

BABEȘ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER
SCIENCE
SPECIALIZATION Computer Science

A Comparison of Different Machine Learning Models for Panic Disorder Detection

Supervisor
Asistent Universitar, Dr. Mihai Andrei

Author
Abrudan Rebeca-Rafela

2024

UNIVERSITATEA BABEȘ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
SPECIALIZAREA Informatică

**Comparație între Diferite Modele de
Învățare Automată pentru Detectarea
Tulburării de Panică**

Conducător științific
Asistent Universitar, Dr. Mihai Andrei

Absolvent
Abrudan Rebeca-Rafela

2024

ABSTRACT

In the recent years, the importance of mental health has gained significant attention, showing the need for effective solutions to diagnose and manage conditions such as panic disorder. A common mental health issue is panic disorder. Panic Disorder is characterized by intense stress, followed by frequent and unexpected panic attacks. Early and accurate detection of panic disorder is a critical step towards effective treatment and management. Currently, the diagnostic of panic disorder happens by talking with a health care provider. AI-driven approaches can help with monitoring and tracking the symptoms and risk factors associated with this disorder. This, alongside real-time data collection can facilitate personalized support for the people in need.

Therefore, the main purpose of this thesis is to create an AI-based application aimed to predict the likelihood of panic disorder in an individual. Using an open-source dataset, the application integrates three distinct machine learning models, Support Vector Machine (SVM), Decision Tree Classifier (DT), and XGBoost Classifier (XGBoost), in order to improve the efficiency of panic disorder detection. These predictions can assist both healthcare providers and patients. In the end, the XGBoost model achieved the highest accuracy and the best performance out of the three models. The accuracy on training set is 99.29% and on testing is 99.11%.

The developed application features a user-friendly interface, allowing users to select models and input data for predictions. This system facilitates early intervention and management of panic disorder, offering an accessible solution for mental health care.

Overall, this research demonstrates the potential of AI to revolutionize the detection and management of panic disorder, providing timely support and improving the quality of life for individuals affected by this condition. Future work will focus on further improving model performance, expanding the dataset, improving the user interface, and integrating the application with clinical tools to better support mental health diagnostics.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Objectives	1
1.3	Thesis structure	2
1.4	Personal Contribution	2
1.5	Declaration of Generative AI and AI-Assisted technologies in the writing process	3
2	Related work	4
2.1	Overview of Panic Disorder	4
2.1.1	Risk Factors and Diagnostic Criteria	4
2.1.2	Conventional Treatment and Diagnostic	5
2.2	Treatment and Diagnostic Optimization	5
3	Theoretical foundations	9
3.1	Introduction to Machine Learning	9
3.1.1	Supervised Learning	10
3.1.2	Machine Learning Algorithms	11
3.1.3	Ethical Considerations	14
3.2	Technologies	15
3.2.1	Frontend technologies	16
3.2.2	Backend technologies	21
3.2.3	Machine Learning technologies	22
4	AI models	25
4.1	Dataset	25
4.2	Data Preprocessing	26
4.3	Cross validation	28
4.4	Model Training	30
4.5	Model Evaluation	31

5	Application	35
5.1	Architecture	35
5.1.1	Use Case Diagram	35
5.1.2	Sequence Diagrams	37
5.2	Implementation	39
5.2.1	Frontend	41
5.2.2	Backend	42
5.2.3	Database	43
5.3	UX/UI	43
5.4	Testing	47
6	Conclusions and Future Work	49
	Bibliography	50

Chapter 1

Introduction

1.1 Motivation

Mental health is a big topic nowadays, and panic disorder is one of the most common issues people face. Panic disorder involves sudden and intense feelings of fear, known as panic attacks, which can seriously affect a person's daily life. Traditionally, diagnosing panic disorder involves consultations with healthcare providers, which can delay getting the right help. Luckily, with new technology, especially artificial intelligence (AI), there is presented a chance to change how panic disorder is traditionally diagnosed and managed. AI can keep track of symptoms and risk factors all the time, giving quick insights and personalized help to those who need it. This can make a significant difference in how mental health care is handled, making it faster and more effective.

1.2 Objectives

The primary objective of this thesis is to explore AI-driven approaches in order to improve the detection and diagnosis of panic disorder. This is done by developing a web application that can predict the likelihood of panic disorder in individuals, by implementing three machine learning models. They are XGBoost Classifier (XGBoost), Support Vector Machine (SVM) and Decision Tree Classifier (DTC). The application processes and encodes relevant features from a detailed dataset and it provides accurate predictions that can help both healthcare providers and patients. For a better and easier understanding, in the web application we will be able to add new data and select one of the three models in order to test the possibility of panic disorder.

1.3 Thesis structure

The paper is divided into six chapters, each chapter providing information related to the Panic Disorder Detection Application.

Chapter 2: Related Work - In this chapter it is presented a brief overview of panic disorder and the importance of detecting it. It continues with the diagnostic criteria, the way medical consultations happen at the moment and the benefits machine learning could bring. Furthermore, this chapter reviews some related works and studies regarding panic disorder detection and using machine learning for its detection.

Chapter 3: Theoretical foundations - This chapter sets the foundational concepts that need to be understood in order to create the application. Two sections are taken into account. Firstly, the Supervised Machine Learning and the algorithms that will be used (Decision Tree, SVM and xgboost). Also the ethical considerations when working with AI in healthcare and possible issues related to biased algorithms. Secondly, the technologies that will be used and the reason for selecting those specific technologies.

Chapter 4: AI Models - This chapter shows the dataset and its fields. Then presents the preprocessing and cleaning of the data. Furthermore, the tuning of the AI models and the results are compared in this chapter.

Chapter 5: Application - This chapter describes the user-friendly implementation of the proposed solution for the Panic Disorder Detection Application. It also covers the architecture of the application, showing its flow and how each element interacts with the other.

Chapter 6: Conclusion - An overview of the thesis and the potential impact of it in the mental health field is presented in the final chapter. Moreover, areas for future developments to both the application and the detectors are suggested.

1.4 Personal Contribution

In this thesis, I was responsible for selecting and preprocessing the dataset to ensure it was suitable for the application. I implemented and fine-tuned three models: SVM, Decision Tree, and XGBoost, and compared their results. Additionally, I integrated the models into the web application, focusing on a user-friendly interface that allows easy data input and prediction. Throughout the process, I documented my work and tested the application's functionalities. This thesis shows the potential of AI in healthcare and sets a foundation for future advancements in the field.

1.5 Declaration of Generative AI and AI-Assisted technologies in the writing process

During the preparation of this thesis, the author used CHAT-Gpt in order to improve the flow in her ideas and to check the spelling and grammar. After using the tool/service, the author reviewed and edited the content as needed and takes full responsibility of the content of the thesis.

Chapter 2

Related work

In this chapter, we will go through the basics of panic disorder in order to gain a deeper understanding of its causes. We will also see examples of related work and the use of AI in healthcare. This will help us develop a more effective and accurate detection model for panic disorders.

2.1 Overview of Panic Disorder

Panic Disorder is characterized by intense stress, followed by frequent and unexpected panic attacks. Panic attacks can include symptoms such as rapid heart rate, trembling, sweating, difficulty breathing and chest pain. These symptoms are similar to the ones of heart attack.

Due to this unpredictability, people with panic disorder often worry about the moment they could have another attack. This can result in the development of phobias, such as agoraphobia, and exaggerate anxiety, making everyday activities extremely challenging for those suffering from this condition. Furthermore, this leads to individuals avoiding places or situations where previous panic attack happened, causing social exclusion and other mental health issues.

2.1.1 Risk Factors and Diagnostic Criteria

Like for the most mental health conditions, the specific cause of panic disorder remains not fully understood yet. But it can be linked to some primary risk factors. These include family history of disorder, exposure to traumatic events, significant life stressors and environmental factors, or an imbalance of neurotransmitter activity in the brain. Research have also found that panic attacks resemble "false alarms", where the body's survival mechanisms are triggered too frequently or too strongly [Nat22].

In order to diagnose any psychiatric illness, doctors from European Union follow the protocols from the 10th Revision of the International Classification of Diseases, (ICD-10). Another common reference source in academic researches and used in USA, is Diagnostic and Statistical Manual of Mental Disorders (DMS). In both of them, the criteria for Panic Disorder without Agoraphobia is defined similarly.

According to ICD-10 (F41.0), the diagnosis of panic disorder is based on the presence of recurrent and unexpected panic attack, characterized by sudden episodes of severe fear or distress accompanied by various psychological and physical symptoms. Some of the symptoms can include heart palpitations, sweating, chest pain, difficulty breathing, abdominal distress, feelings of depersonalization, fear of losing control, and fear of dying[ICD92]. In DMS, it is diagnosed as panic disorder after at least one month of persistent concern related to having additional attacks or the consequences of the attacks [Cri16].

2.1.2 Conventional Treatment and Diagnostic

Early and accurate detection of panic disorder is a critical step towards effective treatment and management. Currently, the diagnostic of panic disorder happens by talking with a health care provider. He will conduct a physical examination, ensuring there are no unrelated physical problems causing the symptoms. After that, a mental health professional will give a diagnosis based on established diagnostic criteria. The treatment usually consists of psychotherapy, medication or both. The psychotherapy used is Cognitive behavioral therapy (CBT) which consists of learning to think and react differently to the feelings that happen during or after a panic attack.

2.2 Treatment and Diagnostic Optimization

There has been an increasing interest in using AI in Healthcare and other domains as well. As an outcome, there are many relevant studies about detection and prevention of many disorders, in our case Panic Disorder. AI-driven approaches can help with monitoring and tracking the symptoms and risk factors associated with this disorder. These approaches can analyze static data, such as age, family history, medical history, current stressors, and so on, then identify the patterns of panic disorder. This, alongside real-time data collection can facilitate personalized support for the people in need. In the rest of the section, there will be presented 3 researches related to the use of AI for Panic Disorder.

Prediction Using Wearable Devices and Machine Learning

One of the main symptoms of panic attacks is increased heart rate. Thus, the most promising tool to predict them is using wearable devices that can monitor the heart rate, sleep patterns and activity level. After the data is collected, it can be analyzed using different machine learning methods. But there are only a few datasets available and there is a high risk of bias among them.

However, a study by [Cha22], where 59 participants with Panic Disorder were enrolled, and continuously monitored for a year, demonstrates that its prediction is indeed possible. The study aimed to find a direct relationship between panic attacks and different factors, such as heart rate, air pollution and other environmental conditions. Initially, it was found that on its own, the heart rate data wasn't enough to accurately predict panic attacks. So the dataset used for training the models included not only physiological data (heart rate, sleep patterns) but also contained environmental data (air quality index) and questionnaire data (anxiety and depression scores).

The classifiers used in the study were Extreme Gradient Boosting (XGBoost), Linear Discriminant Analysis (LDA), Adaptive Boosting (AdaBoost) and Regularized Greedy Forests (RGFs). The analysis was performed using Scikit-learn library version 0.23.1 and Python 3.6.10. For predicting panic attacks within a 7-day window, the models showed varying accuracy, ranging from 67.4% to 81.3%, depending on the model used. As it can be seen in Figure 2.1. This indicates that while single-source data like heart rate might not be sufficient, combining multiple data sources improves the performance of the AI.

The study identified a number of important variables that influence the prediction of panic attacks. Physiological measures like resting heart rate, average heart rate, and length of sleep were important components. A significant role had also the psychological factors, which were evaluated using tools such as the Panic Disorder Severity Scale Self-Report, the State-Trait Anxiety Inventory (STAI), the Beck Depression Inventory (BDI), and the Beck Anxiety Inventory (BAI). The occurrence of panic attacks was found to connect with environmental data, specifically the Air Quality Index (AQI), indicating a major contribution from external stressors to the initiation of these episodes.

Despite the challenges presented by the limited availability and potential biases in medical datasets, considerable progress has been made in understanding and predicting panic disorder using AI. This study represents an important step forward, demonstrating that a large, multimodal dataset can significantly improve the accuracy of predictive models. The results suggest that machine learning models can effectively use this diverse data to provide accurate predictions, which can be in-

strumental in early intervention and personalized treatment for individuals with Panic Disorder.

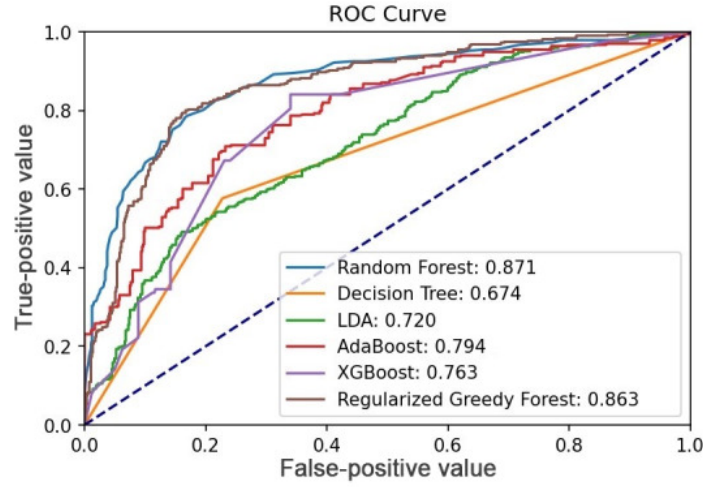


Figure 2.1: Existent AI models results [Cha22]

Prediction using Machine Learning and Real-Time Biometric and Spatiotemporal Data

In another study [LKHT22], participants were continuously monitored using wearable devices and smartphones. This research showed the potential combining biometric and spatiotemporal data analysis might have in order to predict panic attacks.

Real-time analysis of the gathered dataset, which included location data, activity levels, speed, steps, gender, age, weight and heart rate, was done. By capturing a subject's physical and emotional state, this extensive data collection attempts to provide machine learning algorithms a strong base on analyzing and predicting panic episodes.

Using the dataset, various machine learning models were trained. Some of them included Gaussian Support Vector Machine (SVM) and Long Short-Term Memory (LSTM) neural network. The Gaussian SVM achieved an accuracy of 94.5%, slightly outperforming the LSTM model, which had an accuracy of 94%. Despite employing deep learning techniques like a Deep Neural Network (DNN) and the LSTM network, the Gaussian SVM proved to be the most effective model.

A significant contribution of this study is the introduction of the heart rate moving average deviation (HRMAD) feature. This feature measures the deviation of current heart rate readings from a moving average calculated over specific time intervals (10, 30, and 60 seconds). Incorporating HRMAD into the models substantially enhanced their classification accuracy, demonstrating its importance in panic

detection.

This study demonstrated the value of a multimodal approach by showing that accurate prediction of panic episodes could not be achieved using heart rate alone. The integration of biometric data with spatiotemporal context significantly improved the prediction models' performance.

Prediction Model Based on Machine Learning Algorithm

The research paper [MKC22] , focuses on addressing the severity and impact of panic disorder. Studies on the prevalence of panic disorder indicate that it is a major mental health issue, with around 5% of the global population meeting the clinical diagnosis criteria, and about 10% having subclinical symptoms. Additionally, 9% of teenagers have clinical symptoms of panic disorder, such as palpitations, tremors, panic episodes, and chest pain. Panic disorder can have serious repercussions for sufferers and their families if treatment is not received. The authors highlight the importance of early prediction of panic disorder and propose machine learning algorithms for this.

The study analyzes the machine learning models, in order to determine how accurate and effective they are at predicting panic disorder. Among the models tested, the artificial neural network (ANN) model stands out, achieving the highest Area Under the Curve (AUC) score of 0.8255. This suggests that out of all the models examined, the ANN model had the highest accuracy. Moreover, all of the models also showed a running duration of under a second, which qualifies them for real-time prediction applications.

Despite the challenges presented by most of the medical datasets available, there has been made significant progress in understanding the causes and the factors of panic disorder and in using AI for detecting it. The results from these studies indicate the potential of AI in accurately predicting panic disorder and panic attacks, when a comprehensive set of data is used.

Chapter 3

Theoretical foundations

A possible approach in detecting panic disorder is through Machine Learning. In this chapter we will explore fundamental concepts in machine learning, including different models such as decision trees, support vector machines (SVM) and XG-Boost Classifier.

3.1 Introduction to Machine Learning

Artificial Intelligence includes various branches, one of them being machine learning. With the help of machine learning, the program can autonomously improve by learning and experience, using training data. In conventional programming, the developer creates a set of instructions for the computer to convert the input data into intended output. Contrarily, machine learning is an automated process that allows the algorithm to tackle the problems with minimal or even no human intervention and to make decisions based on previous observations. Based on the data provided, the algorithm can make predictions using insights learned from datasets.

Machine learning, credited to Arthur Samuel in 1959, is defined as the field of study that enables computers to learn without explicit programming. Samuel proposed that computers could be taught to understand the world and perform tasks independently, laying the foundation for machine learning as a critical aspect of artificial intelligence.

Machine learning algorithms are commonly applied to tasks such as prediction and classification. The training phase involves input data that may be either labeled or unlabeled. Labeled data includes known correct answers before the training begins, while unlabeled data does not provide the correct answers. Then, the algorithm analyzes the data in order to identify patterns and generate predictions.

This error function compares the algorithm's predictions to the actual known outcomes and evaluates the accuracy of the predictions. The algorithm adjusts its

weights, to reduce the error margin, if the predictions are inaccurate. This process of evaluation and adjustment is repeated iteratively until the algorithm achieves the desired accuracy. Over time, as the algorithm processes more data, it refines its predictions and enhances its performance. However, machine learning also faces some challenges such as the lack of quantity and quality in training data.

3.1.1 Supervised Learning

Aurélien Géron [Ger19] categorizes machine learning systems into three classifications, based on these factors:

- **Type of Supervision:** This means whether the algorithms are trained with human guidance or not. The learning can be split into supervised, unsupervised, semi-supervised, and reinforcement.
- **Learning Method:** This factor determines if the system can learn incrementally in real time or if it process data in large groups. The learning can be split into online and batch.
- **Learning Approach:** This involves whether the system only compares new data points to known data points or identifies patterns in the training data to create a predictive model. The learning can be split into instance-based and model-based.

In this subsection we will focus on supervised learning, which uses labeled input data for training the algorithms.

Two of the most frequent supervised tasks are classification and regression. The goal of supervised learning is usually to find the best function $f(x)$, using input data x . Therefore, the records from the dataset become vectors of feature $x(i)$, with $y(i)$ their target label. The parameters in the function $f(x)=ax+b$ are adjusted such that the error between the predicted values and the actual values is minimized.

Classification involves assigning an item (x) to a specific category based on pre-defined labels (y). It uses its features to approximate a mapping function. Algorithms like decision trees, support vector machines, and so on, are used for tasks such as diagnosing patient conditions and recommending treatments.

On the other hand, regression is focused on predicting a continuous output variable. Instead of classifying items into categories, regression models aim to estimate a value. The performance of regression models is evaluated through metrics like the coefficient of determination and root mean square error. Common regression algorithms include linear regression and support vector machines, which are used for applications such as estimating the future risk of patient mortality using clinical data.

3.1.2 Machine Learning Algorithms

SVM

As it is presented in 3.1.1 Support Vector Machines (SVMs) are versatile and can be used for both classification and regression tasks. In the case of linear and non-linear classification, SVMs work by finding the optimal hyperplane that separates data points of different classes with the maximum margin and that predicts continuous outputs. For regression, SVR uses a similar principle to SVM classification but adjusts it to predict continuous values rather than discrete classes.

Some of the benefits of SVMs include:

- **Versatility:** Can be used for both linear and nonlinear data.
- **Effective in high-dimensional spaces:** Particularly useful when the number of dimensions exceeds the number of samples.

Linear SVM

A good SVM model finds a boundary that separates the classes and stays as far away from the nearest data points as possible. This technique is called large margin classification, where the "street" between classes is as wide as it can be. The data points that are on the edge of this street are called support vectors, and they help define the boundary. This can be seen in Figure 3.1 However, SVMs are sensitive to the scale of the features, which can affect the orientation and position of the decision boundary. Therefore, feature scaling is essential for SVM performance.

However, in real-world training sets, data is often not perfectly separable. In consequence, hard margin classification, with a strict boundary, can be impractical. Soft margin classification tries to find a balance between having a wide margin and minimizing the errors. In Scikit-learn, this balance can be controlled using a C hyperparameter. A lower C means a wider street and more margin violations and a higher C is the opposite. As a consequence, reducing the C hyperparameter can help with the overfitting problem.

To implement a linear SVM in Scikit-Learn, you typically scale the features and train the model using the LinearSVC class, setting the appropriate hyperparameters like C and loss function. This ensures the model can effectively separate classes and generalize well to new data.

Nonlinear SVM

Linear SVM classifiers work well for many datasets, but for those that are not linearly separable, adding features like polynomial ones can make them separable. For a lower degree, this polynomial can visibly slow the model. The kernel trick in SVMs allows for handling complex datasets without explicitly adding features, by transforming the data into a higher-dimensional space. Another feature that can be

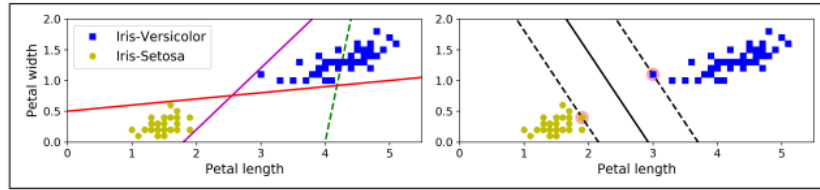


Figure 3.1: Stack overflow trends [Ger19]

useful is the similarity method. The Gaussian Radial Basis Function (RBF) kernel is particularly effective, as it can handle non-linear data efficiently. It calculates the similarity between data points and landmarks using a Gaussian function, allowing the model to create a flexible decision boundary.

SVM Regression

In SVM Regression, the objective is reversed: the algorithm aims to fit as many instances as possible within a margin (controlled by the hyperparameter), while limiting margin violations. For linear SVM Regression, Scikit-Learn's LinearSVR class can be used, which produces models insensitive to additional training instances within the margin. Nonlinear regression tasks can be tackled using a kernelized SVM model, such as the SVR class in Scikit-Learn, which supports the kernel trick and can apply polynomial kernels for more complex datasets.

Mathematical Formula

- **Hyperplane Equation:** $w \cdot x + b = 0$
- **Objective:** Minimize $\frac{1}{2} \|w\|^2$ subject to $y_i(w \cdot x_i + b) \geq 1$

DT

Similar to SVM, Decision Trees are capable of performing both classification and regression. They are versatile algorithms and can fit complex datasets. They are also foundational to more advanced models like Random Forests. When training Decision Trees, the process involves splitting the dataset into subsets based on feature values, aiming to create the purest possible subsets at each node. This process is known as the CART (Classification and Regression Trees) algorithm, a greedy algorithm, which Scikit-Learn uses. For instance, in the context of classifying iris flowers, the tree asks a series of yes/no questions about the features (like petal length and width) to make a prediction.

Some of the benefits of DTs include:

- **Easy to interpret:** Their decision-making process is straightforward and visualizable.

- **No need for feature scaling:** Unlike other algorithms, decision trees do not require feature scaling or centering, making them straightforward to implement. They require very little data preprocessing.

Drawbacks of DT:

- **Prone to overfitting:** Decision trees can overfit to the training data. To prevent this, it can be mitigated by parameters like 'max_depth', 'min_samples_split', and 'min_samples_leaf'.
- **Sensitivity to data variations:** Small changes in data can result in a completely different tree structure. They are sensitive to the rotation of the training data and to small variations in the training data. Techniques like Principal Component Analysis help mitigate this sensitivity.

For regression tasks, DT can be trained using Scikit-Learn's 'DecisionTreeRegressor' class and for classification tasks 'DecisionTreeClassifier'. The CART algorithm works similarly for regression as it does for classification. The primary difference is that, for regression, it tries to split the dataset in a way that minimizes the MSE rather than impurity. To prevent overfitting, regression trees can be regularized with the help of parameters like 'min_samples_leaf'.

Mathematical Formula

- **Gini Impurity for Classification:** $Gini = 1 - \sum_{i=1}^n p_i^2$
- **Mean Squared Error for Regression:** $MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$

XGBOOST

Boosting is an ensemble technique that builds a series of predictive models sequentially. Each new model focuses on correcting the errors made by the previous ones. This iterative process turns weak learners into strong learners, effectively reducing bias and variance.

Gradient boosting is a specific type of boosting introduced by Jerome Friedman in 2001. It builds models sequentially, with each new model trying to minimize the loss function of the previous one. This method often uses decision trees as the base models, where each tree focuses on the errors made by the previous ones. Extreme gradient boosting (XGBoost) includes regularization techniques and uses multithread processing for a better execution speed and model performance.

The data preparation steps highlighted by [Bro18] are label encoding the string values, one hot encoding categorical data and handling missing data by minimizing

the loss function. Additionally, The next step after cleaning the data and training the model is using the right technique to evaluate the model. Some of these include k-fold cross-validation, stratified cross-validation and train/test split data.

These key features of XGBoost present its benefits:

- **Handles missing data:** XGBoost can handle real-world data with missing values without requiring significant preprocessing.
- **Parallel processing:** XGBoost has built-in support for parallel processing, making it possible to train models on large datasets in a reasonable amount of time [Wad20].

However, a drawback can be its complexity. XGBoost is more complex to tune compared to simpler models like decision trees or SVMs.

Mathematical Formula:

- **Objective:** Minimize a regularized loss function combining prediction accuracy and model complexity.

$$L(\Theta) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

where $\Omega(f) = \gamma T + \frac{1}{2} \lambda ||w||^2$

3.1.3 Ethical Considerations

As machine learning models are increasingly adopted across various sectors, healthcare being one of them, it is important to address their ethical challenges. One of the primary applications of machine learning in healthcare revolves around patient diagnosis and treatment. Therefore, the integration of AI in healthcare raises ethical questions about how data is used, patient privacy, and the impact of predictive models. It's essential that AI systems are transparent, unbiased, and follow regulations to be trusted and effective. The ethics of AI became a part of researcher's discussions since 1985.

In "Machine Learning and AI for Healthcare" [Pan19], Arjun Panesar highlights some of the ethical concerns regarding AI's integration into healthcare. This section presents some of these concerns.

Bias

Bias is one of the main concerns. Machine bias occurs when models reflect the biases of their creators or the data they are trained on. This can lead to unfair or discriminatory outcomes, especially for the already marginalized population.

The data machine learning models use to train in the most important factor. Therefore intelligence bias appears due to human bias regarding gender, disabilities, race, social positions and so on. However, there is no guarantee that an unbiased AI model won't learn the same biases humans do.

Addressing bias involves acknowledging its existence and implementing measures like data normalization, resampling, and model calibration. Ensuring diversity in the workplace, including AI developers and researchers can also help reduce bias in AI systems. However, since bias is mainly related to societal issues, it can be challenging to completely avoid it.

Data governance

Data is an important asset for all companies. Data ownership leads to questions about who can access the data and what it can be used for and who is accountable for any risks involving this data. It is crucial to ensure only authorized services and organizations have access to it. Data-driven quality improvement includes encrypting and anonymizing data by removing identifiable features, such as name. However, this may not fully guarantee privacy, as data can be de-anonymized. Therefore, AI systems must comply with regulations like Health Insurance Portability and Accountability Act (HIPAA) to secure data.

Informed consent

Patients should be adequately informed about how AI systems will be used in their care, including potential risks and benefits, to make informed decisions about their treatment. Patients should be able to decide whether to share their data or not, this is the freedom of choice concept.

In conclusion, while machine learning holds great promise for improving healthcare outcomes, it is essential to approach its implementation with caution, ensuring that ethical considerations are taken into consideration. Despite the efficiency of AI, human oversight remains crucial. Healthcare professionals should validate AI-driven recommendations to ensure accuracy and appropriateness in patient care.

3.2 Technologies

This section outlines the technologies used in the development of the Panic Disorder Detection Application. Furthermore, it highlights their relevance and the purpose for selecting them by assessing some of their advantages.

3.2.1 Frontend technologies

Using the right web framework is important as it can significantly reduce the development time by providing pre-made components and templates, allowing developers to focus on unique features. Frameworks can also enhance code consistency and maintainability, by enforcing standardized design practices. Another benefit of using a web framework are the built-in tools and components that support responsive design, ensuring applications work seamlessly across various devices and screen sizes. Additionally, popular frameworks tend to have active community support which is valuable for solving problems quickly and efficiently. Overall, the right web framework not only accelerates the development process but also contributes to the creation of robust, scalable, and secure web applications.

According to a study by [SK23], which analyzed Github data to see the number of the most recent created repository and the data from stack Overflow trends some of the most popular frameworks are JavaScript based. These include React, Angular, Vue, as it can be seen in Figure 3.2.

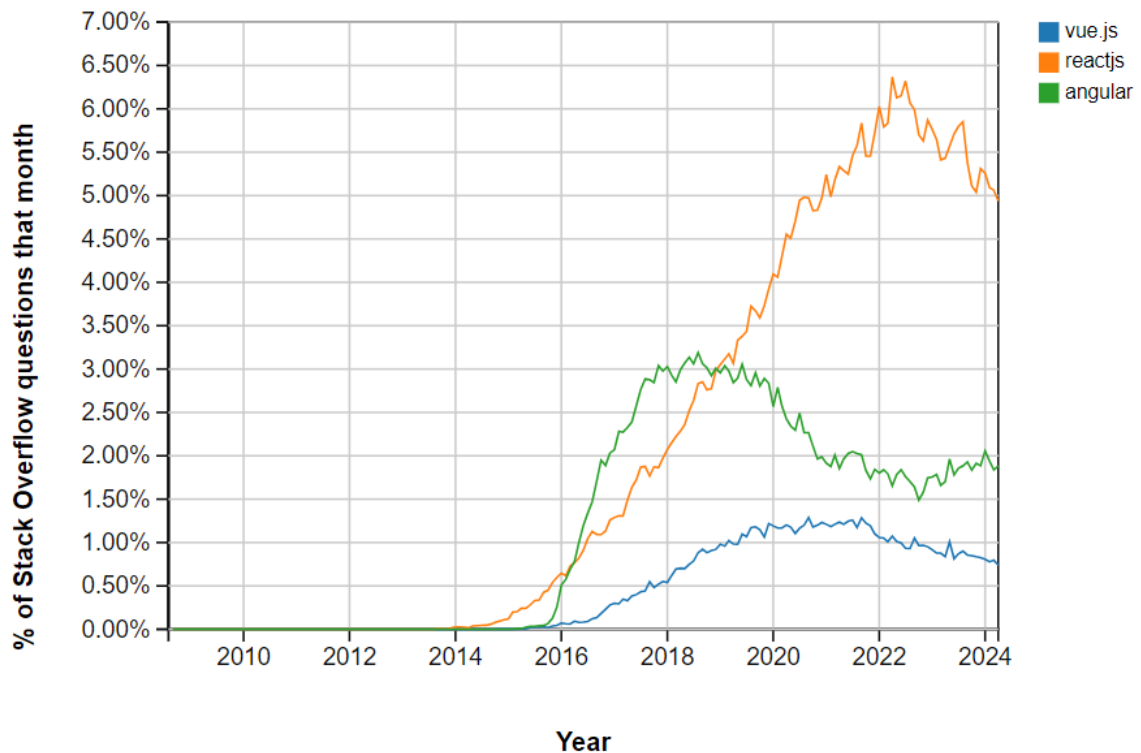


Figure 3.2: Stack overflow trends [Ove]

In order to compare the three frameworks, we will use the analysis made by Rishi Vyas [Vya22]. The main parts of the comparison will be the architecture involved, some of their main features, the performance, the data binding and the code syntax. The performance will be split into startup performance, which measures the loading

speed and the time the user can interact with the application. The most important factor for startup performance is the code bundle size. Secondly, we will compare the runtime performance, which signifies how fast the app updates the UI based on the user input. For this, the most significant factor is how the internals handle the Document Object Model (DOM). Data binding is the mechanism that synchronizes the application's user interface (UI) with its data models. Through data binding, users can interact with website elements directly in the browser.

Vue

Vue is a progressive front-end framework for creating efficient and complex single-page web applications. Developed by Evan You and released in 2014, Vue is known for its ease of integration with existing projects and also its simplicity. It aims to be incrementally adoptable, meaning developers can use as much or as little of the framework as they need.

Vue.js has excellent performance due to its lightweight virtual DOM implementation and efficient reactivity system. It supports Ahead-of-Time (AOT) compilation, which reduces file sizes and improves page load speeds. Vue's use of a virtual DOM ensures faster performance compared to frameworks that use a real DOM, as it minimizes the number of DOM manipulations.

Vue follows the Model-View-ViewModel (MVVM) architecture, which can facilitate modifying and upgrading personal views. This design pattern separates the application into three interconnected components:

- **Model:** Manages the business and data logic.
- **View:** Handles the display and UI logic.
- **ViewModel:** Works with data relevant to the View, linking Models and Views together.

Here are a few features of Vue:

- **Functionality:** The framework's modular approach allows developers to pick and choose the libraries they need, ensuring flexibility and customization. It has a built-in support for routing, but the form validation and HTTP are not included
- **Performance:** Vue's startup performance is impressive due to its smaller bundle size. For larger applications, the size of the code bundles is similar across all frameworks when optimization techniques like lazy loading are implemented. Regarding runtime performance, Vue's use of the virtual DOM (VDOM) ensures faster updates and efficient UI rendering.

- **Data Binding:** It provides reactive two-way data binding between model data and interface elements. This automatically chooses the right update method for an element based on its input type.
- **Code syntax:** Vue uses regular JavaScript, with the option to use TypeScript for more structured code and advanced features such as static typing. Vue's Single File Components structure separates CSS, HTML, and JavaScript, making it easier for developers to manage and understand the code. Testing is facilitated by tools like Jest or Mocha.

In summary, Vue.js is a lightweight and flexible framework that excels in performance and simplicity, which makes it ideal for small to medium-sized projects, while still being capable of handling larger projects with its advanced features and comprehensive tooling.

Angular

Angular is a platform and a front-end framework for creating efficient and complex single-page web applications. Developed by Google, Angular provides a set of tools to build dynamic user interfaces, allowing page content to update smoothly without reloading the entire page. It is often referred to as Angular 2+ and it is a complete rewrite from AngularJS.

It has good performance due to internal optimizations and the use of change detection mechanisms. Furthermore, it supports Ahead-of-Time (AOT) compilation, just like Vue. Although Angular's use of a real DOM can sometimes result in slower performance compared to frameworks that use a virtual DOM, its optimizations compensate for this.

Angular follows the Model-View-Controller (MVC) architecture, which is similar to MVVM, but focuses more on data binding between the View and the Controller:

- **Model:** Manages the business and data logic.
- **View:** Handles the display and UI logic.
- **Controller:** Acts as an intermediary between the Model and View, processing user input and updating the Model.

Here are a few features of Angular:

- **Functionality:** Angular CLI (Command Line Interface) facilitates the creation, testing, and building of Angular applications. It has a comprehensive framework with built-in support for form validation, HTTP requests, routing, and

state management, and a strong focus on reusable components and consistent updates.

- **Performance:** Its startup performance for the small application can be slightly bigger than React and Vue apps. However for larger ones, the size of the code bundles is equally. Regarding runtime performance, Angular's use of the real DOM, instead of a virtual one, can result in slower updates, for large applications.
- **Data Binding:** It provides strong two-way data binding between model data and interface elements. This allows for automatic updates to the user interface when data changes and vice versa, making the development straightforward. On the other hand, this can impact performance due to constant synchronization.
- **Code syntax:** It uses TypeScript, a superset of JavaScript, enabling developers to write more structured code and take advantage of advanced features such as static typing. Angular handles DOM interactions behind the scenes, so developers never have to directly manipulate the DOM, and testing is facilitated by Jasmine.

Angular is great for large-scale enterprise projects because it has lots of tools and features. Its strong typing, easy form handling, and dependency injection make it perfect for complex apps with significant business rules.

React

React is a free and open-source JavaScript library created by Facebook engineers, designed to help in building dynamic and interactive user interfaces (UI) for web and mobile applications. React introduced a new way of thinking about web development with its component-based architecture and the use of a virtual DOM.

It supports server-side rendering (SSR) through frameworks like Next.js, which improves both performance and SEO. React's use of a virtual DOM ensures faster performance compared to frameworks that use a real DOM, as it minimizes the number of DOM manipulations.

React is well-known for its ability to help developers create high-performance applications due to its component-based architecture. This means you can build small, manageable components that can be combined to form more complex interfaces, improving the structure and organization of your code and also promoting the reusability of the code.

Here are a few features of Angular:

- **Functionality:** React offers a wide variety of libraries and tools that expand its capabilities. React Router is used for handling navigation and routing, while state management libraries effectively manage application state. Due to React's versatility, developers may select the frameworks and tools that best suit their requirements, ensuring a customized development experience.
- **Performance:** React's startup performance is outstanding due to its efficient bundling and tree-shaking techniques that minimize the size of the code bundle. These methods optimize performance for larger applications by ensuring that only the code that is needed is loaded. Regarding runtime performance, React's use of the virtual DOM guarantee rapid UI component rendering and updates in terms of runtime performance.
- **Data Binding:** React uses one-way data binding, which enhances performance by ensuring that data flows in a single direction. This makes it easier to debug and track data changes, as updates to the UI are predictable and controlled by the state. This data flow is made possible by React's state and props mechanisms, which guarantee that data reaches components via a clearly defined interface.
- **Code syntax:** React uses JavaScript and JSX, a syntax extension that allows writing HTML within JavaScript. By combining the best features of both languages, this method makes the development process flexible and intuitive. JSX is transpiled to JavaScript during the build process, ensuring compatibility with all browsers. Tools like Jest, which come pre-installed in all React libraries and require very little configuration, make testing with React easier.

In summary, React is a flexible and efficient library that excels at creating dynamic user interfaces. Its component-based architecture, virtual DOM and rich ecosystem, makes it ideal for creating both single-page applications and complex user interfaces. React can adapt to a wide range of project requirements because to its ability to select from and integrate different libraries and technologies

The frontend of the Panic Disorder Detection application is developed using React and TypeScript because React offers high performance with its virtual DOM and component-based architecture, and TypeScript provides strong typing that helps catch errors early, making the development process smoother and more reliable. Furthermore, my familiarity with React allows me to build the application more efficiently and to create a user-friendly and reliable interface.

3.2.2 Backend technologies

The backend framework in Machine Learning models handle both the server-side logic and manages interactions with the proper databases and also the execution environment for the trained models. Choosing a right framework is essential for the development cycle, performance and scalability of the project. Usually, python-based frameworks are popular for machine learning projects due to Python's rich set of libraries and tools for data science and AI. Two of the most commonly used are Django and Flask. However, there are suitable frameworks for Java too, such as Spring Boot, or for JavaScript, such as Nodejs.

Django

Django is a high-level Python framework designed help with the quick development and clean design of web applications. The main principle of Django is DRY (don't repeat yourself), which emphasizes reusability of components, therefore less code and rapid development. It also has a large active community which provides support in finding solutions.

Here are a few features of Django:

- **Usage:** Django is often preferred for more complex applications. It offers built-in features such as Object-Relational Mapping (ORM), which allows developers to directly modify the databases using Python code.
- **Security:** Security is important in managing a web application, especially when dealing with sensitive data. Django offers protection against clickjacking, SQL injection, cross-site request forgery (CSRF), cross-site scripting (XSS). Its user authentication system is comprehensive and can be extended to meet custom security requirements.
- **Scalability:** Django is designed to handle websites and applications with a lot of traffic. It can scale to accommodate growing user demands because of its architecture, which allows for the use of multiple caching mechanisms and database optimization strategies. This makes it suitable for a ML project, which usually grows quickly as more data is being processed.

Django is a strong and flexible framework that's great for adding machine learning models to web applications. It has a lot of built-in features, good security, and a big community for support, which makes it a great choice for developing advanced and fast web applications.

Flask

Flask is a lightweight Python web framework designed to be simple yet powerful, allowing developers to build web applications easily. It is based on Werkzeug and Jinja2 and it follows a minimalistic approach, providing the essentials and leaving the rest up to the developer. Flask's flexibility makes it a popular choice for many developers.

Here are a few features of Flask:

- **Usage:** Flask is often preferred for smaller applications or when the developer wants more control over the components they use. It does not come with built-in ORM, but it allows for easy integration with SQLAlchemy or other ORM libraries, giving developers the freedom to choose their tools.
- **Security:** Although Flask does not include built-in security features like Django, it provides extensions that offer protection against common security threats such as SQL injection, cross-site request forgery (CSRF), cross-site scripting (XSS), and so on. Developers can customize their security implementations to meet specific needs.
- **Scalability:** Flask's simple and modular design allows for easy scaling. It can handle growing user demands by integrating with tools like Redis for caching and Celery for background task processing. This flexibility makes Flask suitable for machine learning projects that might expand over time.

The backend of the application is built with Python using Flask, with data management handled by SQLAlchemy connected to a SQLite3 database. Flask offers a flexible framework for creating web APIs, while SQLAlchemy simplifies database interactions with its ORM capabilities. SQLite3 is a lightweight, serverless database that is perfect for development and smaller applications, ensuring efficient data storage and retrieval. This combination ensures seamless integration, efficient data handling, and easy API management.

3.2.3 Machine Learning technologies

Machine learning libraries are available for a wide range of uses, including computer vision, classification, prediction, and AI-powered tools. Therefore, it is important to select the appropriate tool for building, training, and deploying the machine learning models. These frameworks simplify the implementation of complex algorithms and enable the development of scalable and efficient machine learning systems. Some of the most popular libraries are TensorFlow, Pytorch, Scikit-learn and Keras.

TensorFlow

TensorFlow is a framework designed for both traditional and deep learning. It was developed by Google and it is well-suited for complex computations and large-scale environments. Some of its strengths are flexibility and scalability. It supports neural network architectures and it can be deployed on various platforms.

Keras

Keras is a high-level API designed to simplify the training of deep learning models. It runs on top of TensorFlow, and it offers a user-friendly interface. Therefore, building and training models with Keras is quick and easy, but it offers less control than TensorFlow.

Scikit-learn

Scikit-learn is optimized for performance in classical machine learning tasks but may not handle deep learning as efficiently. It is more suited for academic and smaller-scale projects without extensive deployment needs, as mentioned in [Rit24].

Scikit-learn is good at handling classical algorithms such as support vector machines (SVM), decision trees (DT), and gradient boosting techniques like XGBoost. These features make it particularly suitable for projects that involve structured data and require well-tested algorithms for predictive modeling.

In this thesis, Scikit-learn was chosen for several reasons:

- **Ease of Use:** Scikit-learn provides a simple and consistent API that works well with other Python libraries such as pandas and NumPy. This makes data preprocessing and model training straightforward and efficient.
- **Performance:** The library is fast and efficient for traditional machine learning tasks, allowing for quick testing and improvement of the models.
- **Comprehensive Tools:** It includes many tools for data preprocessing, like turning text into numbers, which is a key step in preparing the dataset for model training.
- **Wide Algorithm Support:** It supports a variety of machine learning algorithms out of the box. For this application, SVM, decision trees, and XGBoost were trained due to their effectiveness in handling the features of the dataset and providing accurate predictions.
- **Integration with Jupyter Notebooks:** Using Jupyter Notebooks makes it easy to work interactively, see results quickly, and adjust the code as needed. This is very helpful for exploring the data and tuning the models.

Additionally, the library has great documentation and lots of community support, which makes it a good choice for both beginners and more experienced users. By using Scikit-learn, this project can efficiently detect panic disorders using proven machine learning techniques.

Chapter 4

AI models

This chapter contains three sections. In the first section it is presented the dataset used for training the models. The second section details the preprocessing steps applied to the data. The third section describes the training process for the three algorithms: SVM, Decision Tree, and XGBoost. Lastly, a section which compares and evaluates the results, including tuning the best model.

SVM, DT, and XGBoost were chosen for this project for several reasons. SVM is great for handling high-dimensional data. Decision Trees are easy to interpret and can easily capture non-linear patterns. XGBoost is known for its high accuracy and can handle missing data well. They also provide useful insights into feature importance. Together, these models offer a good mix of simplicity, interpretability, and high accuracy.

4.1 Dataset

The dataset used for the Panic Disorder Detection application is sourced from Kaggle, provided by Muhammad Shahid Azeem [Aze21]. It contains 120000 records, which were gathered from documents observations. The data is split into two files, one for training with 100000 labeled records, and one for testing with 20000 records. This dataset includes attributes related to the health of the participant and the demographics. These are:

- **Participant ID:** An identifier for each participant. This does not affect the prediction of the model.
- **Age:** The participant's age at the time of data collection
- **Gender:** The gender (Male, Female).
- **Family History:** Indicates whether there is a family history of panic disorder (No, Yes).

- **Personal History:** Indicates whether the participant has a personal history of panic disorder (Yes, No).
- **Current Stressors:** The level of current stressors that could influence their mental health (Moderate, High, Low).
- **Symptoms:** A list of symptoms experienced by the participant (e.g., Shortness of breath, Panic attacks, Chest pain).
- **Severity:** The severity of the symptom (Mild, Moderate, Severe).
- **Impact on Life:** The impact of the symptom on the participant's life (Mild, Significant, Moderate).
- **Demographics:** The living area of the participant (Rural, Urban).
- **Medical History:** Any existing medical conditions (Diabetes, Asthma, Heart disease).
- **Psychiatric History:** Any existing psychiatric conditions (Bipolar disorder, Anxiety disorder, Depressive disorder).
- **Substance Use:** Substance use history (Drugs, Alcohol).
- **Coping Mechanisms:** Strategies used by the participant to cope with the symptoms or stress (Socializing, Exercise, Seeking therapy, Meditation).
- **Social Support:** The level of social support available (High, Moderate, Low).
- **Lifestyle Factors:** Factors such as sleep quality, exercise, and diet.
- **Panic Disorder Diagnosis:** Represents the final prediction (0 for no panic disorder, 1 for possible panic disorder).

4.2 Data Preprocessing

After analyzing the data using exploratory data analysis, the preprocessing part starts.

Missing Values

The dataset used for training has missing values for Medical History, Psychiatric History and Substance Use. This is not a problem for DTs or XGBoost, however SVMs cannot handle missing values. These are categorical values and they do not

have a specific importance order. To address the missing values, we use predictive imputation.

This method involves using a model to predict and fill in the missing values based on the other available data in the dataset. For this, we create a function which will use a Random Forest model. This algorithm builds DTs during training and uses the average or majority vote of each DT prediction [Das22]. It is a suitable choice because it handles missing data and calculates the importance of each feature. The function splits the data into training and testing sets based on whether the target column has missing values. It then separates the features (X) from the target variable (y) and converts categorical variables into numerical format using one-hot encoding. Lastly, it trains a Random Forest model (Classifier or Regressor) to predict and fill in the missing values, updating the original DataFrame. This function will be used to loop through each column that requires imputation.

Another way used to handle missing values is with a simple imputation (SimpleImputer() in sklearn), by replacing the NaN categorical values with the most frequent value, and numerical values with the mean.

Feature Scaling

In the dataset, we have the column Age which is a numerical feature. The technique used for scaling it is standardization (StandardScaler in Sklearn), which centres the values around the mean with the deviation. Feature scaling is crucial for SVMs, but not important for DT or XGBoost.

Encoding

The rest of the columns have categorical variables. Therefore, encoding is essential. For the SVM, One-Hot encoding was used, and for DT and XGBoost, Label encoding.

One hot encoding transforms each column into multiple binary columns. So for the Current Stressor feature the Table 4.1 will be created:

Current Stressor	Current Stressor_Low	Current Stressor_Moderate	Current Stressor_High
Low	1	0	0
Moderate	0	1	0
High	0	0	1

Table 4.1: One Hot Encoding for "Current Stressors"

Label encoding creates a unique integer value for each category. The Table for Current Stressor is 4.2:

We repeat for the test data the same processes.

Current Stressor	Label
Low	0
Moderate	1
High	2

Table 4.2: Label Encoding for "Current Stressors"

4.3 Cross validation

For the SVM and other models like Logistic Regression, the approach with the predictive imputation using Random Forest and the one hot encoding showed a high and consistent performance across both the training and validation sets. The learning curve for the SVM indicates that the model improves as there is more training data available. This it can be seen in Figure 4.1.

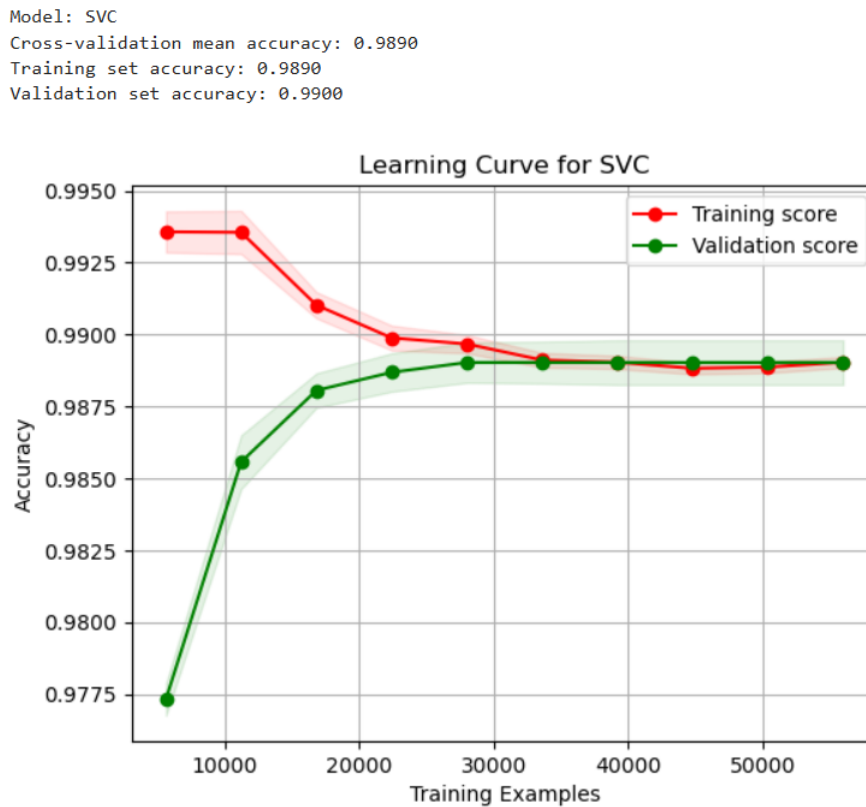
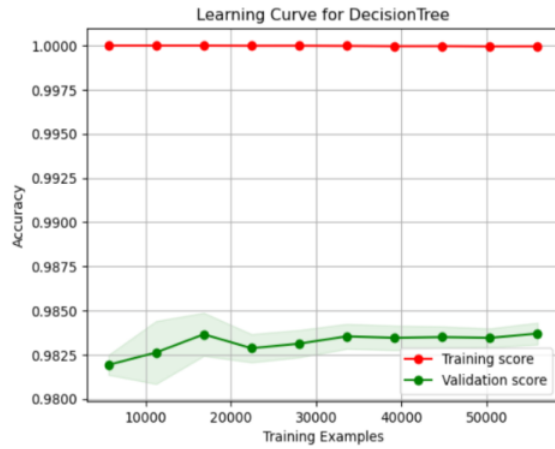


Figure 4.1: SVM Learning Curve

For the tree based models, such as DT, Random Forest, XGBoost, this approach, without any constraints and regularization, lead to overfitting, Figure 4.2. Therefore, the accuracy for the validation sets was lower. For Simple Gradient Boosting and AdaBoost, the accuracy was slightly lower, but the they did not overfit that much.

To try to further improve the performance of the XGBoost and DT models, another approach was implemented. For the missing values a simple imputation was

Model: DecisionTree
 Cross-validation mean accuracy: 0.9837
 Training set accuracy: 0.9999
 Validation set accuracy: 0.9847



Model: XGBClassifier
 Cross-validation mean accuracy: 0.9868
 Training set accuracy: 0.9980
 Validation set accuracy: 0.9880

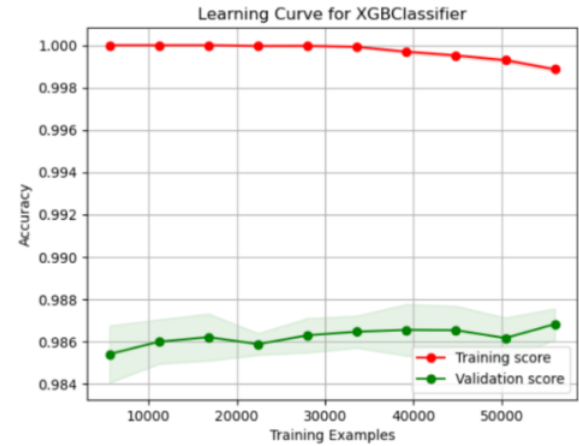
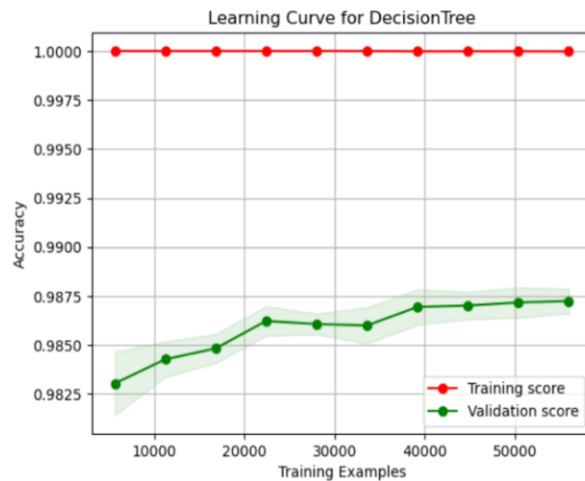


Figure 4.2: DT Learning Curve

used, and for categorical values, label encoding, since it is a better suite for tree based models. Despite these changes, the learning curves remained similar to those from before, shown in figure 4.3. This indicates that while the preprocessing steps are essential for ensuring clean and consistent data, they did not significantly modify the models learning behavior.

Model: DecisionTree
 Cross-validation mean accuracy: 0.9872
 Training set accuracy: 1.0000
 Validation set accuracy: 0.9889
 Training set error: 0.0000
 Validation set error: 0.0111



Model: XGBoost
 Cross-validation mean accuracy: 0.9895
 Training set accuracy: 0.9983
 Validation set accuracy: 0.9900
 Training set error: 0.0017
 Validation set error: 0.0100

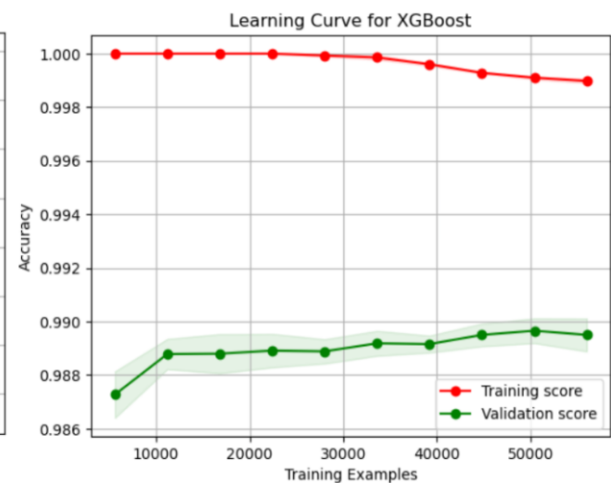


Figure 4.3: DT new Learning Curve

4.4 Model Training

Initially, the training dataset is split into training and validation, by allocating 70% of the data for training and 30% for validation. After splitting the data, a pipeline that includes both preprocessing and the classifier is created. The preprocessing method for each algorithm is explained in 4.2. After this step, the stratified k-fold cross-validation is used in order to examine overfitting, more details are in 4.3. The model is then trained using the fit method of the pipeline and its accuracy is evaluated on the validation set. This method applies all the preprocessing steps and then fits the ML model to the training data at the same time. By using the pipeline, we ensure that any new data, like validation or test data, goes through the same preprocessing steps.

The SVM model is then evaluated on the test data, which has been preprocessed in the same way as the training and validation sets. The DT and XGBoost however, require additional steps since their learning curves indicated overfitting possibility.

Overfitting occurs when the model learns irrelevant details instead of the patterns, reducing its ability to adapt to unseen data.

DT

For the DT, the training error is 0.000 and validation error is 0.011, which is a sign of overfitting.

Pruning is commonly used to reduce overfitting for DT. In order to find the hyperparameters 'max_depth', 'min_samples_split', 'min_samples_leaf', a Bayesian Optimization is performed. This builds a probabilistic model of the objective function and uses this model to select the hyperparameters that are likely to improve the objective. It is more efficient than Randomised Search [SFP⁺05], because its focus is on the most relevant hyperparameter combinations. This is computed for a different number of iterations, in order to find the best result, but after the 100th the results remained the same. The results for 100 iterations are 'max_depth'=10, 'min_samples_leaf'=1, 'min_samples_split'=20. The validation accuracy is 0.9905.

The results after pruning the DT are presented in Figure 4.4. There is a visible improvement, the model having a higher accuracy rate, and minimized overfitting now.

XGBoost

For XGBoost the difference is slightly lower than for DT, with training error 0.0017 and validation error 0.0100. The technique for reducing overfitting in XGBoost is tuning the hyperparameters.

Decision Tree Pruned Validation Accuracy: 0.9905
 Decision Tree Pruned Training Accuracy: 0.9913
 Decision Tree Pruned Training Error: 0.0087
 Decision Tree Pruned Validation Error: 0.0095

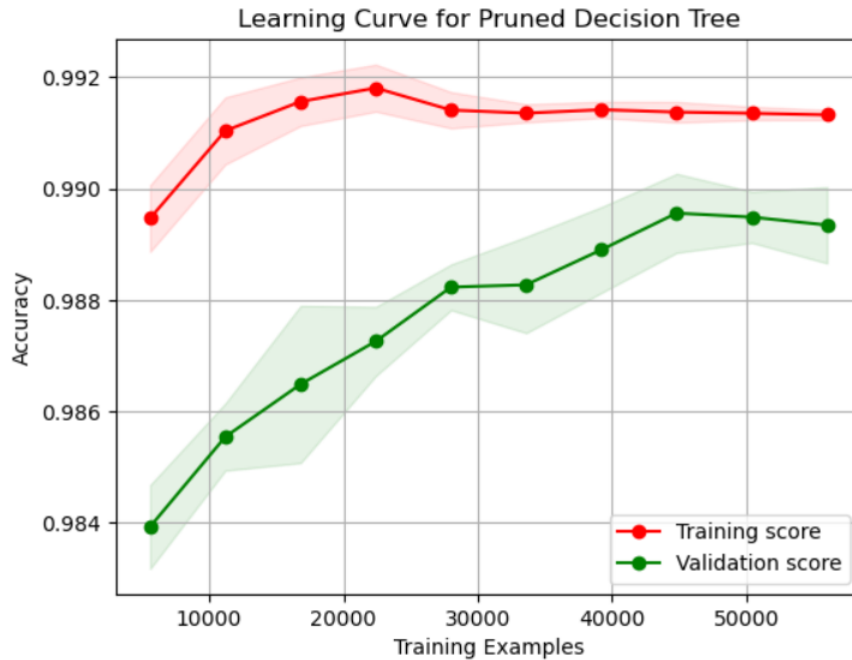


Figure 4.4: DT new Learning Curve

Optuna is used in order to find the best parameters because it automates the hyperparameter search, providing efficiency. For the number of iterations 50 and 200 the hyperparameters were identical. The results are 'n_estimators'=353, 'max_depth'=4, 'learning_rate'=0.076, 'subsample'=0.976, 'colsample_bytree'=0.864, 'gamma'=0.059, 'min_child_weight'=4. After tuning the algorithm, we can observe in Figure 4.5 an improvement in the overfitting.

4.5 Model Evaluation

Performance Comparison

These results, from Table 4.3 show that all three models captured the underlying patterns in the data and can make reliable predictions on new data. XGBoost had the best performance out of the three models.

Feature Importance

Both the DT and XGBoost models maintained a consistent high accuracy (0.990) across most feature levels, such as Gender, Family History and Personal History. However, there were challenges in predicting some features, suggesting room for improvement in those areas.

Training accuracy: 0.9929
 Validation accuracy: 0.9908
 Training error: 0.0071
 Validation error: 0.0092

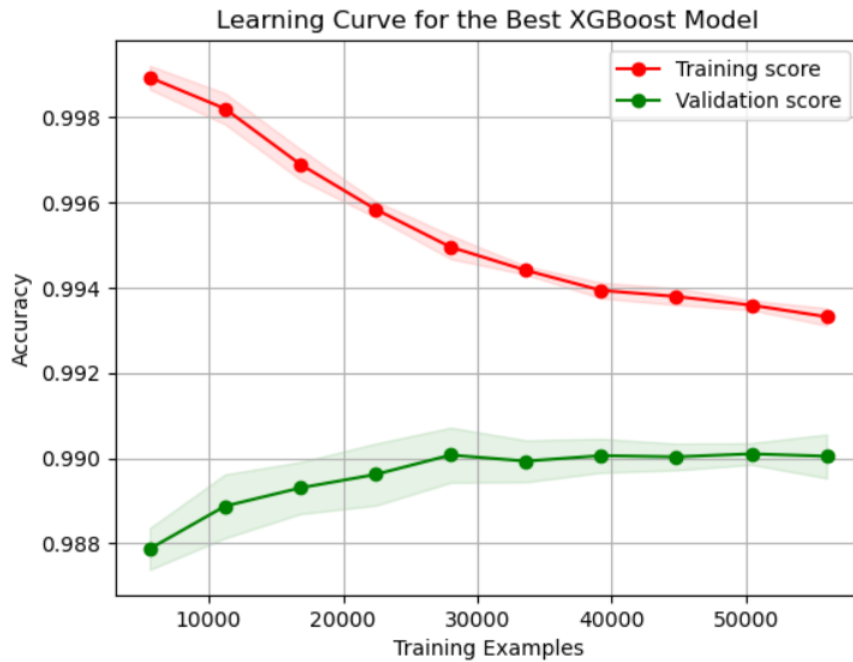


Figure 4.5: XGBoost new Learning Curve

Table 4.3: Model Performance Comparison

Metric	SVM	Decision Tree	XGBoost
Training Accuracy	98.90%	99.13%	99.29%
Validation Accuracy	99.00%	99.05%	99.08%
Testing Accuracy	98.99%	99.07%	99.11%
Training Error	1.10%	0.87%	0.71%
Validation Error	1.00%	0.95%	0.92%
Testing Error	1.01%	0.93%	0.89%

Some specific feature challenges are:

- **Age:** Both models showed variations, however the XGBoost model maintained a better consistency across more age levels.
- **Severity:** Both models struggled with Severity 'Severe', which means it is more challenging to predict it accurately.
- **Impact on Life:** Both models found Impact on Life 'Significant' to be a difficult level.
- **Current Stressors:** The lowest accuracy was found for Current Stressors 'High' in both models, but the XGBoost model handled it better.

- **Symptoms:** Symptoms 'Panic attacks' was the lowest performing level in both models.
- **Lifestyle Factors:** Both models showed significant differences in performance for Lifestyle Factors 'Sleep Quality'.

The dependence plots, from Figure 4.6 provide a visual representation of how changing a feature's value influences the prediction, thereby aligning with the feature importance scores in the table 4.4.

Table 4.4: Feature Importance

Feature	DT Importance	XGBoost Importance
Lifestyle Factors	0.1064	0.504852
Severity	0.1492	0.095438
Family History	0.1334	0.076442
Current Stressors	0.1021	0.075464
Impact on Life	0.1234	0.065972
Symptoms	0.0789	0.064504
Personal History	0.0774	0.035729
Social Support	0.0591	0.024029
Coping Mechanisms	0.0894	0.023138
Demographics	0.0277	0.012005
Medical History	0.0338	0.009131
Psychiatric History	0.0085	0.005238
Substance Use	0.0043	0.004424
Age	0.0060	0.001903
Gender	0.0003	0.001731

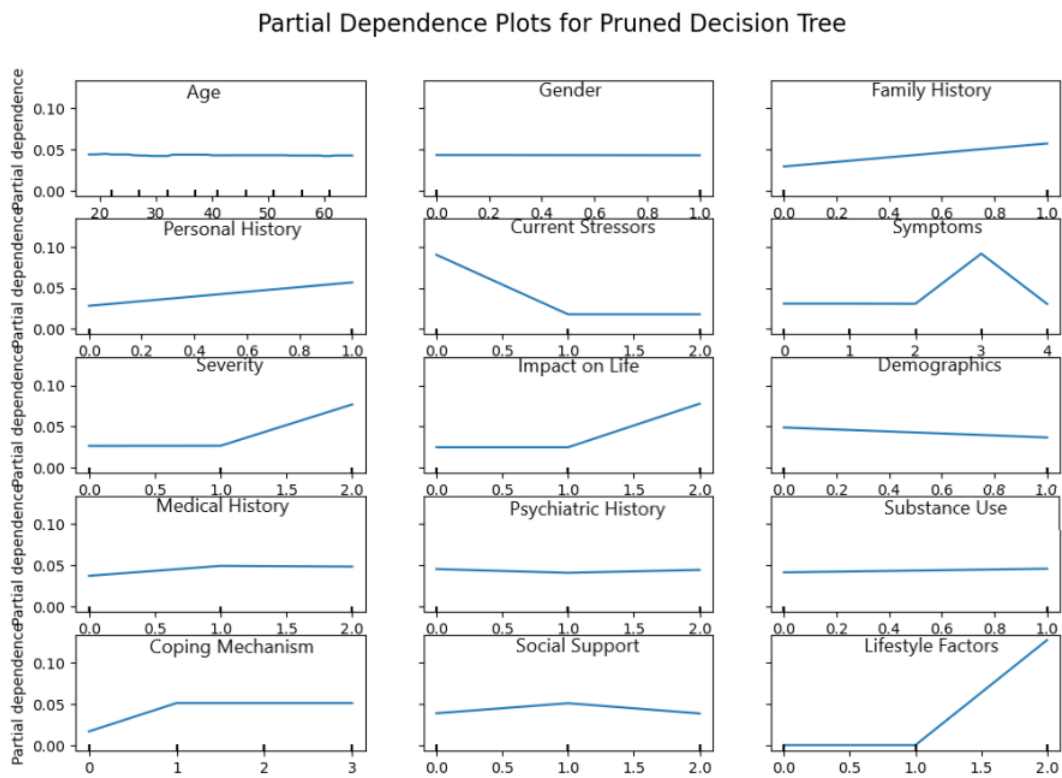


Figure 4.6: Dependence Plots for DT

Chapter 5

Application

In this chapter, we will focus on the practical aspects of our solution. Firstly, the architecture of the application will be presented, followed by the implementation together with UX/UI design.

5.1 Architecture

This section will cover the interaction between different components of the application. In the first part, the use case diagram illustrates the overall functionality and interactions of the users with the system. In the next part, sequence diagrams will provide a detailed view of how the components interact with each other over time to accomplish specific tasks.

5.1.1 Use Case Diagram

A Use Case Diagram illustrates a series of action sequences that a program performs when interacting with external entities (actors) to achieve a specific, useful outcome for those actors. It outlines what the program or subprogram does without detailing how the functionality is implemented. The behavior of a use case can be specified by describing a clear flow of events that includes how the use case starts and ends when interacting with actors, what objects are exchanged, and any alternative flows. In other words, it represents possible scenarios for using the system.

For the Panic Disorder Detection Application the Use Case Diagram is the one from Figure 5.1.

The actor in this Use Case Diagram is the User, which represents the medical staff that will use the detection system.

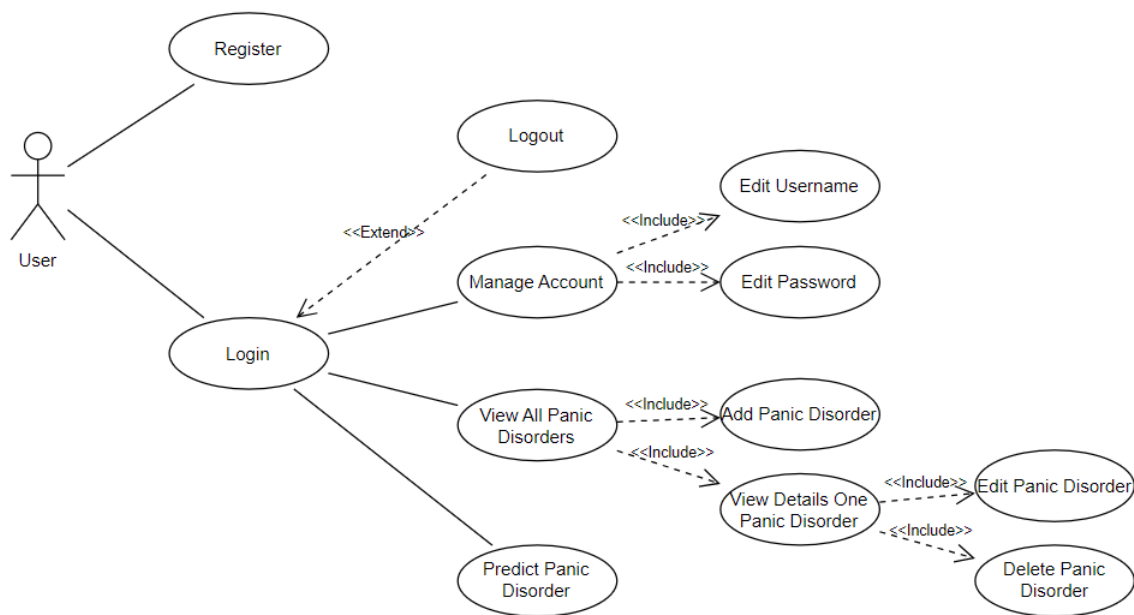


Figure 5.1: Use Case Diagram

Use Cases Description

- **Register:** Sign up functionality. The user can create a new account by specifying the username, name and password.
- **Login:** Authenticate functionality. The user has to log in in order to access the system, by entering their username and password.
- **Logout:** Allows the user to end the current session.
- **Manage Account:** Allows a user to manage their account fields, by providing 'Edit Username' and 'Edit Password' fields.
- **Edit Username:** Allows a user to change their name.
- **Edit Password:** Allows a user to change their account password.
- **View All Panic Disorders:** Provides a list with all existing panic disorder entries from the system.
- **Add Panic Disorder:** The user can add a new panic disorder entry to the system by completing all the required fields.
- **View Details One Panic Disorder:** Provides the detailed information about a specific panic disorder entry.

- Edit Panic Disorder: Modify all the fields from the entry.
- Delete Panic Disorder: Remove the panic disorder entry from the system
- Predict Panic Disorder: Provides functionality to predict the likelihood of a panic disorder based on user inputs and selecting the algorithm.
- Help Disorder: Offers managing techniques for panic disorder.

5.1.2 Sequence Diagrams

Interaction diagrams are used to model how a set of objects behave and interact within a specific context to achieve a particular goal. These diagrams include the objects involved and the relationships between them. Sequence diagrams are a type of interaction diagram that visually represent how various components of the system, such as the User, Website, Server, and Database, interact over time to complete specific tasks. These diagrams focus on the temporal sequence of messages exchanged between objects. The X-axis of a sequence diagram indicates the different objects, while the Y-axis shows the messages they exchange in chronological order. Each object has a lifeline depicted by a vertical dotted line, and the periods during which objects are active and executing tasks are shown as thin rectangles on their lifelines.

In this section, we will examine the workflow of the application using sequence diagrams.

Register Sequence Diagram

The sequence diagram for the register functionality is presented in Figure 5.2. The user navigates to the registration page, where the frontend shows a registration form. After the user fills in their details (username, password, first name) and submits the form, the frontend validates the provided data. If the data is valid, the frontend sends a request to the backend to register the user. The backend checks the database to see if the username already exists. If the username is available, the backend adds the new user details to the database, confirms the registration, and sends a success message back to the frontend, which then displays it to the user. If the username is already taken, an error message is returned to the frontend. If the input data is invalid, the frontend shows a validation error message to the user.

The Login process works similarly. When the user navigates to the login page and submits their credentials (username and password), the frontend validates the input data. If the data is valid, the frontend sends a login request to the backend. The backend checks the database to verify the username and password. If the credentials are correct, the backend generates a JWT token and sends it back to the frontend. The

frontend then stores the token and updates the user interface to reflect the logged-in state. If the credentials are incorrect, an error message is returned to the frontend, which then displays it to the user. If the input data is invalid, the frontend shows a validation error message.

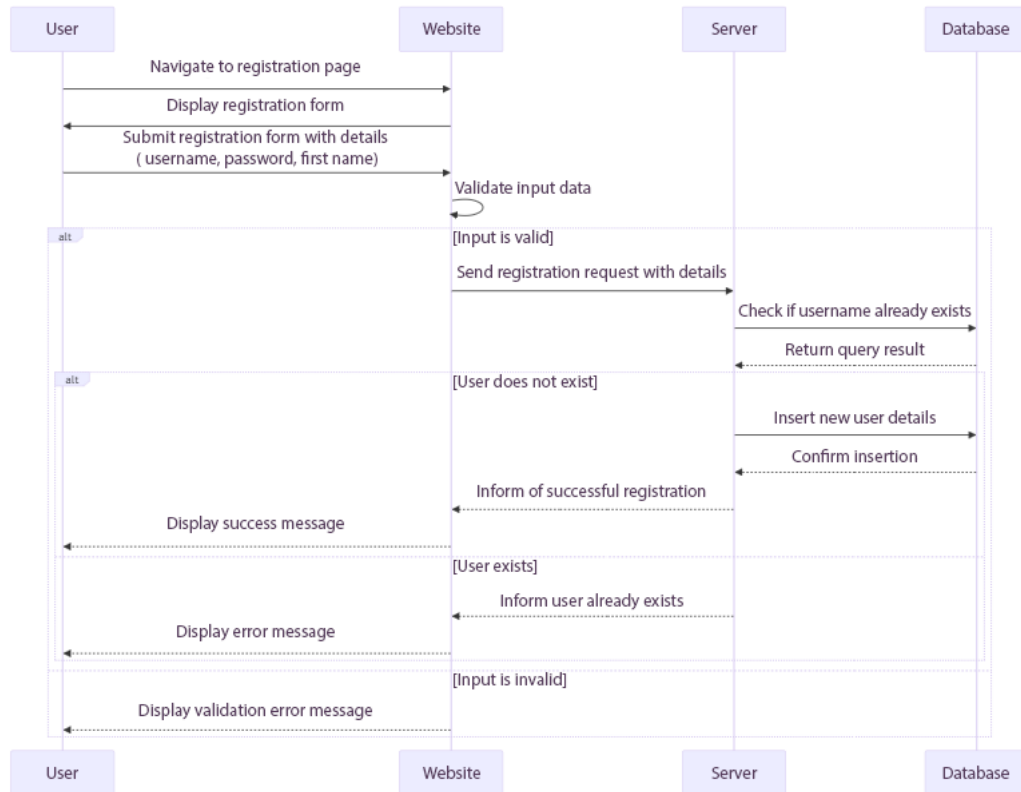


Figure 5.2: Register Sequence Diagram

Panic Disorder Entries Management Sequence Diagram

In Figure 5.3 is shown the management for panic disorder entries. The user navigates to the dashboard to view all disorders, where the frontend requests and displays the list of disorders from the backend. Upon clicking a disorder, the frontend requests and displays detailed information for that disorder, on a new page. The user can then choose to edit or delete the disorder. For editing, the frontend sends the updated details to the backend, which updates the database and confirms the update. For deletion, the frontend sends a delete request to the backend, which removes the disorder from the database and confirms the deletion. The frontend then displays success messages for both actions. The user can also cancel the operation at any time, without affecting the workflow.

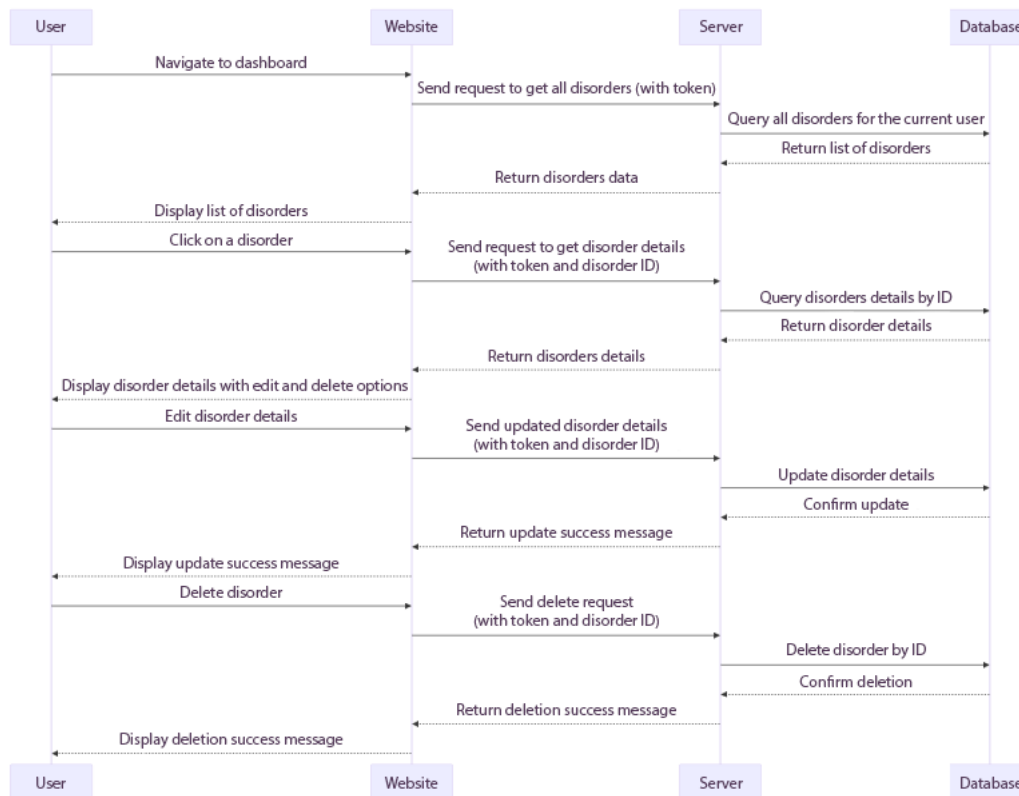


Figure 5.3: Panic Disorder Entries Management Sequence Diagram

Prediction Sequence Diagram

The prediction feature is displayed in Figure 5.4. Firstly, the user navigates to the prediction page, where the frontend displays a form for selecting an algorithm and a disorder. After the user makes their selection, the frontend sends a prediction request to the backend, which retrieves the disorder details from the database. The backend then processes the prediction using the specified algorithm (SVM, Decision Tree, or XGBoost) and returns the result to the frontend. If an invalid algorithm is specified, an error message is returned. In the end, the frontend displays the prediction result or error message to the user.

5.2 Implementation

In this section, the implementation approach for the Panic Disorder Detection Application is presented. The Model Deployment is presented in Figure 5.5. In the first section the Frontend technical details will be presented, then the Backend and lastly the Database.

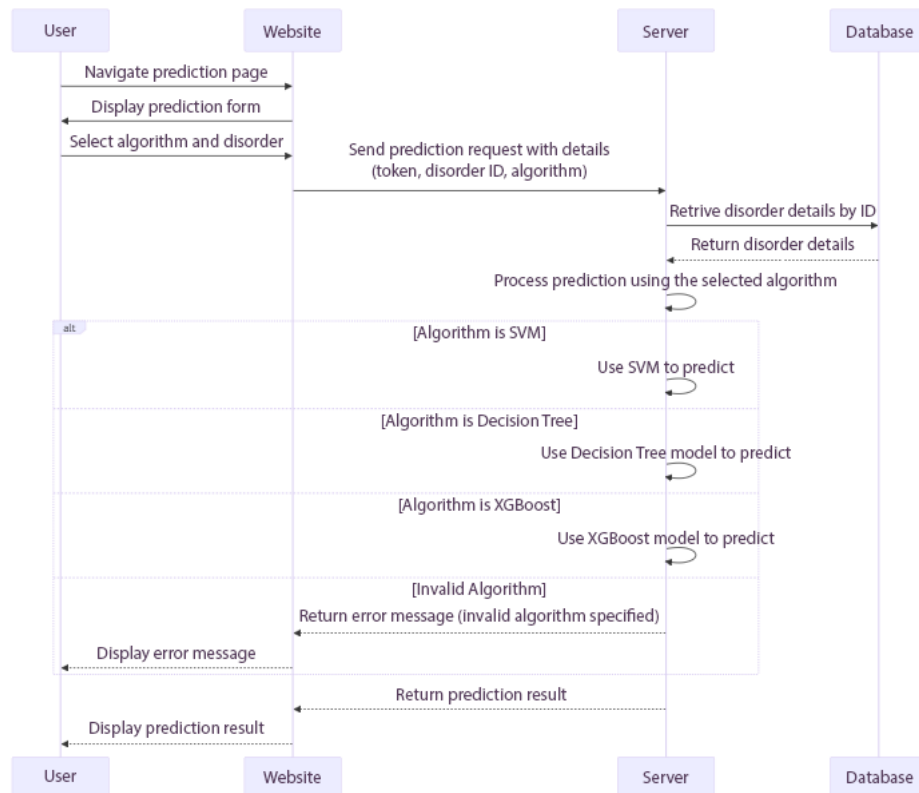


Figure 5.4: Prediction Sequence Diagram

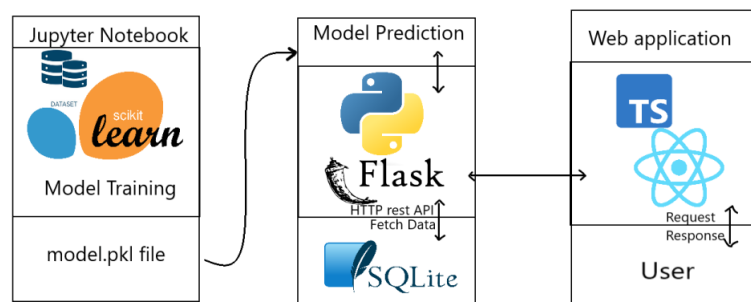


Figure 5.5: ML model deployment

5.2.1 Frontend

For the frontend of the application React along with TypeScript was used. This offers better type safety, enhanced development experience, and improved scalability. Typescript provides static type checking, which is useful in catching error during development. React allows creating reusable components, managing state effectively and handling routing smoothly.

Here are some of the key functionalities used from React:

React Hooks

- **useState:** This hook is used to manage the local state of components. For example, in the DisorderPage component, I used `useState` to manage the state of the disorder details, the editing mode, and the delete confirmation state.
- **useEffect:** This hook is used for side effects such as data fetching. In the DisorderPage, I used `useEffect` to fetch the disorder details when the component mounts or when certain dependencies change.

React Router For client-side routing, I used `react-router-dom`. This library allows defining routes for the application, managing navigation, and creating dynamic route parameters.

- **Routes:** The main routes of the application are defined in the App component. This includes routes for login, prediction, disorder details, and so on.
- **Navigation:** The `useNavigate` hook is used to programmatically navigate between different routes. For instance, after successfully updating or deleting a disorder, it is used to navigate the user back to the DisordersPage.

Component-Based Architecture By breaking down the UI into reusable components, we can manage and test each part of the application independently. Components like UserMenu, UserPanel, and DisorderPage are some examples of how the functionality and UI elements were encapsulated.

Fluent UI Fluent UI provides a set of customizable and accessible components that help in building modern web applications quickly. Components, such as `DefaultButton`, `Dialog`, and `Panel` were used to create a consistent and visually appealing UI.

The Frontend of the project is organized into different layers in order to maintain a clean architecture. Each layer is responsible for a specific aspect of the application, ensuring that the code is modular and reusable.

There is a components layer, which has a 'login-register' for handling login and registration. Similarly, a 'user-menu' component for displaying the options for navigating different parts of the application. Lastly, a 'user-panel' component which manages actions such as logging out.

The next layer is used for the models 'User', 'Disorder' and the login response.

Each page is included in the Page layer. Some of the pages included are Login page, Get Disorders page, Add Disorder page and so on.

The API calls related to disorders and users are found in the services layer. In the same layer there is a file that manages authentication related API calls.

There is also an Utils layer, which stores functions and components that support the main functionalities. This ensures that the functions used throughout the application can be reused across different component and pages.

Lastly, the App.tsx is the main part of the React application, handling all the routing and integrating components, pages. It uses hooks for state management and navigation, ensuring that only authenticated users can access protected routes.

5.2.2 Backend

For the backend of the application I chose to use Flask because it's simple and flexible. Flask is a lightweight web framework that's easy to set up and perfect for building scalable web applications.

Here are some of the key functionalities used from Flask:

Flask-Restx This extension was used to define API endpoints and handle requests and responses.

JWT Authentication The JSON Web Token ensures that only authenticated users can access the application, by securing the API endpoints.

Model Serialization The data is converted into JSON format using SQLAlchemy models. This facilitated the send and receive data process between the frontend and the backend.

The backend follows the separation of concerns principle. There is an API layer, in routes.py, where all the API endpoints are defined. It also handles the HTTP requests and responses and the authentication checks. Another layer is the data layer, in models.py. The models (User and PanicDisorderEntry) are defined here. The configuration layer holds the configuration settings, such as database connection and JWT secret keys. The pre-trained AI models are stored in a different directory, together with the preprocessing steps for a new entry.

5.2.3 Database

The database system used is SQLite. SQLite is a lightweight database and it provides a simple and efficient way to store data locally without the need for a separate database server. SQLite is known for its reliability, ease of use, and integration with various programming environments.

To interact with the database, SQLAlchemy was utilized as the Object Relational Mapper (ORM). SQLAlchemy provides a high-level abstraction for database interactions, allowing developers to work with Python objects instead of writing raw SQL queries. The main reasons for choosing these two are their simplicity and seamless integration with Flask.

The database includes three tables, as it can be seen in Figure 5.6. One of them is the User table which manages user accounts. It stores information such as user ID, username, first name, the password (hashed) and a flag for authentication.

Another table is The PanicDisorderEntry which is used to store anonymous records related to panic disorder cases. This table includes factors that contribute to panic disorders. For each individual entry there are included various aspects such as demographic data, medical and psychiatric histories, current stressors, symptoms, and coping mechanisms.

The table which keeps track of the JWT tokens is JWTTokenBlocklist. The records include Id, JWT token and the date of creation. This table prevents unauthorized access.

5.3 UX/UI

UI (User Interface) design refers to what users see on the application screen – text, colors, backgrounds, icons, and moving elements like animations. UX (User Experience) design focuses on how users navigate through all the different elements of the UI. For UX designers, it's about optimizing the user experience flow and eliminating any issues during interactions. In this section, we'll explore UI and UX design, focusing on how they work together to create the final application.

The main pages of the Panic Disorder Detection application are:

Login 5.7

The first thing the user sees when he enters the application is the Login Page. This contains two fields, the username, and the password. It contains a simple login button as well. There is also an option for sign up in case the user does not have an account yet. The register page is similar to the login page.

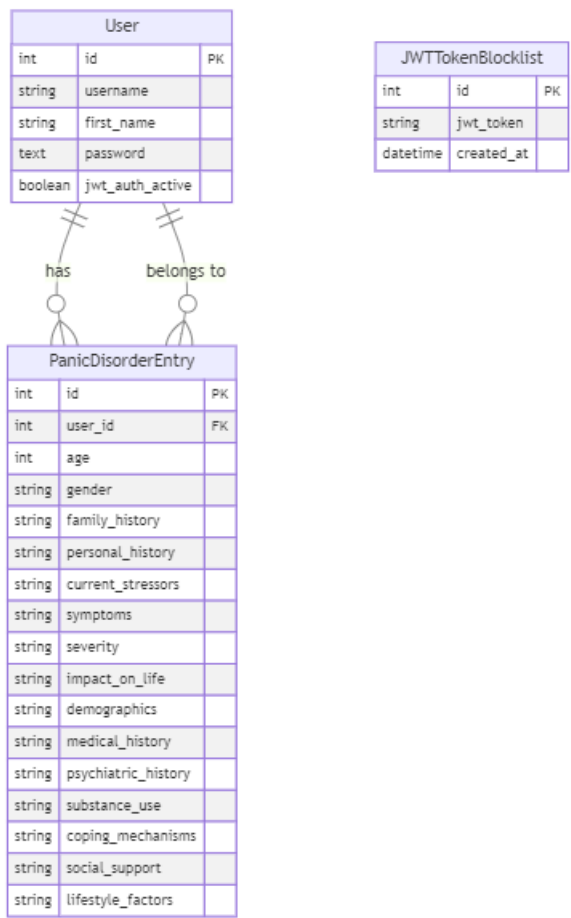


Figure 5.6: ML model deployment

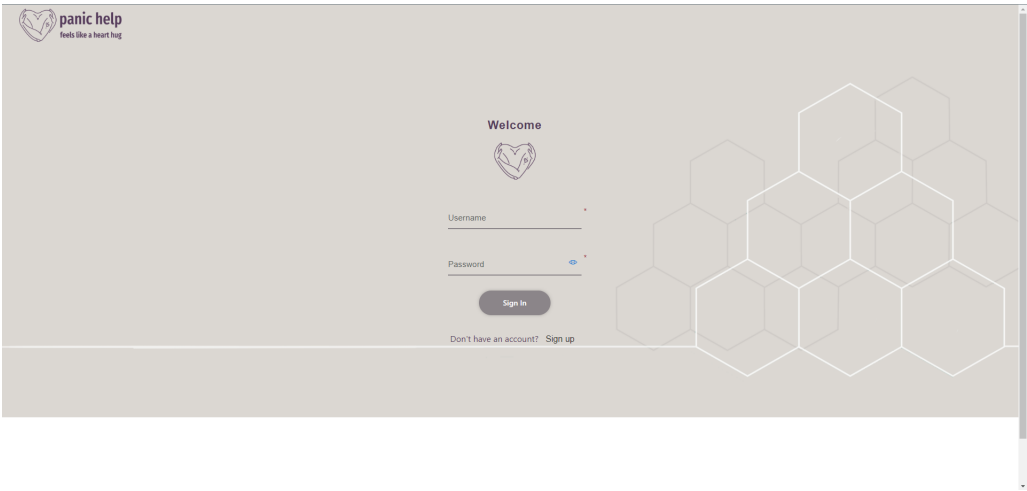


Figure 5.7: Login Page

All Disorders 5.8

After logging in, the user is redirected to a page which contains all its entries. On this page, the user can select the Add button in order to add a new panic disorder entry, or click on one of the available entries in order to see its details, edit or delete it.



Figure 5.8: All Disorders Page

Add Disorder 5.9

This is the page for adding a new entry. The fields are prepopulated in order to offer clear instructions. The user can either save the entry or cancel the action.

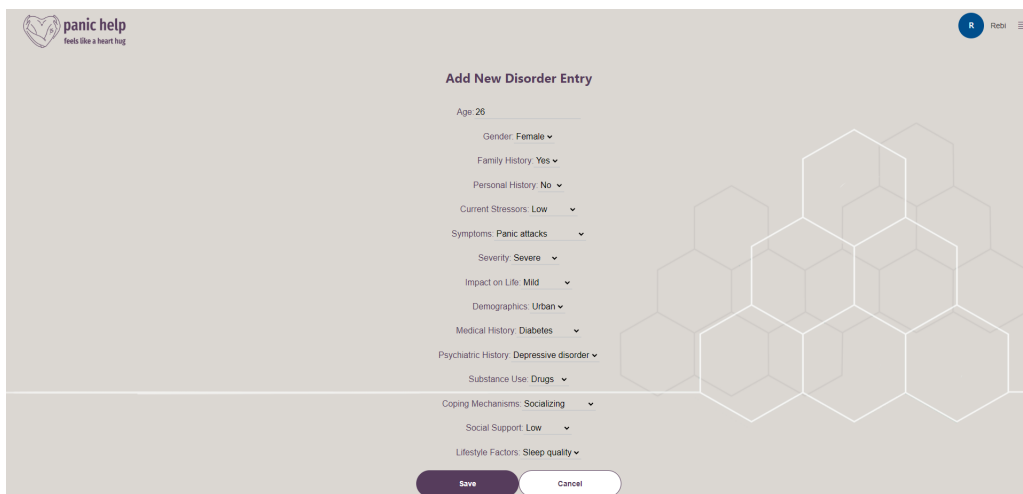
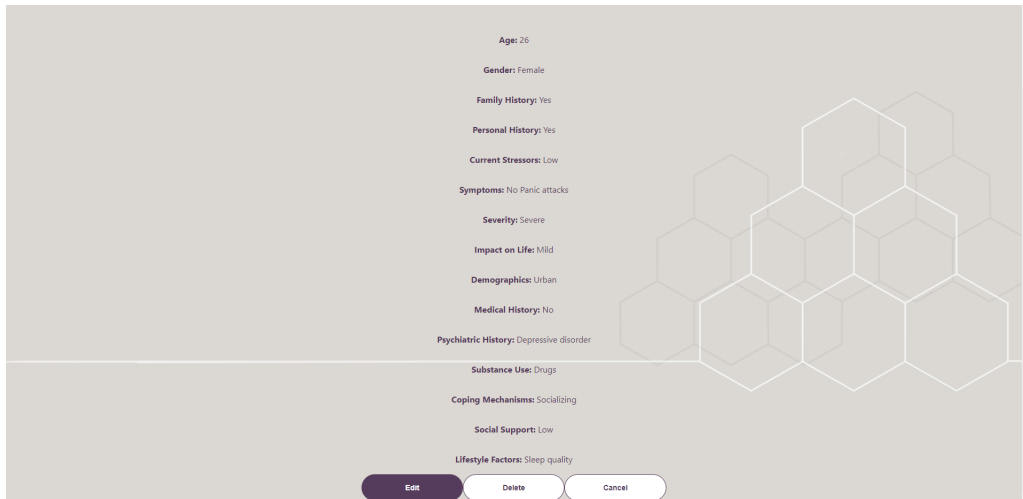


Figure 5.9: Add Disorder Page

One Disorder 5.10

On this page, the details of the entry are shown. The user can either modify it, delete it, or cancel the action. When pressing the Cancel button, the user is taken back to the All Disorders page. If the choice is Edit, the user can modify any of the fields.



The screenshot shows a form titled 'One Disorder' with the following fields:

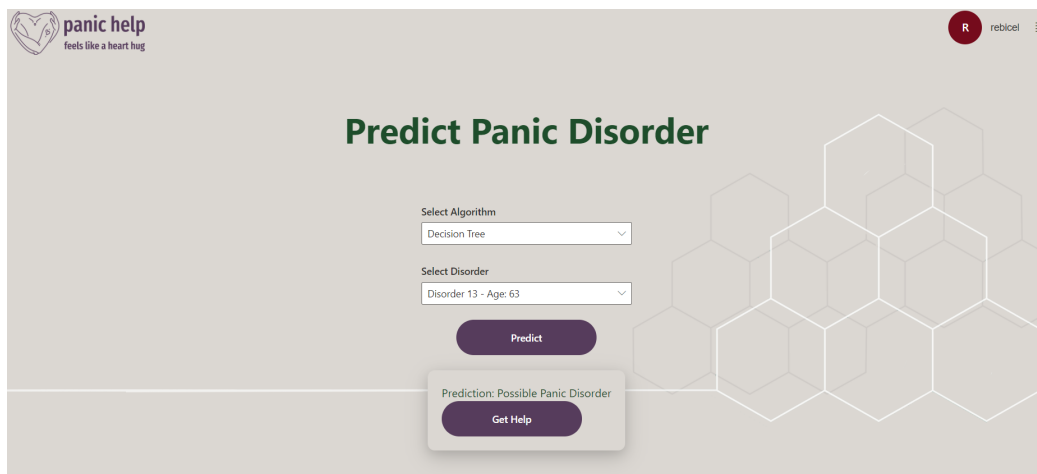
- Age: 26
- Gender: Female
- Family History: Yes
- Personal History: Yes
- Current Stressors: Low
- Symptoms: No Panic attacks
- Severity: Severe
- Impact on Life: Mild
- Demographics: Urban
- Medical History: No
- Psychiatric History: Depressive disorder
- Substance Use: Drugs
- Coping Mechanisms: Socializing
- Social Support: Low
- Lifestyle Factors: Sleep quality

At the bottom, there are three buttons: 'Edit' (purple), 'Delete' (white), and 'Cancel' (white).

Figure 5.10: One Disorder Page

Predict 5.11

On this page, the user can select one of the three prediction algorithms and data for the panic disorder and he can view the results by clicking the predict button.



The screenshot shows a page titled 'Predict Panic Disorder' with the following elements:

- Logo: 'panic help feels like a heart hug' (top left)
- User: 'R rebicel' (top right)
- Form fields:
 - Select Algorithm: Decision Tree (dropdown)
 - Select Disorder: Disorder 13 - Age: 63 (dropdown)
- Buttons: 'Predict' (purple), 'Get Help' (purple)
- Background: Hexagonal pattern

Figure 5.11: Predict Page

Account 5.12

This page shows information about the user account and it allows the user to modify the details.

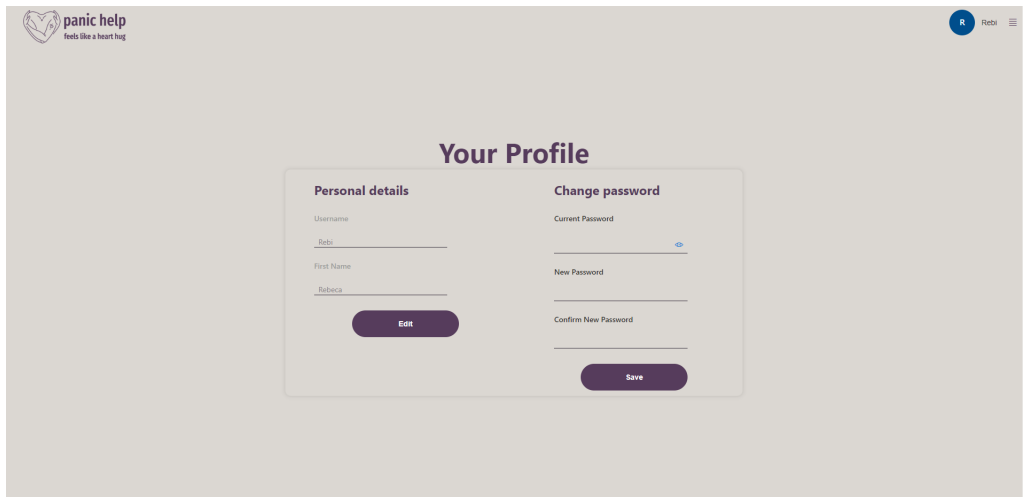


Figure 5.12: Account Page

Furthermore, on every page there is a menu that offers an easy navigation from one page to the other. And there is also a page which shows ways to manage and improve panic disorder.

5.4 Testing

AI Models Testing

The AIs were tested using a dataset which contains 20000 test entries. The testing error for the models varied between 0.89% and 1.01%.

Application Testing

For testing the application flow and functionalities, a Session-Based Test Management (SBTM) was used. This approach focuses on dynamic and exploratory testing sessions to validate various functionalities.

The backend APIs were tested using Postman. This testing includes:

- **Authentication APIs:** Validating login and registration functionalities, verifying that JWT tokens are correctly generated and used.
- **CRUD Operations:** Ensuring Create, Read, Update, and Delete operations for panic disorder entries and for user entry are functioning correctly.
- **Prediction API:** Testing the prediction endpoints with various inputs to verify the AI models' responses. Testing the prediction results was done by using data from the existent training and testing databases.

The frontend was tested using direct interaction with the final product. Key aspects tested include:

- **Form Validations:** Ensuring all forms, such as login, have proper validations for required fields, input formats, and error messages.
- **Navigation:** Verifying that the navigation between different pages, such as the login page, dashboard, add disorder page, prediction page, is functional and intuitive.
- **UI/UX:** Checking that the user interface is user-friendly. This was tested by asking user feedback from users who were not familiar with the application.

Most of these functionalities were tested during the development of the application too. This iterative testing process allowed for continuous improvement and early detection of any bugs.

Chapter 6

Conclusions and Future Work

The aim of this thesis is to compare three different AI models, SVM, Decision tree and XGBoost, for detecting panic disorder. Out of the three models, XGBoost performed the best with the accuracy on training set 99.29% and on testing set 99.11%. However, the other two models were efficient too in detecting panic disorder.

A web application was developed to provide a user-friendly interface for the models. This app allows users to input relevant data and receive a prediction about the likelihood of a panic disorder, which makes it a potential tool in assisting health-care professionals.

As this application is at its first version, there are several improvements that can be made. The dataset used for training is limited. Therefore, the first improvement is collecting more diverse data and then tuning the models with new data fields, such as air quality index and heart rate. This can improve the model's accuracy in real life situations and improve any biased predictions.

Another area for future work is the development of the web application, by adding more functionalities such as a support for patient real-time monitoring. This could process the physiological data in real time and offer timed predictions, to prevent any possible panic attacks. Also, the page which offers solutions for managing panic disorder can use an AI, in order to give personalized support.

In conclusion, while this thesis achieved its purpose of comparing different machine learning models and creating an application that incorporates them, both the application and the AI models can benefit from further development in the future.

Bibliography

- [Aze21] Muhammad Shahid Azeem. Panic disorder detection dataset, 2021. Online; accessed 9 April 2024. <https://www.kaggle.com/datasets/muhammadshahidazeem/panic-disorder-detection-dataset>.
- [Bro18] Jason Brownlee. *XGBoost with Python: Gradient Boosted Trees with XGBoost and scikit-learn*. Machine Learning Mastery, v1.10 edition, 2018.
- [Cha22] Chan-Hen Tsai, Pei-Chen Chen, Ding-Shan Liu, Ying-Ying Kuo, Tsung-Ting Hsieh, Dai-Lun Chiang, Feipei Lai, Chia-Tung Wu. Panic Attack Prediction Using Wearable Devices and Machine Learning: Development and Cohort Study. *JMIR Med Inform*;10(2):e33063, 2022.
- [Cri16] Cristie Glasheen, Kathryn Batts, Rhonda Kargn, Jonaki Bose, Sarra Hedden, Kathryn Piscopo . Impact of the DSM-IV to DSM-5 Changes on the National Survey on Drug Use and Health. *Substance Abuse and Mental Health Services Administration*, pages 91–95, 2016.
- [Das22] Sunil Kumar Dash. Handling missing values with random forest, 2022. Online; accessed 20 April 2024. <https://www.analyticsvidhya.com/blog/2022/05/handling-missing-values-with-random-forest/>.
- [Ger19] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, pages 159–191. O’Reilly Media, Sebastopol, CA, 2nd edition, 2019.
- [ICD92] *International Statistical Classification of Diseases and Related Health Problems*. ICD-10. World Health Organization, Geneva, Switzerland, 10th revision edition, 1992.
- [LKHT22] Ilias Lazarou, Anastasios L. Kesidis, George Hloupis, and Andreas Tsatsaris. Panic detection using machine learning and real-time biometric and spatiotemporal data. *ISPRS International Journal of Geo-Information*, 11, 2022.

- [MKC22] Huan Meng, Yun-Jeong Kang, and Dong-Oun Choi. Panic disorder prediction model based on machine learning algorithm. pages 1644–1648, 2022.
- [Nat22] National Institute of Mental Health. Panic Disorder: When Fear Overwhelms, 2022. Online; accessed 9 April 2024. https://www.nimh.nih.gov/health/publications/panic-disorder-when-fear-overwhelms#part_6105.
- [Ove] Stack Overflow. Tag trends: Vue.js, react.js, angular. Online; accessed 6 May 2024. <https://trends.stackoverflow.co/?tags=vue.js,reactjs,angular>.
- [Pan19] Arjun Panesar. *Machine Learning and AI for Healthcare: Big Data for Improved Health Outcomes*, pages 207–254. Apress, New York, NY, USA, 2019.
- [Rit24] Ritza. Scikit-learn vs. tensorflow vs. pytorch vs. keras, 2024. Online; accessed 6 May 2024. <https://ritza.co/articles/scikit-learn-vs-tensorflow-vs-pytorch-vs-keras/>.
- [SFP⁺05] Vitaly Schetinin, Jonathan Fieldsend, Derek Partridge, Wojtek Krzanowski, Richard Everson, Trevor Bailey, and Adolfo Hernandez. Comparison of the bayesian and randomised decision tree ensembles within an uncertainty envelope technique. *Journal of Mathematical Modelling and Algorithms*, 5, 2005.
- [SK23] Jakub Swacha and Artur Kulpa. Evolution of popularity and multiaspectual comparison of widely used web development frameworks. *Electronics*, 12(17):3563, 2023.
- [Vya22] Rishi Vyas. Comparative analysis on front-end frameworks for web applications. *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, 10, 2022.
- [Wad20] Corey Wade. *Hands-On Gradient Boosting with XGBoost and scikit-learn*. Packt Publishing Ltd., Livery Place, Birmingham, UK, 2020.