.

Key Features and Benefits

1. **IntelliSense and Autocompletion**: VS Code's IntelliSense provides intelligent code suggestions, helping developers reduce errors and code faster by offering syntax suggestions, function names, and variable autocompletion.
2. **Integrated Debugging**: With built-in debugging tools, VS Code allows developers to set breakpoints, step through code, and inspect variables, making debugging simpler and more efficient without needing external tools.
3. **Customization and Extensions**: VS Code's extensive marketplace offers extensions for different programming languages, frameworks, and tools, enabling developers to tailor the editor to their project requirements.
4. **Git Integration**: VS Code supports version control directly within the editor through Git integration, allowing developers to commit, push, pull, and manage code versions seamlessly.

2.2 LANGUAGES USED

FRONT END / USER INTERFACE

2.2.1 HTML (HyperText Markup Language)

HTML is the foundational language for creating web content structure. It organizes elements like text, images, and multimedia within a webpage, laying the groundwork for an interactive user experience. By defining sections, forms, and multimedia placement, HTML forms the basis upon which all web content is built.

Purpose in the Project:
In our project, HTML is essential for structuring the interface components and interaction points:

- **User Interaction Forms**: HTML is used to create structured forms for data submission, ensuring that user inputs are collected accurately.
- **Application Data Display**: HTML enables the organized presentation of application data, facilitating intuitive navigation and information retrieval.
- **Content Layout**: HTML organizes content elements, providing a clear structure for user-friendly interaction.

By establishing a well-organized layout, HTML contributes to creating an intuitive and accessible user experience.

2.2.2 CSS (Cascading Style Sheets)

CSS is used to style HTML elements, giving them a polished, visually appealing presentation. It enables developers to define color schemes, fonts, layout spacing, and responsive designs, ensuring the interface is aesthetically pleasing and accessible on multiple devices.

Purpose in the Project:

CSS is essential in our project to create a cohesive, professional design:

1. **Visual Consistency**: CSS maintains a consistent look across the application by styling components like buttons, forms, and tables.
2. **Enhanced Usability**: CSS defines a clear visual hierarchy, making navigation intuitive and aiding users in locating important sections quickly.
3. **Responsive Design**: CSS ensures adaptability across mobile, tablet, and desktop views, delivering a seamless experience regardless of the device.
4. **User Feedback**: CSS animations and transitions provide interactive feedback, enriching the user experience with dynamic visual responses.

2.2.3 JavaScript (Programming Language)

JavaScript is a versatile language that enables interactive web functionalities. It powers user-driven behaviors, handles client-side data processing, and enables responsive interactions. With frameworks like Node.js, JavaScript also powers back-end functionality, enabling full-stack applications.

Purpose in the Project:

JavaScript plays a crucial role in providing dynamic functionality and responsiveness:

- **Real-Time Interactivity**: JavaScript allows interactive behaviors such as updating data displays without reloading the page.
- **Form Validation**: JavaScript validates user input in real-time, enhancing the data accuracy before submission.
- **Client-Server Communication**: JavaScript is used to fetch data asynchronously, facilitating smooth interactions with the back-end database.
- **Session Management**: JavaScript manages application states, maintaining data as users navigate between various sections of the application.

2.3 Database

2.3.1 MySQL (Relational Database Management System)

MySQL is a popular open-source relational database management system (RDBMS) that organizes data into structured tables and supports efficient data management. Known for its reliability, MySQL provides the data integrity and query flexibility necessary for applications that handle user data.

Purpose of MySQL in the Project:

In our project, MySQL is the core database system, enabling data storage and retrieval for user interactions:

- **Structured Data Management**: MySQL organizes application data into tables, storing it in a structured format for easy retrieval and manipulation.
- **Efficient Data Retrieval**: SQL queries enable rapid access to data, such as retrieving user profiles, session histories, and application logs.
- **Data Integrity and Security**: MySQL's transaction capabilities ensure data consistency, especially useful in handling concurrent data access.
- **Scalability**: As a robust RDBMS, MySQL allows our application to scale by managing larger volumes of data and supporting additional users over time.

2.4 Frames and Modules

Node.js and Express.js Frameworks

Node.js

Node.js is a powerful, open-source, cross-platform runtime environment that enables developers to run JavaScript code outside the browser. It is built on Chrome's V8 JavaScript engine and is primarily used for building scalable, fast, and lightweight server-side applications. Node.js uses an event-driven, non-blocking I/O model, making it ideal for applications that handle multiple connections simultaneously, such as real-time chat apps, APIs, and data-intensive web applications.

Key Features of Node.js:

- **Asynchronous and Event-Driven**: Non-blocking execution enhances performance.
- **High Performance**: Powered by the V8 engine for fast code execution.
- **Single Programming Language**: Enables full-stack development using JavaScript.
- **Rich Ecosystem**: npm (Node Package Manager) provides thousands of libraries and modules.

Express.js

Express.js is a minimal and flexible web application framework for Node.js. It simplifies the process of building web applications and APIs by providing a set of robust features and middleware for handling HTTP requests, routing, and more. Express serves as a foundation for building server-side applications and is highly extensible, allowing developers to integrate additional modules as needed.

Key Features of Express.js:

- **Routing**: Efficient and simple routing mechanisms to handle various HTTP requests.
- **Middleware**: Support for middleware functions to process requests and responses.
- **Template Engines**: Integration with template engines like EJS, Pug, or Handlebars for dynamic HTML generation.
- **Extensibility**: Easily integrates with databases, authentication libraries, and other tools.
- **REST API Development**: Ideal for building robust and scalable RESTful APIs.

Node.js + Express.js Together

Express.js is commonly used with Node.js to streamline backend development. While Node.js handles the underlying runtime and server environment, Express provides a framework for structuring the application logic. Together, they form a powerful combination for creating modern, high-performance web and API services.

3.1 Requirement Specification

The project requirements are divided into functional and non-functional requirements to ensure clarity and completeness in addressing both the operational and quality aspects of the system.

3.1.1 Functional Requirements

1. **Data Acquisition and Processing:**
   - The system must read EEG data from the provided sensors or dataset.
   - Preprocess raw EEG data to remove noise and normalize values.
2. **Clustering and Prediction:**
   - Apply unsupervised learning algorithms to cluster the EEG data.
   - Predict stroke risks or neurological conditions based on clustering results.
3. **Visualization:**
   - Generate visualizations of EEG metrics and clustering results for better interpretability.
   - Display scatter plots, cluster patterns, and insights in a user-friendly format.
4. **Export and Reporting:**
   - Allow exporting of processed data and predictions in formats like CSV or PDF.
   - Generate summary reports with key findings and insights.
5. **System Interaction:**
   - Provide an interface for inputting EEG data and viewing predictions.
   - Allow real-time or batch mode processing of EEG data.

3.1.2 Non-Functional Requirements

1. **Performance:**
   - The system must process EEG data efficiently, ensuring minimal latency during clustering and prediction tasks.
2. **Scalability:**
   - The system should handle large datasets, allowing integration with future data sources.
3. **Usability:**
   - The user interface must be simple and intuitive for healthcare professionals or researchers to use.
4. **Accuracy:**
   - Ensure reliable clustering and predictions by using robust machine learning models.

5. **Security:**
    - Protect sensitive EEG data using encryption and secure storage**.**
6. **Maintainability:**
    - The codebase must be modular and well-documented for ease of updates and debugging.

3.2 Hardware and Software Requirements

3.2.1 Hardware Requirements

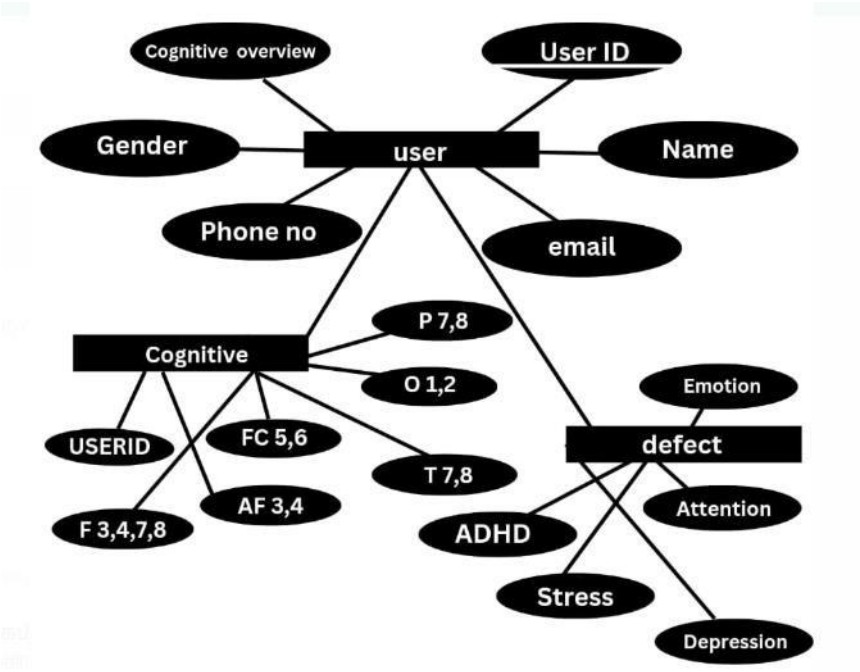
1. **For Local System Deployment:**
    - Processor: Intel i5 or equivalent, 2.5 GHz or higher
    - RAM: 8 GB or more
    - Storage: Minimum 256 GB SSD or equivalent
    - GPU: Recommended for advanced machine learning tasks (e.g., NVIDIA GTX 1050 or higher)
2. **For Cloud Deployment (Optional):**
    - Virtual Machine with 4 vCPUs
    - 16 GB RAM
    - 100 GB SSD storage
    - GPU instance (if neural networks are used)
3. **EEG Hardware (Optional for Real-Time Data):**
    - EEG sensors supporting metrics like Attention, Meditation, Delta, Theta, and Alpha.

3.2.2 Software Requirements

1. **Programming and Libraries:**
    - Python 3.8 or higher
    - Libraries:
        - Data Processing: NumPy, Pandas
        - Visualization: Matplotlib, Seaborn
        - Machine Learning: scikit-learn, TensorFlow/PyTorch (if required)
2. **Development Environment:**
    - Jupyter Notebook or IDE like PyCharm/VS Code
3. **Frameworks:**
    - Node.js (for building APIs or front-end-backend integration)
    - Express.js (for RESTful API development)
4. **Database**:
    - SQLite or PostgreSQL (for storing EEG data and predictions)

5. **Operating System:**
    - Windows 10/11, macOS, or Linux (Ubuntu 20.04 or later)
6. **Optional Tools:**
    - Streamlit (for creating a web-based dashboard)
    - Docker (for containerizing the application for easy deployment)

ER DIAGRAM:



Program Code

Front End

Registration Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Register</title>
    <style>
        body {
            background-color: #000;
            display: flex;
            flex-direction: column;
            align-items: center;
            min-height: 100vh;
            margin: 0;
            font-family: Arial, sans-serif;
        }

        .container {
            background-color: #2c2c2c;
            padding: 30px;
            border-radius: 8px;
            box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
            width: 350px;
            text-align: center;
            border: 1px solid #444;
            margin-top: 20px;
        }

        h1 {
            font-size: 24px;
            color: #fff;
        }

        input {
            width: 100%;
            padding: 12px;
            margin: 10px 0;
            border-radius: 4px;
            border: 1px solid #ccc;
            font-size: 16px;
```

```
        }

        button {
            width: 100%;
            padding: 12px;
            background-color: #0833f3;
            border: none;
            border-radius: 4px;
            color: white;
            font-size: 16px;
            cursor: pointer;
        }

        button:hover {
            background-color: #0452fa;
        }

        .error {
            color: red;
            margin-top: 10px;
        }

        .success {
            color: green;
            margin-top: 10px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Register Here</h1>
    </header>
    <div class="container">
        <form id="registerForm">
            <input type="text" id="name" placeholder="Name" required>
            <input type="email" id="email" placeholder="Email" required>
            <input type="password" id="password" placeholder="Password" required>
            <button type="submit">Register</button>
            <div class="error" id="errorMessage"></div>
            <div class="success" id="successMessage"></div>
            <a href="signin.html">Already have an account? Sign in</a>
        </form>
    </div>

    <script>
        document.getElementById('registerForm').addEventListener('submit', async
function (e) {
            e.preventDefault();

            const name = document.getElementById('name').value;
            const email = document.getElementById('email').value;
```

```
            const password = document.getElementById('password').value;

            // Clear previous messages
            document.getElementById('errorMessage').textContent = '';
            document.getElementById('successMessage').textContent = '';

            try {
                const response = await fetch('http://localhost:209/registration', {
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ name, email, password })
                });

                const data = await response.json();

                if (response.ok) {
                    document.getElementById('successMessage').textContent =
data.message || 'Registered successfully!';
                } else {
                    document.getElementById('errorMessage').textContent =
data.error || 'Registration failed. Please try again.';
                }

            } catch (error) {
                document.getElementById('errorMessage').textContent = 'Error
connecting to server';
                console.error('Error:', error); // Log the error to the console for
debugging
            }
        });
    </script>
</body>
</html>
```

Sign Up

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sign In</title>
    <style>
        /* Overall page styling */
        body {
```

```
        background-color: #000;
        display: flex;
        flex-direction: column;
        align-items: center;
        min-height: 100vh;
        margin: 0;
        font-family: Arial, sans-serif;
        animation: fadeInBackground 1s ease-out forwards;
    }

    /* Navbar styling */
    nav {
        width: 100%;
        display: flex;
        justify-content: space-between;
        padding: 10px;
        background-color: #111;
    }

    nav ul {
        list-style: none;
        margin: 0;
        padding: 0;
        display: flex;
    }

    nav li {
        margin-right: 10px;
    }

    nav a {
        color: #fff;
        text-decoration: none;
    }

    nav .logo {
        color: white;
        font-weight: bold;
        font-size: 20px;
    }

    /* Container styling */
    .container {
        background-color: #2c2c2c;
        padding: 30px;
        border-radius: 8px;
        box-shadow: 0 4px 12px rgba(0, 0, 0, 0.3);
        width: 350px;
        text-align: center;
        border: 1px solid #444;
        animation: fadeIn 1.5s ease-out forwards;
```

```
        opacity: 0;
        transform: translateY(20px);
        margin-top: 20px;
    }

    /* Title styling */
    h1 {
        font-size: 24px;
        color: #fff;
    }

    /* Input field styling with focus animations */
    input {
        width: 100%;
        padding: 12px;
        margin: 10px 0;
        border-radius: 4px;
        border: 1px solid #ccc;
        font-size: 16px;
        transition: border-color 0.3s, box-shadow 0.3s;
    }

    input:focus {
        border-color: #4CAF50;
        box-shadow: 0 0 8px rgba(76, 175, 80, 0.5);
        outline: none;
    }

    /* Button with hover effect */
    button {
        width: 100%;
        padding: 12px;
        background-color: #0833f3;
        border: none;
        border-radius: 4px;
        color: white;
        font-size: 16px;
        cursor: pointer;
        transition: background-color 0.3s, transform 0.3s;
    }

    button:hover {
        background-color: #0452fa;
        transform: translateY(-2px);
    }

    /* Error and success messages styling */
    .error {
        color: red;
        margin-top: 10px;
        animation: fadeInMessage 1s ease-in-out;
```

```
        }

        .success {
            color: green;
            margin-top: 10px;
            animation: fadeInMessage 1s ease-in-out;
        }

        /* Keyframes for animations */
        @keyframes fadeIn {
            from {
                opacity: 0;
                transform: translateY(20px);
            }
            to {
                opacity: 1;
                transform: translateY(0);
            }
        }

        @keyframes fadeInMessage {
            from {
                opacity: 0;
                transform: translateY(-5px);
            }
            to {
                opacity: 1;
                transform: translateY(0);
            }
        }

        @keyframes fadeInBackground {
            from {
                background-color: #111;
            }
            to {
                background-color: #000;
            }
        }
    </style>
</head>
<body>
    <nav>
        <div class="logo">NeuroSign</div>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
    <header>
```

```
        <h1>Sign In</h1>
    </header>
    <div class="container">
        <form id="signInForm">
            <input type="email" id="email" placeholder="Email" required>
            <input type="password" id="password" placeholder="Password" required>
            <button type="submit">Sign In</button>
            <div class="error" id="errorMessage"></div>
            <div class="success" id="successMessage"></div>
            <a href="registration.html" style="color: #fff; margin-top: 10px;
display: inline-block;">Don't have an account? Register</a>
        </form>
    </div>
    <script>
        document.getElementById('signInForm').addEventListener('submit', async
function (e) {
            e.preventDefault();

            const email = document.getElementById('email').value;
            const password = document.getElementById('password').value;

            // Clear previous messages
            document.getElementById('errorMessage').textContent = '';
            document.getElementById('successMessage').textContent = '';

            try {
                const response = await fetch('http://localhost:209/signin', { //
Port adjusted to 2005
                    method: 'POST',
                    headers: {
                        'Content-Type': 'application/json'
                    },
                    body: JSON.stringify({ email, password })
                });

                const data = await response.json();

                if (response.ok) {
                    document.getElementById('successMessage').textContent =
data.message || 'Logged in successfully!';
                } else {
                    document.getElementById('errorMessage').textContent =
data.error || 'Login failed. Please try again.';
                }

            } catch (error) {
                document.getElementById('errorMessage').textContent = 'Error
connecting to server';
                console.error('Error:', error); // Log the error for debugging
            }
        });
```

```
    </script>
</body>
</html>
```

Home Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>NeuroSign</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            background-color: black;
            color: white;
            font-family: 'Arial', sans-serif;
            scroll-behavior: smooth; /* Enable smooth scrolling */
        }

        nav {
            background-color: #333;
            padding: 15px;
            position: sticky;
            top: 0;
            z-index: 1000;
        }

        nav ul {
            list-style-type: none;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: space-between;
            align-items: center;
        }

        nav ul p {
            color: white;
            font-weight: bold;
            margin: 0;
        }

        nav ul li {
            margin-left: 30px;
        }
```

```
        nav ul li a {
            color: white;
            text-decoration: none;
            font-weight: bold;
            transition: color 0.3s;
        }

        nav ul li a:hover {
            color: #00ccff;
        }

        .container {
            text-align: left;
            padding: 50px;
            max-width: 800px;
            margin: auto;
            min-height: 100vh;
        }

        h1 {
            font-size: 2.5em;
            margin: 0;
        }

        .dynamic-description {
            font-size: 1.2em;
            margin-top: 20px;
        }

        .chat-button {
            display: inline-block;
            margin-top: 20px;
            padding: 10px 20px;
            background-color: #00ccff;
            color: black;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-weight: bold;
            transition: background-color 0.3s;
        }

        .chat-button:hover {
            background-color: #0099cc;
        }

        footer {
            text-align: left;
            padding: 20px;
            position: relative;
            bottom: 0;
```

```
        width: 100%;
        background-color: #333;
    }

    .dropdown {
        position: relative;
        display: inline-block;
        margin-top: 20px;
    }

    .dropbtn {
        background-color: #007bff;
        color: white;
        padding: 10px 20px;
        border: none;
        border-radius: 7px;
        cursor: pointer;
        font-size: 16px;
    }

    .dropdown-content {
        display: none;
        position: absolute;
        background-color: #f9f9f9;
        min-width: 160px;
        box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);
        z-index: 1;
    }

    .dropdown-content a {
        color: black;
        padding: 12px 16px;
        text-decoration: none;
        display: block;
    }

    .dropdown-content a:hover {
        background-color: #f1f1f1;
    }

    .dropdown:hover .dropdown-content {
        display: block;
    }

    .dropdown:hover .dropbtn {
        background-color: #0056b3;
    }

    .prediction {
        text-align: center;
        margin-top: 40px;
```

```
        }

        form {
            display: flex;
            flex-direction: column;
            align-items: center;
            margin-top: 20px;
        }

        form div {
            margin-bottom: 15px;
            width: 100%;
            max-width: 300px;
        }

        form label {
            margin-right: 10px;
            font-weight: bold;
        }

        form input {
            padding: 8px;
            width: 100%;
            border: 1px solid #ccc;
            border-radius: 4px;
            background-color: #333;
            color: white;
        }

        .output-area {
            margin-top: 20px;
            margin-bottom: 20px;
            padding: 20px;
            background-color: #222;
            border: 1px solid #444;
            border-radius: 5px;
            max-width: 400px;
            margin-left: auto;
            margin-right: auto;
        }

        .notes-area {
            margin-top: 20px;
            padding: 10px;
            background-color: #222;
            border: 1px solid #444;
            border-radius: 5px;
            max-width: 400px;
            margin-left: auto;
            margin-right: auto;
        }
```

```
    </style>
</head>
<body>
    <nav>
        <ul>
            <li><p>NeuroSign</p></li>
            <li><a href="#home">Home</a></li>
            <li><a href="#about">About</a></li>
            <li><a href="#prediction">Predictive Analysis</a></li> <!-- Updated
link -->
            <li><a href="chatgpt.html">Chat</a></li>
            <li><a href="#services">Services</a></li>
            <li><a href="#contact">Contact</a></li>
        </ul>
    </nav>

    <div class="container">
        <div id="header-content">
            <h1>NeuroSign</h1>
            <h2 id="dynamic-header">Welcome to NeuroSign</h2>
            <p class="dynamic-description">Discover the future of neural monitoring
and analysis.</p>
        </div>

        <p>
            Unlock the potential of your mind with the Neuro Sense Band.
            Our advanced EEG analysis provides real-time insights into your
            mental health and cognitive function. By leveraging AI to interpret
your
            brain activity, you can receive personalized feedback, manage stress,
and enhance focus.
            Monitor your sleep patterns and track your progress over time to foster
            a healthier, more balanced mind. Join us on a journey to better mental
well-being
            through the power of neuro-sensory analysis!
        </p>

        <button class="chat-button"
onclick="window.location.href='chatgpt.html'">Go to Chat</button>
    </div>

    <div class="container" id="prediction"> <!-- Added ID here -->
        <h1>Predictive Analysis</h1>
        <div class="pred dropdown">
            <button class="dropbtn">Cognitive State</button>
            <div class="dropdown-content">
                <a href="#">Link 1</a>
                <a href="#">Link 2</a>
                <a href="#">Link 3</a>
                <a href="#">Link 4</a>
                <a href="#">Link 5</a>
```

```
            </div>
        </div>
        <section class="prediction">
            <form action="#">
                <div>
                    <label for="userid">USER ID:</label>
                    <input type="number" id="userid" placeholder="Enter your User
ID">
                </div>
                <div>
                    <label for="username">USER NAME:</label>
                    <input type="text" id="username" placeholder="Enter your User
Name">
                </div>
                <button class="chat-button" type="submit">Submit</button>
            </form>

            <div class="output-area">
                <h2>Output</h2>
                <p id="output-text">Results will be displayed here.</p>
            </div>
        </section>
    </div>

    <footer>
        &copy; 2024 Neuro Signal Sensing. All rights reserved.
    </footer>

    <script>
        const headers = [
            {
                title: "About Neuro Signs",
                description: "Explore how each of the five senses—sight, sound,
touch, taste, and smell—affects perception and decision-making."
            },
            {
                title: "Future of Mental Health",
                description: "Use real-time feedback to adjust sensory
environments, optimizing experiences based on user responses."
            },
            {
                title: "Unlock Your Mind",
                description: "Unlock the potential of your mind with the Neuro
Sense Band. Our advanced EEG analysis provides real-time insights."
            }
        ];

        let currentIndex = 0;

        function changeHeader() {
            const currentHeader = headers[currentIndex];
```

```
            document.getElementById("dynamic-header").innerText =
currentHeader.title;
            document.querySelector(".dynamic-description").innerText =
currentHeader.description;
            currentIndex = (currentIndex + 1) % headers.length;
        }

        setInterval(changeHeader, 2000); // Change header and content every 2
seconds
    </script>
</body>
</html>
```

Prediction Chat Page

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Chat Page</title>
  <style>
    * {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
      font-family: Arial, sans-serif;
    }

    body {
      display: flex;
      flex-direction: column;
      align-items: center;
      height: 100vh;
      background-color: #f4f4f9;
    }

    .title {
      width: 100%;
      background-color: black;
      color: white;
      padding: 15px;
      text-align: center;
    }

    nav {
      margin-top: 10px;
      text-align: right;
    }
```

```css
nav ul {
  list-style: none;
  padding: 0;
  display: flex;
  justify-content: flex-end;
}

nav li {
  margin-right: 15px;
}

nav a {
  color: #fff;
  text-decoration: none;
  font-weight: bold;
}

nav a:hover {
  color: #ddd;
}

.chat-container {
  width: 350px;
  max-width: 100%;
  background-color: #fff;
  border-radius: 8px;
  box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
  overflow: hidden;
  display: flex;
  flex-direction: column;
  margin-top: 20px;
}

.chat-box {
  height: 400px;
  overflow-y: auto;
  padding: 20px;
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.message {
  padding: 10px 15px;
  border-radius: 20px;
  max-width: 80%;
}

.user-message {
  background-color: #cce5ff;
```

```
      align-self: flex-end;
    }

    .bot-message {
      background-color: #d1e7dd;
      align-self: flex-start;
    }

    .input-container {
      display: flex;
      border-top: 1px solid #ddd;
    }

    #user-input {
      flex: 1;
      padding: 10px;
      border: none;
      outline: none;
    }

    button {
      padding: 10px;
      background-color: #007bff;
      color: #fff;
      border: none;
      cursor: pointer;
    }

    button:hover {
      background-color: #0056b3;
    }
  </style>
</head>
<body>
  <header class="title">
    <h4>CHAT PAGE</h4>
    <nav>
      <ul>
        <li><a href="honepage.html">Home</a></li>
        <li><a href="homepage.html">Predictions</a></li>
        <li><a href="#">Chat</a></li>
        <li><a href="#">Services</a></li>
        <li><a href="#">Contact</a></li>
      </ul>
    </nav>
  </header>

  <div class="chat-container">
    <div id="chat-box" class="chat-box"></div>
    <div class="input-container">
```

```
      <input type="text" id="user-input" placeholder="Type a message..."
onkeypress="handleKeyPress(event)">
      <button onclick="sendMessage()">Send</button>
    </div>
  </div>

  <script>
    async function sendMessage() {
      const userInput = document.getElementById('user-input');
      const messageText = userInput.value.trim();

      if (messageText === '') return;

      // Add user message to the chat box
      addMessage(messageText, 'user-message');

      // Clear the input field
      userInput.value = '';

      // Fetch response from OpenAI API
      try {
        const botResponse = await fetchBotResponse(messageText);
        addMessage(botResponse, 'bot-message');
      } catch (error) {
        console.error('Error fetching bot response:', error);
        addMessage("Sorry, I couldn't retrieve a response.", 'bot-message');
      }
    }

    function addMessage(text, className) {
      const chatBox = document.getElementById('chat-box');
      const message = document.createElement('div');
      message.className = `message ${className}`;
      message.innerText = text;
      chatBox.appendChild(message);

      // Scroll to the bottom of the chat box
      chatBox.scrollTop = chatBox.scrollHeight;
    }

    function handleKeyPress(event) {
      if (event.key === 'Enter') {
        sendMessage();
      }
    }

    // Function to call the OpenAI API and get a response
    async function fetchBotResponse(userMessage) {
      const response = await fetch("https://api.openai.com/v1/chat/completions", {
        method: "POST",
        headers: {
```

```
          "Content-Type": "application/json",
          "Authorization": `Bearer YOUR_API_KEY`  // Replace with your actual API
key
        },
        body: JSON.stringify({
          model: "gpt-3.5-turbo",
          messages: [{ role: "user", content: userMessage }]
        })
      });

      if (!response.ok) {
        throw new Error("API request failed");
      }

      const data = await response.json();
      return data.choices[0].message.content || "No response from bot"; // Adjust
according to your API's response structure
    }
  </script>
</body>
</html>
```

About

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About - NeuroSign</title>
    <style>
        body {
            margin: 0;
            padding: 0;
            background-color: black;
            color: white;
            font-family: 'Arial', sans-serif;
            scroll-behavior: smooth;
        }

        nav {
            background-color: #333;
            padding: 15px;
            position: sticky;
            top: 0;
            z-index: 1000;
        }

        nav ul {
```

```
        list-style-type: none;
        margin: 0;
        padding: 0;
        display: flex;
        justify-content: space-between;
        align-items: center;
    }

    nav ul p {
        color: white;
        font-weight: bold;
        margin: 0;
    }

    nav ul li {
        margin-left: 30px;
    }

    nav ul li a {
        color: white;
        text-decoration: none;
        font-weight: bold;
        transition: color 0.3s;
    }

    nav ul li a:hover {
        color: #00ccff;
    }

    .container {
        text-align: left;
        padding: 50px;
        max-width: 900px;
        margin: auto;
        min-height: 100vh;
    }

    h1 {
        font-size: 2.5em;
        margin-bottom: 20px;
    }

    h2 {
        font-size: 1.8em;
        color: #00ccff;
        margin-top: 0;
    }

    p {
        font-size: 1.2em;
        line-height: 1.6;
```

```
        }

        footer {
            text-align: center;
            padding: 20px;
            position: relative;
            bottom: 0;
            width: 100%;
            background-color: #333;
        }

    </style>
</head>
<body>
    <nav>
        <ul>
            <li><p>NeuroSign</p></li>
            <li><a href="homepage.html">Home</a></li>
            <li><a href="about.html">About</a></li>
            <li><a href="#prediction">Predictive Analysis</a></li>
            <li><a href="chatgpt.html">Chat</a></li>
            <li><a href="#services">Services</a></li>
            <li><a href="#contact">Contact</a></li>
        </ul>
    </nav>

    <div class="container">
        <h1>About NeuroSign</h1>
        <h2>Revolutionizing Mental Health and Cognitive Well-Being</h2>

        <p>
            At NeuroSign, we are at the forefront of neurotechnology innovation. Our mission is to bring the power of EEG-based neuro-sensing devices to users, enabling them to unlock their cognitive potential and improve mental well-being.
        </p>

        <p>
            NeuroSign is powered by a blend of artificial intelligence, real-time data analysis, and neuro-science. By utilizing the Neuro Sense Band, individuals can gain insightful feedback into their brain activity, enabling them to manage stress, enhance focus, and monitor their mental health. We provide personalized reports that help users track progress and make informed decisions regarding their cognitive health.
        </p>

        <h2>Our Vision</h2>
        <p>
            We envision a world where mental health is as easy to monitor and improve as physical health. Through innovative neuro-sensing technology, we aim to provide a deeper understanding of the brain and empower individuals to achieve optimal cognitive and emotional states.
```

```
        </p>

        <h2>Our Technology</h2>
        <p>
            The Neuro Sense Band uses advanced EEG sensors to detect brain activity and provide real-time feedback. Our AI-driven algorithms interpret the data, allowing for accurate, personalized insights. Whether it's tracking sleep, managing stress, or improving focus, NeuroSign ensures that every user gets actionable recommendations for better mental health.
        </p>

        <h2>Why Choose Us?</h2>
        <p>
            NeuroSign is more than just a product—it's a movement toward a healthier, more balanced mind. We offer a unique combination of cutting-edge technology, scientific research, and real-time feedback that no other device on the market provides. By choosing NeuroSign, you are investing in your mental health and cognitive well-being.
        </p>
    </div>

    <footer>
        &copy; 2024 Neuro Sign Sensing. All rights reserved.
    </footer>
</body>
</html>
```

Streamlit Code

```
import streamlit as st
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.cluster import KMeans
from scipy import stats

# Streamlit settings
st.title("EEG Data Analysis and Prediction App")
st.sidebar.header("EEG Dataset Analysis")

# File upload
uploaded_file = st.sidebar.file_uploader("Upload EEG Dataset CSV", type="csv")
if uploaded_file:
```

```
    # Load data from uploaded file
    pf_1 = pd.read_csv(uploaded_file)

    # Dataset overview
    st.write("### Dataset Overview")
    st.write(pf_1.head())

    # Sidebar options
    if st.sidebar.checkbox("Show Dataset Info"):
        st.write("### Dataset Info")
        st.write(pf_1.info())

    if st.sidebar.checkbox("Show Descriptive Statistics"):
        st.write("### Descriptive Statistics")
        st.write(pf_1.describe())

    if st.sidebar.checkbox("Show Missing Values"):
        st.write("### Missing Values")
        st.write(pf_1.isnull().sum())

    if st.sidebar.checkbox("Show Data Duplicates"):
        st.write("### Data Duplicates")
        st.write(pf_1.duplicated().sum())

    # Histogram and Boxplot for each numerical column
    numerical_columns = pf_1.select_dtypes(include=np.number).columns
    if st.sidebar.checkbox("Show Histograms"):
        for col in numerical_columns:
            plt.hist(pf_1[pf_1['eyeDetection'] == 1][col], color='red', alpha=0.5,
label='Eye Movement Detected')
            plt.hist(pf_1[pf_1['eyeDetection'] == 0][col], color='blue', alpha=0.5,
label='No Eye Movement')
            plt.xlabel(col)
            plt.ylabel("Frequency")
            plt.legend()
            st.pyplot(plt)
            plt.clf()

    if st.sidebar.checkbox("Show Boxplots"):
        for col in numerical_columns:
            sns.boxplot(x=pf_1[col])
            plt.title(col)
            st.pyplot(plt)
            plt.clf()

    # Correlation Heatmap
    if st.sidebar.checkbox("Show Correlation Heatmap"):
        st.write("### Correlation Heatmap")
        plt.figure(figsize=(10, 10))
        sns.heatmap(pf_1.corr(), annot=True, fmt=".2f", cmap="coolwarm")
        st.pyplot(plt)
```

```
    # Outlier Detection
    if st.sidebar.checkbox("Show Outliers"):
        z_scores = np.abs(stats.zscore(pf_1.select_dtypes(include=np.number)))
        outliers = np.where(z_scores > 3)
        st.write(f"Outliers detected at: {outliers}")

    # Data Preprocessing and Modeling
    if st.sidebar.checkbox("Run RandomForest Model for Eye Detection"):
        X_eye = pf_1.drop('eyeDetection', axis=1)
        y_eye = pf_1['eyeDetection']

        X_train_eye, X_test_eye, y_train_eye, y_test_eye = train_test_split(X_eye,
y_eye, test_size=0.3, random_state=42)
        scaler = StandardScaler()
        X_train_eye = scaler.fit_transform(X_train_eye)
        X_test_eye = scaler.transform(X_test_eye)

        model_eye = RandomForestClassifier(n_estimators=100, random_state=42)
        model_eye.fit(X_train_eye, y_train_eye)
        y_pred_eye = model_eye.predict(X_test_eye)

        st.write("### RandomForest Classification Report for Eye Detection")
        st.text(classification_report(y_test_eye, y_pred_eye))
        eye_accuracy = accuracy_score(y_test_eye, y_pred_eye)
        st.write(f"RandomForest Model Accuracy: {eye_accuracy * 100:.2f}%")

    # Stress Level Prediction
    if st.sidebar.checkbox("Run DecisionTree Model for Stress Level Prediction"):
        pf_1['beta-alpha-ratio'] = (pf_1['Beta1'] + pf_1['Beta2']) /
(pf_1['Alpha1'] + pf_1['Alpha2'])
        pf_1['stress_level'] = pf_1['beta-alpha-ratio'].apply(lambda x: 2 if x >
1.5 else (1 if x > 1 else 0))

        X_stress = pf_1[['Beta1', 'Beta2', 'Alpha1', 'Alpha2']]
        y_stress = pf_1['stress_level']
        X_train_stress, X_test_stress, y_train_stress, y_test_stress =
train_test_split(X_stress, y_stress, test_size=0.2, random_state=42)

        model_stress = DecisionTreeClassifier(random_state=42)
        model_stress.fit(X_train_stress, y_train_stress)
        y_pred_stress = model_stress.predict(X_test_stress)

        st.write("### DecisionTree Classification Report for Stress Level
Prediction")
        st.text(classification_report(y_test_stress, y_pred_stress))
        stress_accuracy = accuracy_score(y_test_stress, y_pred_stress)
        st.write(f"DecisionTree Model Accuracy: {stress_accuracy * 100:.2f}%")

    # K-Means Clustering for ADHD Prediction
    if st.sidebar.checkbox("Run K-Means Clustering for ADHD Prediction"):
```

```
        X_adhd = pf_1[['Mediation', 'Attention']]
        scaler = StandardScaler()
        X_adhd_scaled = scaler.fit_transform(X_adhd)

        kmeans = KMeans(n_clusters=2, random_state=42)
        pf_1['ADHD_Cluster'] = kmeans.fit_predict(X_adhd_scaled)

        st.write("### K-Means Clustering Results")
        plt.scatter(pf_1['Mediation'], pf_1['Attention'], c=pf_1['ADHD_Cluster'],
cmap='viridis')
        plt.xlabel("Mediation")
        plt.ylabel("Attention")
        plt.title("K-Means Clustering for ADHD Prediction")
        st.pyplot(plt)

        # ADHD Prediction based on clustering
        pf_1['ADHD_Prediction'] = np.where(pf_1['ADHD_Cluster'] == 0, 1, 0)
        st.write("ADHD Prediction (1: ADHD detected, 0: No ADHD)")
        st.write(pf_1[['Mediation', 'Attention', 'ADHD_Prediction']])
```

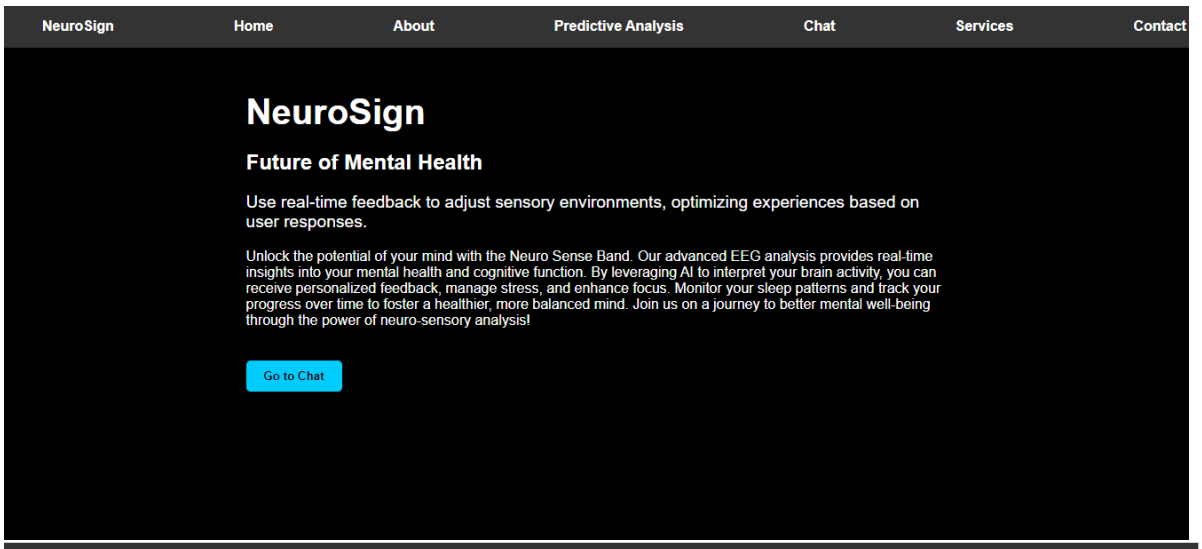
BACK END

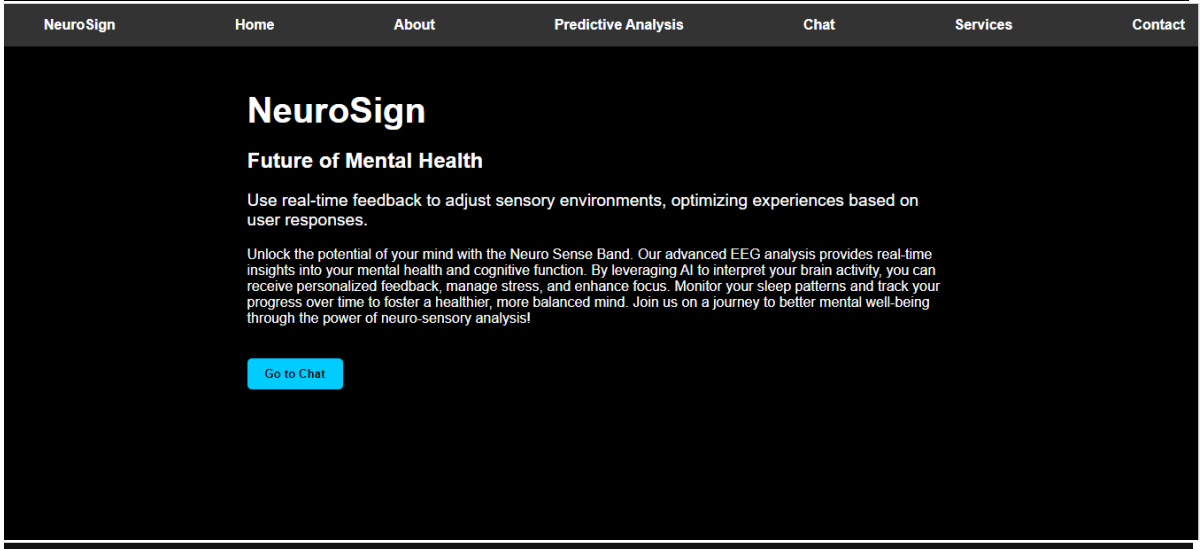
```
const express = require('express');
const sqlite3 = require('sqlite3');
const bcrypt = require('bcrypt');
const bodyParser = require('body-parser');
const path = require('path');


const app = express();
const port = 209;


app.use(bodyParser.json());

app.use(express.static('public'));


const db = new sqlite3.Database('neurosign.db', (err) => {
    if (err) {
        console.error('Error opening database:', err.message);
    } else {
        console.log('Connected to SQLite database');
    }
});


db.run(`CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
```

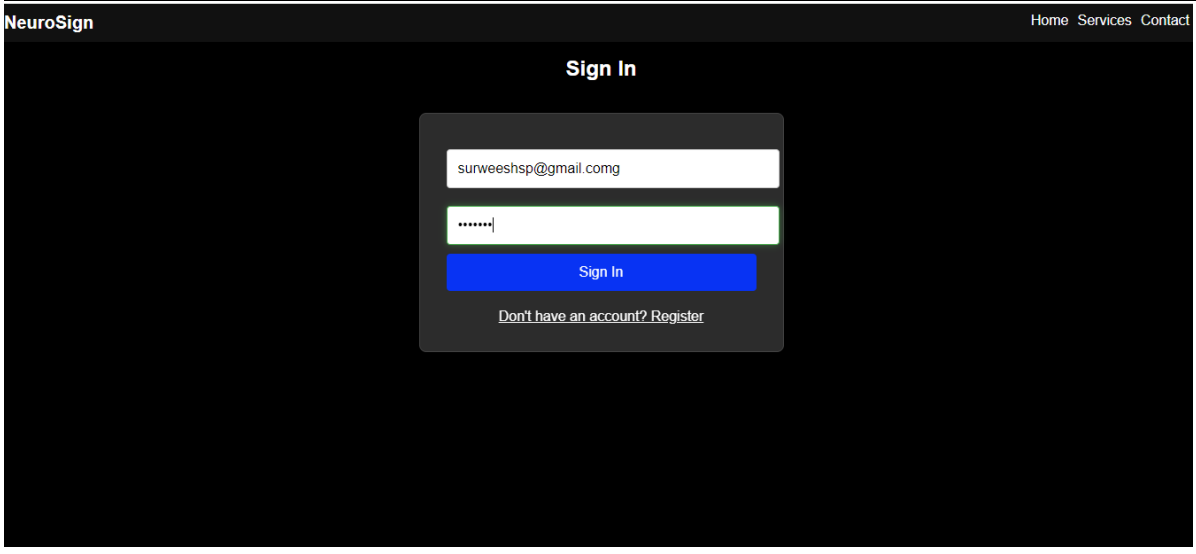
```
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    password TEXT NOT NULL
);`);


app.post('/registration', async (req, res) => {
    const { name, email, password } = req.body;

    if (!name || !email || !password) {
        return res.status(400).json({ error: 'All fields are required' });
    }


    const hashedPassword = await bcrypt.hash(password, 10);

    const query = 'INSERT INTO users (name, email, password) VALUES (?, ?, ?)';
    db.run(query, [name, email, hashedPassword], function (err) {
        if (err) {
            console.error('Error inserting user:', err.message);
            return res.status(500).json({ error: 'Database error: ' + err.message
});
        }


        console.log('Inserted new user with ID:', this.lastID);


        res.status(201).json({
            message: 'User registered successfully',
            userId: this.lastID
        });
    });
});

app.post('/signin', (req, res) => {
    const { email, password } = req.body;

    const query = 'SELECT * FROM users WHERE email = ?';
    db.get(query, [email], async (err, user) => {
        if (err) {
            console.error('Error fetching user:', err.message);
            return res.status(500).json({ error: 'Database error: ' + err.message
});
        }

        if (!user) {
            return res.status(401).json({ error: 'Invalid email or password' });
        }
```

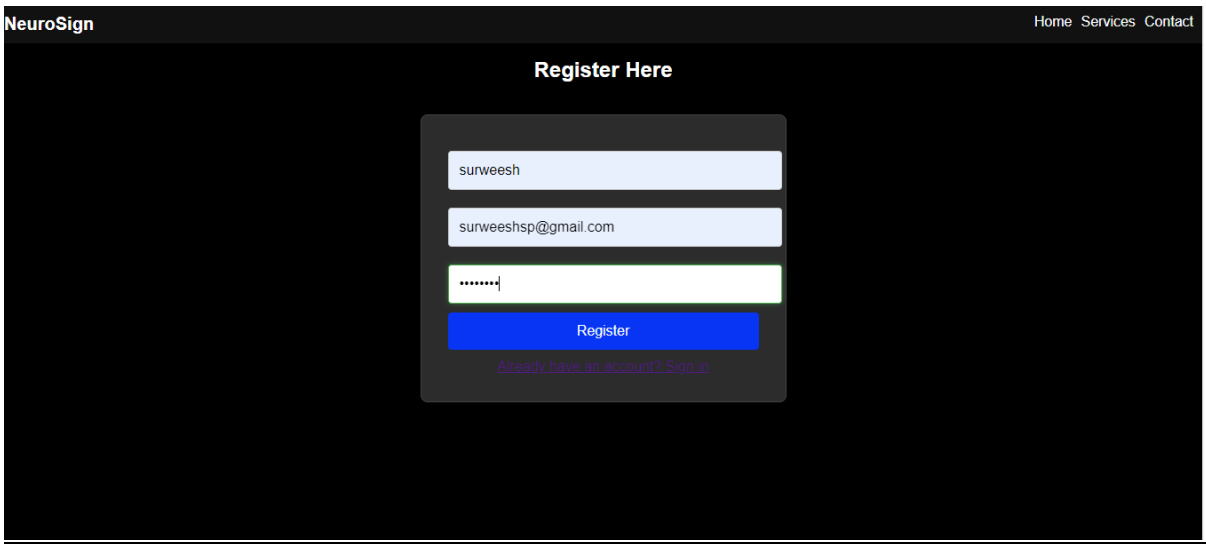
```
        const passwordMatch = await bcrypt.compare(password, user.password);
        if (!passwordMatch) {
            return res.status(401).json({ error: 'Invalid email or password' });
        }


        res.status(200).json({ message: 'Logged in successfully!' });
    });
});

app.get('/homepage', (req, res) => {
    res.sendFile(path.join(__dirname, '/public/homepage.html'));
});

app.get('/chatgpt', (req, res) => {
    res.sendFile(path.join(__dirname, '/public/chatgpt.html'));
});

app.get('/registration', (req, res) => {
    res.sendFile(path.join(__dirname, '/public/registration.html'));
});

app.get('/signin', (req, res) => {
    res.sendFile(path.join(__dirname, '/public/signin.html'));
});


app.listen(port, () => {
    console.log(`Server running at http://localhost:${port}`);
});
```

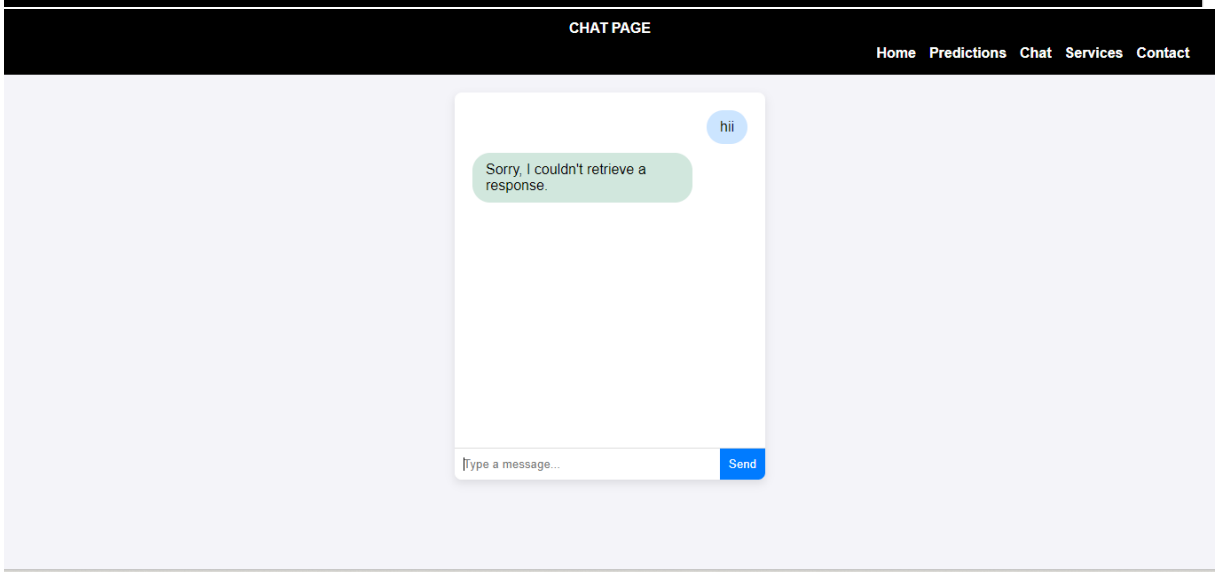
OUTPUT


NeuroSign
Home
About
Predictive Analysis
Chat
Services
Contact
NeuroSign
Future of Mental Health
Use real-time feedback to adjust sensory environments, optimizing experiences based on user responses.
Unlock the potential of your mind with the Neuro Sense Band. Our advanced EEG analysis provides real-time insights into your mental health and cognitive function. By leveraging AI to interpret your brain activity, you can receive personalized feedback, manage stress, and enhance focus. Monitor your sleep patterns and track your progress over time to foster a healthier, more balanced mind. Join us on a journey to better mental well-being through the power of neuro-sensory analysis!
Go to Chat


NeuroSign
Home
About
Predictive Analysis
Chat
Services
Contact
NeuroSign
Future of Mental Health
Use real-time feedback to adjust sensory environments, optimizing experiences based on user responses.
Unlock the potential of your mind with the Neuro Sense Band. Our advanced EEG analysis provides real-time insights into your mental health and cognitive function. By leveraging AI to interpret your brain activity, you can receive personalized feedback, manage stress, and enhance focus. Monitor your sleep patterns and track your progress over time to foster a healthier, more balanced mind. Join us on a journey to better mental well-being through the power of neuro-sensory analysis!
Go to Chat

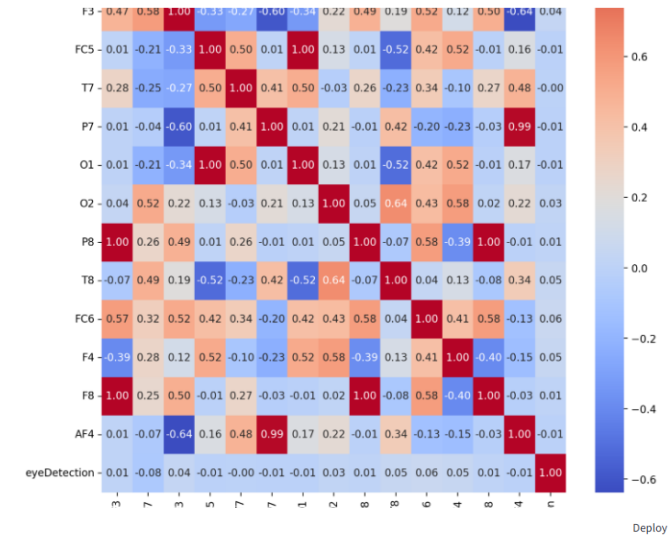

NeuroSign
Home
Services
Contact
Sign In
surweeshsp@gmail.comg
Sign In
Don't have an account? Register


NeuroSign
Home Services Contact
Register Here
surweesh
surweeshsp@gmail.com
Register
Already have an account? Sign in


CHAT PAGE
Home Predictions Chat Services Contact
hii
Sorry, I couldn't retrieve a response.
Type a message...
Send

- [ ] Show Dataset Info
- [x] Show Descriptive Statistics
- [x] Show Missing Values
- [x] Show Data Duplicates
- [x] Show Histograms
- [x] Show Boxplots
- [x] Show Correlation Heatmap
- [ ] Show Outliers
- [ ] Run RandomForest Model for Eye Detection
- [ ] Run DecisionTree Model for Stress Level Prediction
- [ ] Run K-Means Clustering for ADHD Prediction

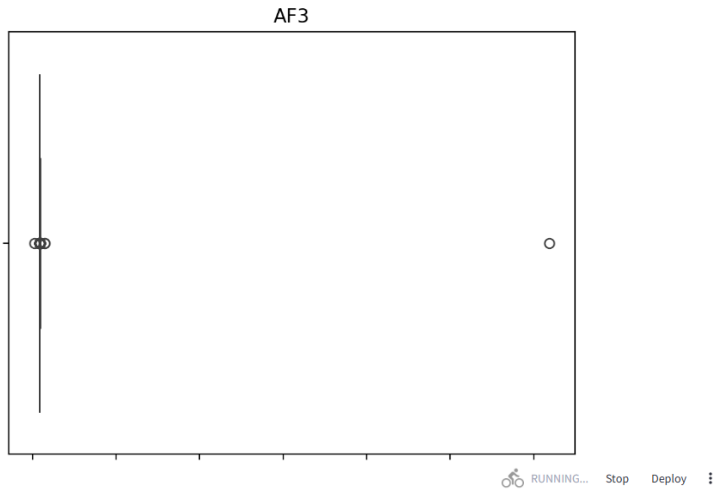
EEG dataset.csv
1.6MB

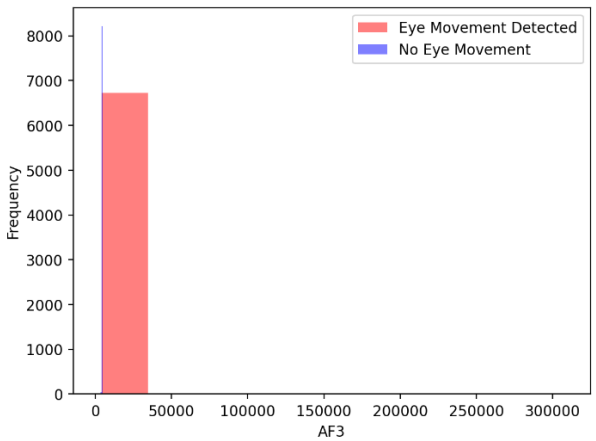


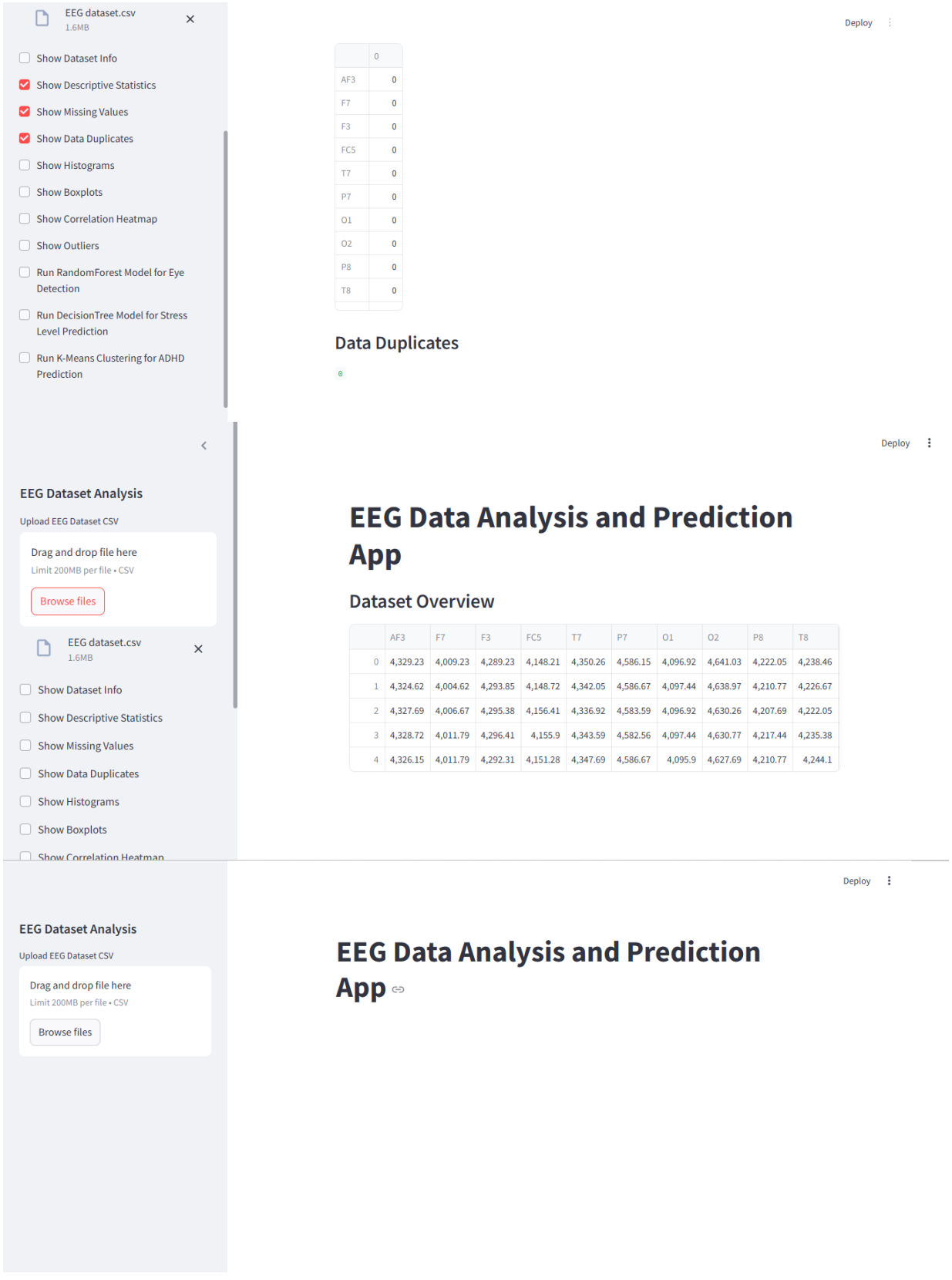
- [ ] Show Dataset Info
- [x] Show Descriptive Statistics
- [x] Show Missing Values
- [x] Show Data Duplicates
- [x] Show Histograms
- [x] Show Boxplots
- [ ] Show Correlation Heatmap
- [ ] Show Outliers
- [ ] Run RandomForest Model for Eye Detection
- [ ] Run DecisionTree Model for Stress Level Prediction
- [ ] Run K-Means Clustering for ADHD Prediction

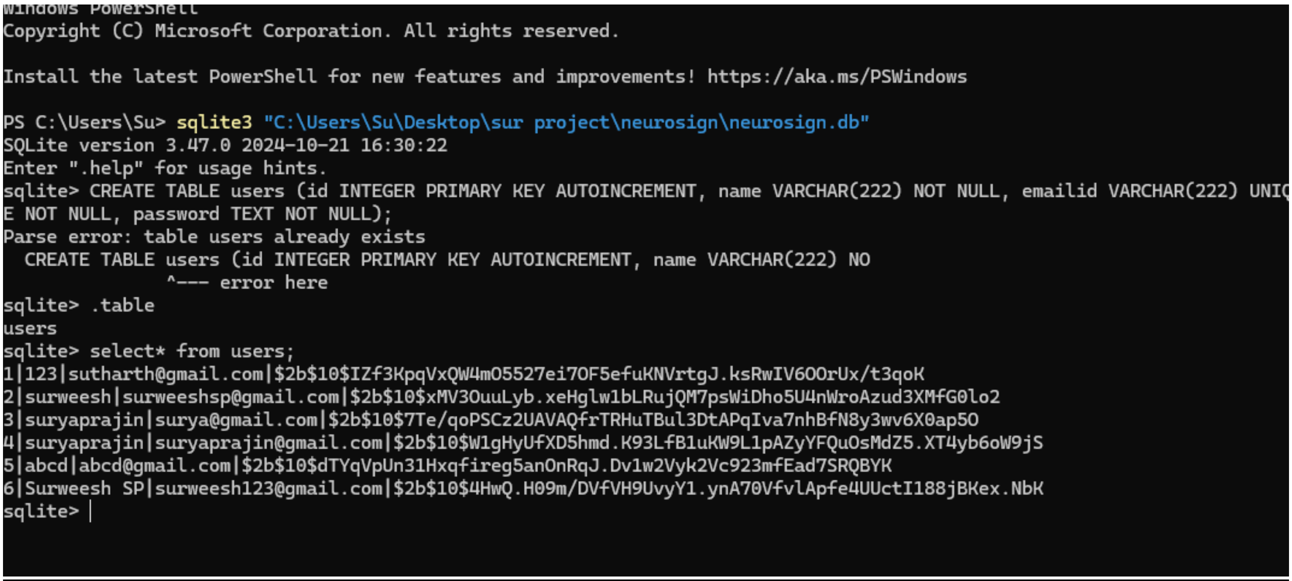
EEG dataset.csv
1.6MB

- [ ] Show Dataset Info
- [x] Show Descriptive Statistics
- [x] Show Missing Values
- [x] Show Data Duplicates
- [x] Show Histograms
- [ ] Show Boxplots
- [ ] Show Correlation Heatmap
- [ ] Show Outliers
- [ ] Run RandomForest Model for Eye Detection
- [ ] Run DecisionTree Model for Stress Level Prediction
- [ ] Run K-Means Clustering for ADHD Prediction

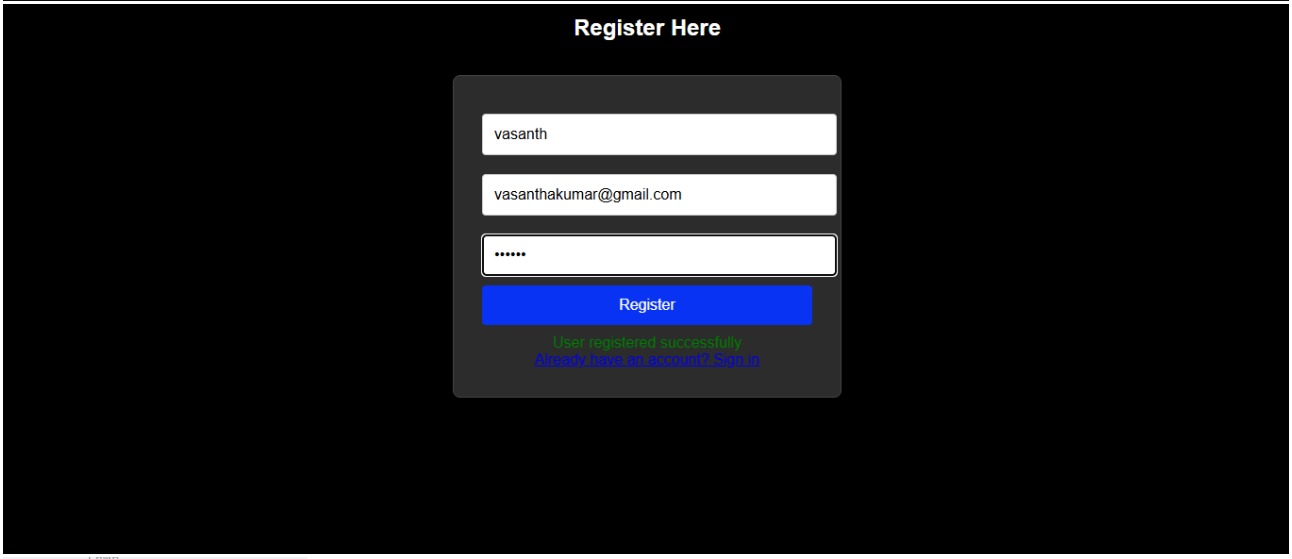




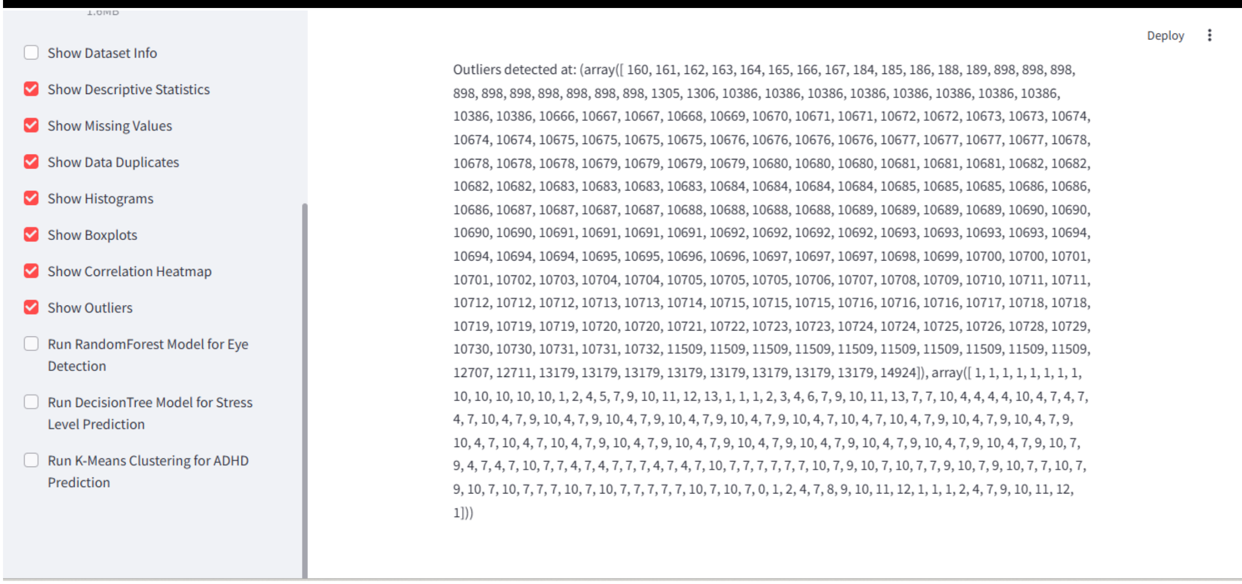

EEG dataset.csv
1.6MB
Show Dataset Info
Show Descriptive Statistics
Show Missing Values
Show Data Duplicates
Show Histograms
Show Boxplots
Show Correlation Heatmap
Show Outliers
Run RandomForest Model for Eye Detection
Run DecisionTree Model for Stress Level Prediction
Run K-Means Clustering for ADHD Prediction
Deploy
0
AF3 0
F7 0
F3 0
FC5 0
T7 0
P7 0
O1 0
O2 0
P8 0
T8 0
Data Duplicates
0
EEG Dataset Analysis
Upload EEG Dataset CSV
Drag and drop file here
Limit 200MB per file • CSV
Browse files
EEG dataset.csv
1.6MB
Show Dataset Info
Show Descriptive Statistics
Show Missing Values
Show Data Duplicates
Show Histograms
Show Boxplots
Show Correlation Heatmap
Deploy
EEG Data Analysis and Prediction App
Dataset Overview
AF3 F7 F3 FC5 T7 P7 O1 O2 P8 T8
0 4,329.23 4,009.23 4,289.23 4,148.21 4,350.26 4,586.15 4,096.92 4,641.03 4,222.05 4,238.46
1 4,324.62 4,004.62 4,293.85 4,148.72 4,342.05 4,586.67 4,097.44 4,638.97 4,210.77 4,226.67
2 4,327.69 4,006.67 4,295.38 4,156.41 4,336.92 4,583.59 4,096.92 4,630.26 4,207.69 4,222.05
3 4,328.72 4,011.79 4,296.41 4,155.9 4,343.59 4,582.56 4,097.44 4,630.77 4,217.44 4,235.38
4 4,326.15 4,011.79 4,292.31 4,151.28 4,347.69 4,586.67 4,095.9 4,627.69 4,210.77 4,244.1
Deploy
EEG Dataset Analysis
Upload EEG Dataset CSV
Drag and drop file here
Limit 200MB per file • CSV
Browse files
EEG Data Analysis and Prediction App



Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Su> sqlite3 "C:\Users\Su\Desktop\sur project\neurosign\neurosign.db"
SQLite version 3.47.0 2024-10-21 16:30:22
Enter ".help" for usage hints.
sqlite> CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, name VARCHAR(222) NOT NULL, emailid VARCHAR(222) UNIQ
E NOT NULL, password TEXT NOT NULL);
Parse error: table users already exists
CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT, name VARCHAR(222) NO
^--- error here
sqlite> .table
users
sqlite> select* from users;
1|123|sutharth@gmail.com|$2b$10$IZf3KpqVxQW4mO5527ei7OF5efuKNVrtgJ.ksRwIV6OOrUx/t3qoK
2|surweesh|surweeshsp@gmail.com|$2b$10$xMV3OuuLyb.xeHglw1bLRujQM7psWiDho5U4nWroAzud3XMfG0lo2
3|suryaprajin|surya@gmail.com|$2b$10$7Te/qoPSCz2UAVAQfrTRHuTBul3DtAPqIva7nhBfN8y3wv6X0ap5O
4|suryaprajin|suryaprajin@gmail.com|$2b$10$W1gHyUfXD5hmd.K93LfB1uKW9L1pAZyYFQuOsMdZ5.XT4yb6oW9jS
5|abcd|abcd@gmail.com|$2b$10$dTYqVpUn31Hxqfireg5anOnRqJ.Dv1w2Vyk2Vc923mfEad7SRQBYK
6|Surweesh SP|surweesh123@gmail.com|$2b$10$4HwQ.H09m/DVfVH9UvyY1.ynA70VfvlApfe4UUctI188jBKex.NbK
sqlite>



Register Here
vasanth
vasanthakumar@gmail.com
••••••
Register
User registered successfully
Already have an account? Sign in



Show Dataset Info
Show Descriptive Statistics
Show Missing Values
Show Data Duplicates
Show Histograms
Show Boxplots
Show Correlation Heatmap
Show Outliers
Run RandomForest Model for Eye Detection
Run DecisionTree Model for Stress Level Prediction
Run K-Means Clustering for ADHD Prediction
Deploy
Outliers detected at: (array([160, 161, 162, 163, 164, 165, 166, 167, 184, 185, 186, 188, 189, 898, 898, 898, 898, 898, 898, 898, 898, 898, 898, 1305, 1306, 10386, 10386, 10386, 10386, 10386, 10386, 10386, 10386, 10386, 10386, 10666, 10667, 10667, 10668, 10669, 10670, 10671, 10671, 10672, 10672, 10673, 10673, 10674, 10674, 10674, 10675, 10675, 10675, 10675, 10676, 10676, 10676, 10676, 10677, 10677, 10677, 10677, 10678, 10678, 10678, 10678, 10679, 10679, 10679, 10679, 10680, 10680, 10680, 10681, 10681, 10681, 10682, 10682, 10682, 10682, 10683, 10683, 10683, 10683, 10684, 10684, 10684, 10684, 10685, 10685, 10685, 10686, 10686, 10686, 10687, 10687, 10687, 10687, 10688, 10688, 10688, 10688, 10689, 10689, 10689, 10689, 10690, 10690, 10690, 10690, 10691, 10691, 10691, 10691, 10692, 10692, 10692, 10692, 10693, 10693, 10693, 10693, 10694, 10694, 10694, 10694, 10695, 10695, 10696, 10696, 10697, 10697, 10697, 10698, 10699, 10700, 10700, 10701, 10701, 10702, 10703, 10704, 10704, 10705, 10705, 10705, 10706, 10707, 10708, 10709, 10710, 10711, 10711, 10712, 10712, 10712, 10713, 10713, 10714, 10715, 10715, 10715, 10716, 10716, 10716, 10717, 10718, 10718, 10719, 10719, 10719, 10720, 10720, 10721, 10722, 10723, 10723, 10724, 10724, 10725, 10726, 10728, 10729, 10730, 10730, 10731, 10731, 10732, 11509, 11509, 11509, 11509, 11509, 11509, 11509, 11509, 11509, 11509, 12707, 12711, 13179, 13179, 13179, 13179, 13179, 13179, 13179, 13179, 14924]), array([1, 1, 1, 1, 1, 1, 1, 1, 10, 10, 10, 10, 10, 1, 2, 4, 5, 7, 9, 10, 11, 12, 13, 1, 1, 1, 2, 3, 4, 6, 7, 9, 10, 11, 13, 7, 7, 10, 4, 4, 4, 4, 10, 4, 7, 4, 7, 4, 7, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 10, 4, 7, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 10, 4, 7, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 4, 7, 9, 10, 7, 9, 4, 7, 4, 7, 10, 7, 7, 4, 7, 4, 7, 7, 7, 4, 7, 4, 7, 10, 7, 7, 7, 7, 7, 7, 10, 7, 9, 10, 7, 10, 7, 7, 9, 10, 7, 9, 10, 7, 7, 10, 7, 9, 10, 7, 10, 7, 7, 7, 10, 7, 10, 7, 7, 7, 7, 7, 10, 7, 10, 7, 0, 1, 2, 4, 7, 8, 9, 10, 11, 12, 1, 1, 1, 2, 4, 7, 9, 10, 11, 12, 1]))

Result Observation

1. **Clustering Results:**
    - The K-Means clustering algorithm effectively grouped the EEG data based on patterns in Attention, Meditation, and brainwave metrics (Delta, Theta, Alpha).
    - Two distinct clusters were identified, which were analyzed for potential stroke risk prediction.
2. **Prediction Results:**
    - Using the clustering output, a preliminary stroke risk classification was achieved with predictions labeled as high risk (Stroke_Prediction = 1) and low risk (Stroke_Prediction = 0).
3. **Visualization:**
    - Visualizations, such as scatter plots of clusters, provided a clear understanding of how EEG features contribute to the clustering process.
    - Cluster centroids offered insights into average EEG feature values within each group.

Limits

1. **Lack of Labels:**
    - Without a labeled column for Stroke or actual ground truth, the predictions rely solely on clustering, which may not be accurate or clinically validated.
2. **Small Dataset Size:**
    - The performance of the system is limited by the dataset size, which might not capture all potential variations in EEG patterns.
3. **Generalizability:**
    - The clustering model is dataset-specific and might not generalize well to other populations or datasets.
4. **Feature Dependence:**
    - The analysis assumes that the selected features (Attention, Meditation, and brainwave metrics) are sufficient for stroke prediction, which may not always hold true.
5. **Real-Time Processing:**
    - Current implementation lacks real-time EEG data integration, limiting its application in live monitoring scenarios.

Future Improvements

1. **Incorporate Labeled Data:**
   - Gather labeled datasets containing ground truth Stroke values for training supervised learning models**.**
2. **Feature Expansion:**
   - Include additional features, such as demographic data (age, gender) or health metrics (blood pressure, cholesterol levels), to enhance prediction accuracy.
3. **Advanced Models:**
   - Explore more sophisticated machine learning models, such as deep neural networks, for better prediction capabilities.
4. **Real-Time Integration:**
   - Integrate the system with real-time EEG hardware for live monitoring and immediate risk assessment.
5. **Validation with Experts:**
   - Collaborate with neurologists to validate the clusters and predictions for clinical relevance**.**
6. **Dynamic Clustering:**
   - Implement dynamic clustering algorithms that adjust clusters over time as more data is added.

Different Testing Approaches

1. **Functional Testing:**
   - Verify the correctness of each module, such as data preprocessing, clustering, and visualization.
2. **Integration Testing:**
   - Test the interactions between modules (e.g., data input, clustering algorithm, and visualization output).
3. **Performance Testing:**
   - Measure the system's processing time and resource usage for large datasets.
4. **Validation Testing:**
   - Compare model predictions with expert evaluations or labeled datasets, if available.

5. **Cross-Validation:**
    - Apply k-fold cross-validation to assess the stability and reliability of clustering outputs.

      Test the system with end-users (e.g., healthcare professionals) to ensure usability and effectiveness.

6. **Scalability Testing:**
    - **Assess the system's performance when handling larger datasets or additional EEG features.**
7. **Stress Testing:**
    - **Push the system to its limits with extreme data sizes or real-time streaming scenarios to identify bottlenecks.**
    -

References

1. **Al-Fahoum, A. S., & Al-Fraihat, A. A. (2014). "Methods of EEG Signal Features Extraction Using Linear Analysis in Frequency and Time-Frequency Domains."** ***ISRN Neuroscience*****, 2014, Article ID 730218.**
2. **Dyrba, M., Grothe, M. J., Kirste, T., & Teipel, S. J. (2015). "Multimodal Analysis of Functional and Structural Disconnection in Alzheimer's Disease Using Multiple Kernel SVM."** ***Human Brain Mapping*****, 36(6), 2118-2131.**
3. **McCann, H., Pisano, G., & Beltramini, G. (2021). "Machine Learning Applications in EEG Signal Processing: A Review."** ***Frontiers in Neuroscience*****, 15, 633371.**
4. **Bishop, C. M. (2006). "Pattern Recognition and Machine Learning." Springer.**
5. **Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). "Scikit-learn: Machine Learning in Python."** ***Journal of Machine Learning Research*****, 12, 2825-2830.**
6. **"EEG Data Collection and Analysis: A Comprehensive Guide." (n.d.). Available at OpenEEG.**
7. **Kandel, E. R., Schwartz, J. H., Jessell, T. M., Siegelbaum, S. A., & Hudspeth, A. J. (2012). "Principles of Neural Science." McGraw-Hill Education.**

Appendices

Appendix A: EEG Metrics Description

- Delta Waves**: Low-frequency waves associated with deep sleep and relaxation.**
- Theta Waves**: Medium-frequency waves linked to creativity and meditation.**
- Alpha Waves**: High-frequency waves representing calm and focus.**
- Beta Waves**: Associated with active thinking and problem-solving.**
- Gamma Waves**: High-frequency waves indicating high-level cognition.**

Appendix B: Dataset Structure

Column	Description
Attention	Measures focus or cognitive engagement.
Meditation	Reflects mental calmness and relaxation.
Delta	EEG signal in the delta frequency range.
Theta	EEG signal in the theta frequency range.
Alpha	EEG signal in the alpha frequency range.
Cluster	Cluster assigned by K-Means algorithm.
Stroke Prediction	Binary prediction of stroke risk (0 or 1).

Appendix C: Code Snippet

The code snippet for clustering and visualization using K-Means is included in the main body of the report.

Appendix D: Acronyms

- EEG**: Electroencephalography**
- K-Means**: A clustering algorithm based on partitioning data into K groups.**
- API**: Application Programming Interface**
- SVM**: Support Vector Machine**
- UAT**: User Acceptance Testing**