DEVLOPMENT OF THERAPY EFFICACY MODEL FOR AUTISM SPECTRUM DISORDER

A PROJECT REPORT

Submitted by

MEGA V M	**211521244032**
GURU VARSHENI A	**211521244015**
DHANUSHREE S	**211521244011**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND BUSINESS SYSTEMS

PANIMALAR INSTITUTE OF TECHNOLOGY
ANNA UNIVERSITY : CHENNAI 600 025

MAY 2025

PANIMALAR INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DEVELOPMENT OF THERAPY EFFICACY MODEL FOR AUTISM SPECTRUM DISORDER"** is the bonafide work of "**MEGA V M , GURU VARSHENI A AND DHANUSHREE S.**" who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. S.HEMALATHA, Ph.D.,D.Sc	**Dr. S.HEMALATHA,Ph.D,D.Sc**
HEAD OF THE DEPARTMENT	**HEAD OF THE DEPARTMENT**
Department of Computer Science and Business Systems, Panimalar Institute of Technology	Department of Computer Science and Business Systems, Panimalar Institute of Technology
Poonamallee, Chennai 600 123	Poonamallee, Chennai 600 123

Certified that the candidates were examined in the university project viva-voce held on ________________at Panimalar Institute of Technology,Chennai 600 123.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

A project of this magnitude and nature requires kind co-operation and support from many, for successful completion. We wish to express our sincere thanks to all those who were involved in the completion of this project.

We seek the blessings from the **Founder** of our institution **Dr. JEPPIAAR, M.A., Ph.D.,** for having been a role model who has been our source of inspiration behind our success in education in his premier institution.

We would like to express our deep gratitude to our beloved **Secretary and Correspondent Dr. P. CHINNADURAI, M.A., Ph.D.,** for his kind words and enthusiastic motivation which inspired us a lot in completing this project.

We also express our sincere thanks and gratitude to our dynamic **Directors Mrs. C. VIJAYA RAJESHWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYA SREE SAKTHI KUMAR, B.E, M.B.A., Ph.D.,** for providing us with necessary facilities for completion of this project.

We also express our appreciation and gratefulness to our respected **Principal Dr. T. JAYANTHY, M.E., Ph.D.,** who helped us in the completion of the project. We wish to convey our thanks and gratitude to our **Head of the Department, Dr. S. HEMALATHA Ph.D,D.Sc** for her full support by providing ample time to complete our project. We express our indebtedness and special thanks to our **Supervisor, Dr. S. HEMALATHA Ph.D,D.Sc** for her expert advice and valuable information and guidance throughout the completion of the project.

Last, we thank our parents and friends for providing their extensive moral support and encouragement during the course of the project.

TABLE OF CONTENTS

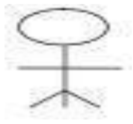
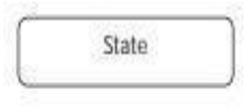
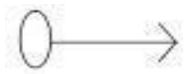
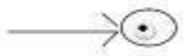
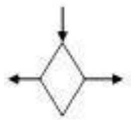
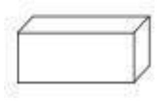
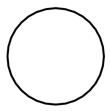
ABSTRACT

The prevalence of Autism Spectrum Disorder (ASD) has been steadily rising, with recent estimates indicating that 1 in 36 children are diagnosed with the condition, highlighting the urgent need for effective therapeutic interventions. However, autism therapy lacks a standardized approach for predicting the efficacy of various interventions, often leading to trial-and-error treatment selection. This study aims to develop robust predictive models for therapy efficacy in ASD patients, utilizing comprehensive clinical and behavioral data. Our analysis incorporates a diverse set of attributes, including patient demographics, cognitive test scores, therapy duration, baseline severity, social engagement levels, and caregiver-reported progress. By integrating these critical factors, we strive to achieve a holistic understanding of treatment effectiveness. Employing advanced machine learning techniques such as ensemble learning (Random Forest, XGBoost, and Support Vector Machines), Long Short-Term Memory (LSTM) networks for time-series tracking, and explainable AI methods like SHAP and LIME, we construct predictive models capable of accurately forecasting therapy outcomes. Rigorous evaluation through precision, recall, and F1-score assessments ensures model reliability. This approach aims to enhance personalized treatment recommendations, enabling data-driven decision-making for healthcare professionals and improving therapeutic outcomes for ASD patients.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

S.NO	NAME	NOTATION	DESCRIPTION
1.	Actor		It aggregates several classes into single classes
2.	Communication		Communication between various use cases.
3.	State		State of the process.
4.	Initial State		Initial state of the object
5.	Final state		Final state of the object
6.	Control flow	X	Represents various control flow between the states.
7.	Decision box		Represents decision making process from a constraint
8.	Node		Represents physical modules which are a collection of components.
9.	Data Process/State		A circle in DFD represents a state or process which has been triggered due to some event or action.

10.	External entity	▭	Represents external entities such as keyboard, sensors, etc.
11.	Transition	→	Represents communication that occurs between processes.
12.	Object Lifeline	▭	Represents the vertical dimensions that the object communications.
13.	Message	message	Represents the message exchanged.

CHAPTER 1

CHAPTER 1
INTRODUCTION

1.1 AN OVERVIEW OF PROJECT

Autism Spectrum Disorder (ASD) is a neurodevelopmental condition that has seen a significant rise in prevalence, with current estimates indicating that 1 in 36 children are diagnosed with ASD. Characterized by challenges in social interaction, communication, and repetitive behaviors, ASD varies in severity, making personalized treatment essential. Therapy for autism is diverse, encompassing interventions such as Applied Behavior Analysis (ABA), Speech Therapy, and Occupational Therapy. However, determining the most effective therapy for each individual remains a challenge due to the heterogeneity of ASD symptoms and responses to treatment.

This study introduces a predictive framework aimed at addressing the complexities of autism therapy selection. Traditional treatment approaches often rely on trial-and-error methods, leading to delays in effective intervention. Our project leverages advanced computational techniques to analyze patient data and predict the efficacy of various therapies, enabling a more structured and data-driven decision-making process.

By incorporating a diverse range of behavioral patterns, therapy response history, the system provides valuable insights into individualized treatment strategies. Utilizing machine learning methodologies, including ensemble models and time-series forecasting, the framework enhances therapy personalization, ensuring that patients receive the most suitable intervention at the earliest stage possible.

Through this research, we strive to bridge the gap between clinical expertise and data-driven decision-making, fostering improved therapeutic outcomes for individuals with ASD. By offering a structured and intelligent approach to therapy prediction, our project aims to contribute to the advancement of personalized autism care and early intervention strategies.

SCOPE OF THE PROJECT

This project focuses on developing a predictive framework for personalized autism therapy recommendations. The scope includes analyzing various factors influencing therapy effectiveness, such as patient demographics, cognitive assessments, therapy duration, social engagement, and caregiver-reported progress. The system leverages machine learning techniques, including ensemble models and time-series forecasting, to enhance therapy selection and optimize treatment outcomes.

The scope is limited to therapy prediction and does not encompass direct medical diagnosis or intervention. The model’s accuracy depends on the quality and diversity of input data, which may introduce biases. Additionally, external factors such as environmental influences and emotional well-being, which impact therapy outcomes, may not be fully captured.

CHAPTER 2

CHAPTER 2
LITERATURE SURVEY

2.1 INTRODUCTION

Autism Spectrum Disorder (ASD) is a highly complex and individualized neurodevelopmental condition, where therapy outcomes can vary significantly from one person to another. Despite the availability of various therapeutic interventions, there is no standardized method to predict which therapy will be most effective for a specific individual. Current treatment approaches often rely on trial-and-error, leading to delays in progress, increased costs, and emotional strain for families. Additionally, the absence of real-time monitoring and data-driven decision-making tools limits the ability of healthcare professionals to adjust therapy plans dynamically based on a patient's evolving needs. The lack of personalized, adaptive, and explainable systems for therapy recommendation represents a critical gap in autism care. Therefore, there is an urgent need for an intelligent platform that can analyze historical and current patient data to predict therapy effectiveness and provide personalized, transparent recommendations to improve treatment outcomes for individuals with ASD.

2.2 LITERATURE SURVEY

Clara Lucato dos Santos et al., (2024) a comprehensive analysis of various behavioral interventions for individuals with Autism Spectrum Disorder (ASD). It examines evidence-based therapies such as Applied Behavior Analysis (ABA). This systematic review followed the PRISMA guidelines. The databases MEDLINE, Embase, CENTRAL (Cochrane), and Lilacs were accessed, and gray and manual searches were performed. The search strategy was created with terms referring to autism and behavioral therapy. The studies were assessed qualitatively. Randomized clinical trials and observational studies demonstrated improvements in cognitive and verbal components of patients who received behavioral therapies in therapeutic settings. These results indicate a positive impact of both cognitive-behavioral therapy and ESDM on the development of patients' skills.

Lei Qin, Haijiao Wang et al., (2024) provides the definition and characteristics of ASD, epidemiological profile, early research and diagnostic history, etiological studies, advances in diagnostic methods, therapeutic approaches and intervention strategies, social and educational integration, and future research directions. The highly heritable nature of ASD, the role of environmental factors, genetic–environmental interactions, and the need for individualized, integrated, and technology-driven treatment strategies are emphasized. Also discussed is the interaction of social policy with ASD research and the outlook for future research and treatment, including the promise of precision medicine and emerging biotechnology applications. They point out that despite the remarkable progress that has been made, there are still many challenges to the comprehensive understanding and effective treatment of ASD, and interdisciplinary and cross-cultural research and global collaboration are needed to further deepen the understanding of ASD and improve the quality of life of patients.

Sheril Sophia et al., (2023) examines the recent progress in early ASD detection through the utilization of multimodal deep learning techniques. The analysis revealed that integrating multiple modalities, including neuroimaging, genetics, and behavioral data, is key to achieving higher accuracy in early ASD detection. It is also evident that, neuroimaging data holds promise and has the potential to contribute to higher accuracy in ASD detection. Among various models used, CNN, DNN, GCN, and hybrid models have exhibited encouraging outcomes in the early detection of ASD.

Leonard Abbeduto et al., (2021) address issues across a wide portion of the lifespan and include tests of the efficacy of specific interventions, methods for measuring treatment efficacy in terms of behavioral symptoms and underlying neurobiological mechanisms, the identification of potential new treatment targets and approaches to treatment, and frameworks for determining who should receive particular treatments and when. And, more importantly, the considerable progress being made toward reducing the disabilities and challenges that are faced by many autistic individuals.

Junxia Han et al., (2022) investigates from internal neurophysiological and external behavior perspectives simultaneously and proposes a new multimodal diagnosis framework for identifying ASD in children with fusion of electroencephalogram (EEG) and eye-tracking (ET) data. Specifically, we designed a two-step multimodal feature learning and fusion model based on a typical deep learning algorithm, stacked denoising autoencoder (SDAE). In the first step, two SDAE models are designed for feature learning for EEG and ET modality, respectively. Then, a third SDAE model in the second step is designed to perform multimodal fusion with learned EEG and ET features in a concatenated way. Our designed multimodal identification model can automatically capture correlations and complementarity from behavior modality and neurophysiological modality in a latent feature space, and generate informative feature representations with better discriminability and generalizationfor enhanced identification performance.

Rui Yang et al., (2021) explored three new models (2D CAM, 3D CAM and 3D Grad-CAM) are proposed for structural Magnetic Resonance Imaging (sMRI) data. The Regions Of Interest (ROI) of subcortical tissues among models and between groups are analyzed based on the heat maps of the three models. The experimental results show that these models mainly distinguish the autism group and the control group according to the voxel value of these ROIs. There are significant differences in mean voxel value and standard deviation of voxel value between the autism group and the control group, such as in the left amygdala, optic chiasm and right hippocampus. According to medical references, these ROIs are closely related to people's speech, cognition and behavior. This can partly explain why autistic patients have unusual symptoms such as speech communication disorder, stereotyped repetitive behavior and so on. The proposed visualization models can provide a good bridge for doctors to understand the brain features learned by the neural network.

Manu Kohli et al., (2022) evaluated the role of technology in ASD detection by answering four research questions analyzing the evolution of technology, use of various bio-behavioral data sources, demographic categories, databases, controls, comparators, and assessment instruments, and data collection, processing, and outcomes of the technology-based methods in ASD detection. The review highlighted the extensive use of machine learning (ML) and Deep Learning (DL) to detect infants (as young as 9 to 12 months) at risk of ASD and Other developmental delays (ODD) using multimodal structured and unstructured data. However, the review reported various internal and external validity threats. Conclusion: Technology can significantly improve the current ASD detection process. The validation and adoption of technology can be fast-tracked by designing robust study protocols, executing multi-cultural field trials, standardizing datasets, data quality, and feature engineering methods.

Mingzhi Wang et al., (2023) constructs a whole-brain functional connectivity network based on functional MRI data, uses non-imaging data with demographic information to complement the classification task for diagnosing subjects, and proposes a multimodal and across-site WL-DeepGCN-based method for classification to diagnose autism spectrum disorder (ASD). This method is used to resolve the existing problem that deep learning ASD identification cannot efficiently utilize multimodal data. In the WL-DeepGCN, a weight-learning network is used to represent the similarity of non-imaging data in the latent space, introducing a new approach for constructing population graph edge weights, and we find that it is beneficial and robust to define pairwise associations in the latent space rather than the input space. They proposed a graph convolutional neural network residual connectivity approach to reduce the information loss due to convolution operations by introducing residual units to avoid gradient disappearance and gradient explosion.

Sweta Jain et al., (2023) explored deep Convolutional Neural Network (CNN) with Dwarf Mongoose optimized Residual Network (DMResNet) for the classification of autism disorder from Magnetic Resonance Imaging (MRI) brain images. Initially, the input brain images are preprocessed to remove the non-brain tissues. The preprocessed images are segmented with hybrid Fuzzy C Means (FCM) and Gaussian Mixture Model (GMM) which partition the image into sub groups to make it easier for classification by reducing the complexity. FCMGMM segments the volume into predefined cortical and sub cortical regions. After segmentation, the features are extracted with Visual Geometry Group (VGG)-16 networks which comprised of several tiny kernels with filters for enhancing the depth of network and permit to extract complicated and discriminative features. Region of Interest (ROI) based functional connectivity feature is extracted with VGG-16 and these features are classified with DM optimized ResNet.

Sheyan Yu et al., (2021) conducted a comprehensive analysis on the qualities of authenticity, immersion, and interactivity brought by VR and explores the way how it increases the trainer's probability of transforming their scenaior's experience and the skills they have developed into an actual situation in daily life, making the therapy effective, practical, and credible assessment possible. By examining the effect of VR treatment from the practical example and designing a VR scenario, the results show that VR greatly enhances the effectiveness and application potential of behavioral corrections and other symptoms.

Matteo Cordioli et al., (2023) presented a framework to assist therapists and children with autism spectrum disorder in their Applied Behavioral Analysis (ABA) therapy. The framework was designed in collaboration with Spazio Autismo, an autism center in Mantova, Italy. The framework is a first step toward transitioning from the current paper-based to fully digital-supported therapy. We evaluated the framework over four months with 18 children diagnosed with classic autism, ranging from 4 to 7 years old. The framework integrates a mobile app that children and therapists use during the sessions with a backend for managing therapy workflow and monitoring progress. Our preliminary results show that the framework can improve the efficacy of the therapy sessions, reducing non-therapeutic time, increasing patient focus, and quickening the completion of the assigned objectives. It can also support therapists in preparing learning materials, data acquisition, and reporting. Finally, the framework demonstrated improved privacy and security of patients' data while maintaining reliability.

Sankar Ganesh Karuppasamy et al., (2022) explores artificial intelligence (AI) technology to help them implement computerized diagnosis and rehabilitation processes. Neuroimaging-based approaches have been the focus of deep learning algorithms for ASD diagnosis. Neuroimaging techniques are benign disease indicators that could aid in identifying ASD. Neuroimaging procedures, both structural and functional, give doctors much information about the brain's anatomy and activity. Because of the brain's complex structure and function, developing optimal processes for ASD identification using neuroimaging data deprived of using Deep Learning is difficult. Our proposed work aims to identify Autism Spectrum Disorder(ASD) from a huge dataset based on brain patterns. Using a convolution neural network, the proposed work identifies ASD patients from ordinary people. It identifies the ROI using the feature extraction technique. The system performance is measured by accuracy, and it achieves 95% accuracy to identify ASD patients.

Intissar Salhi et al., (2022) explored interest in the potential of social robots in the education for learners with autism spectrum disorders. They have complex needs, possibly with additional intellectual disabilities and/or limited oral communication, requiring much more assistance from specialized teachers and paramedical aides than regular schools can usually provide. In this context, our research focuses on robot-assisted therapy and introduces a humanoid social robot in inclusive education. This is achieved by analyzing many aspects of an Autistic behavior, such as verbal interactions, gestures and facial expressions, etc. Therefore, a robot can reproduce consistent experiences and actions for learners with limited communication skills. This is realized by applying a new approach based on the selection of the appropriate therapy. This research is a preliminary study towards the development of a humanoid robot that will assist therapists in recommending a set of therapies using a matrix of correspondence between screening tests and therapies, and in assisting and supervising autistic learners by performing the tasks that are necessary to accomplish prescribed therapies.

Minxiao Wang et al., (2022) presented an Observational Therapy-Assistance Neural Network (OTA-NN) for bridging the observation ability gap between experienced therapists and unprepared therapists (include parents of children). In OTA-NN, an attention based neural network extracts features from children's behavior data and a multiple instance learning (MIL) network predicts score for reflecting child's training state during a therapy session. We adopt weakly-supervised learning to train our model. The experiments on DREAM dataset show our model can successfully distinguish children's unpredictable behaviors under different training states.

Sandeep Vemuri et al., (2023) employed a ResNet50 CNN model to analyze facial characteristics and identify distinctions between children with ASD and typically developing children. The model was trained on a comprehensive dataset of facial images and evaluated based on its accuracy in distinguishing between the two groups. The potential application of this model lies in aiding early detection and intervention, leading to improved outcomes for children with ASD. In the study, two deep learning models were considered: EfficientNet and ResNet50. These models were trained on the dataset prepared for the experiment. By comparing their performances and results, it was found that both models were effective in detecting differences; however, the EfficientNet model exhibited higher accuracy and parameter efficiency compared to the ResNet50 model. For future research, potential directions include exploring video-based prediction of autism and evaluating the different phases or stages of autism.

Yarlagadda Bhargavi et al., (2023) proposed a work where the system compares an input image with a trained dataset that is stored in the database to determine the emotional state of an image. A complicated neurodevelopmental illness called autism spectrum disorder (ASD) impacts communication, language, and social skills. Early identification of ASD individuals, particularly in children, would be advantageous to build and strategize the proper therapeutic strategy at the precise time. A person with autism spectrum disorders can be recognized by looking at their face, making eye contact and other facial features. Utilizing the training datasets and the deep learning models, the proposed method will be trained, tested and evaluated. The outcome of the proposed framework will be more accurate on autism face recognition based on neural networks and implementing the robot for monitoring.

Nevena Ackovska et al., (2023) presented a case in which a robot therapy for children with autism was transferred from clinic to home conditions. The developed application enables the children to continue with the interventions in home conditions. This proved especially important in the COVID-19 pandemic. The application also allows monitoring of the child's activities, through which the therapist can later analyze the patient's behavior and offer appropriate therapy. The application shows reliable results and gives promise to develop beyond the user case we are considering.

Y Srikar et al., (2023) explored machine learning models and their potential to understand autism and provide more effective interventions for autistic patients. However, learning models are particularly challenging to read, making it challenging to understand how the model makes decisions and identify any biases in the model. These are important in the context of research on autism, where it is possible to assess the effects of inaccurate diagnosis or improper treatment recommendations. In their work, various algorithms are applied to an autism spectrum disorder (ASD) dataset with 931 samples and 103 features. Experimental results show that the predictive model Random Forest gives the highest accuracy, precision, recall, and Fl-Score with 100% compared to LR, GNB, KNN, SVM, and ID3.

Furong Sun et al., (2021) reviews the literature in recent years, and discusses the prediction, early recognition and diagnosis of autism, as well as the research on auxiliary intervention. The application of artificial intelligence in the medical field provides broad prospects for the timeliness and accuracy of ASD diagnosis and the improvement of its core symptoms. The problems of ASD rehabilitation caused by insufficient medical resources and high treatment costs are expected to be solved by the intervention of artificial intelligence. Artificial intelligence (AI) is becoming more and more prominent in the efficient, high-precision and high-stability clinical diagnosis of autism.

Erwin Halim et al., (2022) conducted a study where forty ASD participants ranging from the age of 7 to 26 will be divided into two groups of age, which are children and young adults. In total there will be 10 sessions of virtual-based intervention in which the ASD participants will be guided by a coach and a researcher. Each of session has its own social objectives that need to be accomplished by the participants. There will be a pre-test and post-test to measure the progress of each of the participants' social skills and cognition. This research will be conducted for approximately three months in the hope that virtual reality can be deemed a useful therapy for ASD individuals.

Chang Ze Wu et al., (2021) explored the application status of virtual reality technology in the treatment of autism, including social skills, motor skills, attention training, and daily life, and further expand in detail, and finally point out the current shortcomings of virtual reality technology and prospects for the future. Research methods: Literature on the application of virtual reality technology in the treatment of autism in recent years was searched through literature navigation. The children's motor skills, social interaction, daily life, and casual attention were compared with virtual reality technology.

2.3 CONCLUSION

The reviewed literature highlights the growing integration of technology, artificial intelligence, and behavioral therapies in the detection, diagnosis, and intervention of Autism Spectrum Disorder (ASD). Various approaches have been explored, including multimodal deep learning models, neuroimaging analysis, EEG-based systems, and behavior-focused therapies like Applied Behavior Analysis (ABA) and the Early Start Denver Model (ESDM). Technological advancements such as virtual reality environments, social robotics, and AI-driven diagnostic frameworks have shown promising results in improving the accuracy of early detection and enhancing the effectiveness of therapeutic interventions. Additionally, the studies underscore the importance of personalized care models and interdisciplinary collaboration in addressing the diverse needs of individuals with ASD.

However, challenges such as model interpretability, data heterogeneity, cultural validation, and the need for scalable, real-time solutions persist. Future research should focus on:

- Enhancing the interpretability and transparency of AI-driven ASD detection models to support clinical decision-making.
- Standardizing and diversifying datasets to improve the generalizability of diagnostic tools across different populations and age groups.
- Developing real-time, adaptive intervention systems that can personalize therapy based on individual behavioral and neurological responses.
- Integrating virtual reality, robotics, and AI into hybrid therapeutic frameworks for more engaging and accessible interventions.
- Promoting interdisciplinary research combining neuroscience, psychology, data science, and engineering to build comprehensive, evidence-based solutions for ASD care.

CHAPTER 3

CHAPTER 3 SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

In existing system addressing therapy prediction and management for Autism Spectrum Disorder (ASD), basic machine learning techniques have been applied, such as Decision Trees and Logistic Regression. These models are used independently to analyze patient data and predict potential therapy outcomes. However, they often suffer from limited accuracy and generalization due to their inability to effectively handle the complex and heterogeneous nature of ASD. Most of these systems rely on single-model approaches, which may not capture the full variability and depth of multi-dimensional therapy response data. Additionally, these models are often not integrated into real-time or adaptive therapy recommendation platforms, and they typically lack the ability to learn from sequential data over time. As a result, existing systems do not offer dynamic, personalized, or transparent therapy recommendations, highlighting the need for a more robust, intelligent solution.

3.1.1 PROBLEM DEFINITION

- Cannot handle the high variability and complexity in therapy responses among individuals with ASD.
- Lacks adaptability; treatment plans remain static and do not adjust based on patient development.
- Often requires manual interpretation and decision-making by doctors, increasing workload and reducing efficiency.

3.1 PROPOSED SYSTEM

The proposed system aims to revolutionize therapy recommendation and monitoring for individuals with Autism Spectrum Disorder (ASD) by leveraging advanced machine learning and deep learning techniques. Unlike existing systems that rely on single-model approaches, this system uses an ensemble model combining XGBoost, Support Vector Machine (SVM), and Random Forest to enhance prediction accuracy and robustness. Additionally, a Long Short-Term Memory (LSTM) model is integrated to analyze sequential therapy progress data, enabling dynamic and time-aware therapy recommendations. The system features a user-friendly web platform with dedicated modules for patients and doctors, allowing patients to log their therapy progress and enabling doctors to monitor outcomes and receive intelligent, personalized therapy suggestions. Explainable AI (XAI) is incorporated to ensure that all predictions are transparent and interpretable, fostering trust among healthcare professionals and families. By providing continuous monitoring, adaptive therapy plans, and real-time decision support, the proposed system addresses the limitations of current approaches and promotes precision-based, individualized autism care.

ADVANTAGE

- Provides transparent and interpretable model outputs for better decision-making.
- Reduces trial-and-error in therapy planning by relying on intelligent data analysis.
- Uses ensemble models (XGBoost, SVM, Random Forest) to enhance therapy prediction performance.

CHAPTER 4

CHAPTER 4 REQUIREMENT SPECIFICATIONS

4.1 INTRODUCTION

One of the distinctive aspects of this research is the use of feature selection techniques to pinpoint the most influential patient-specific and therapy-related factors that significantly improve the accuracy of therapy outcome predictions while also optimizing the system's execution time. The primary goal of developing this model is to offer a consistent, evidence-based framework for assessing therapy effectiveness in children with Autism Spectrum Disorder (ASD). This document is a critical component of the Software Development Life Cycle (SDLC), detailing the complete functional, technical, and system requirements for the Therapy Efficacy Model (TEM). It acts as a reference guide for developers, data scientists, and other project contributors throughout the phases of design, implementation, testing, and evaluation. Any updates or revisions to the system's specifications in the future will be managed through a structured change management and approval process to ensure the reliability and consistency of the model.

Functional Requirements:

Input: The primary input for the **Therapy Efficacy Model (TEM)** is a comprehensive, pre-processed dataset comprising detailed information about children diagnosed with **Autism Spectrum Disorder (ASD)**. This dataset includes a diverse range of patient-specific attributes such as age, gender, initial ASD severity levels (measured through tools like ADOS or CARS), and baseline cognitive, social, and communication skill scores. In addition to these patient-centered factors, therapy-related parameters such as the type of therapy administered (for example, Applied Behavior Analysis, Speech Therapy, or

Occupational Therapy), therapy duration, frequency, and intensity are also incorporated.

Output: The primary output is the predicted therapy recommendation, where the system suggests the most suitable therapy type for a patient based on their personal attributes, severity of symptoms, and previous therapy outcomes. Alongside this, the system maintains a comprehensive patient history record, which includes demographic details, assessment scores, therapy sessions attended, and progress over time. This enables therapists to track improvements or setbacks in various developmental areas, such as communication, social interaction, and behavioral patterns. Furthermore, the system records and displays the doctor's recommendations and notes for each patient, ensuring that personalized medical advice and clinical observations are stored securely and can be referenced in future consultation.

4.2 HARDWARE AND SOFTWARE SPECIFICATION

4.2.1 HARDWARE REQUIREMENTS

➢ Processor	-	i3, i5, i7
➢ Speed	-	1.1 GHz
➢ RAM	-	8 GB
➢ Hard Disk	-	500 GB

4.2.2 SOFTWARE REQUIREMENTS

➢ Operating System	-	Windows 7/8/10
➢ Front End	-	HTML, CSS

- Language - Python
- Tool - PyCharm

4.2.2.1 PYTHON

Python could be an interpreted, high-level and general programing language. Python is a style philosophy emphasizes code readability with its notable use of great indentation. Its language constructs and object-oriented approach aim to assist programmers write clear, logical code for tiny and large- scale projects. Python is commonly represented as A battery enclosed language because of its comprehensive commonplace library. Python is a multi-paradigm programming language, Object-oriented programming and structured programming language are absolutely supported and plenty of its options support practical programming and aspect-oriented programming several paradigms are supported via extensions, together with design by contract and logic programming. Python uses dynamic writing and a mix of reference numeration and a cycle-detecting dustman for memory management. It also options dynamic name resolution (late binding), that binds technique and variable names throughout program execution.

Python is an interpreter, object-oriented, high-level programing language with dynamic semantics. Its high-level inbuilt information structures, combined with dynamic typing and dynamic binding build it terribly engaging for fast Application Development, still as to be used as a scripting or glue language to attach existing parts together. Python' simple, straightforward to find out syntax emphasizes readability and so reduces the cost of program maintenance. Python supports modules and packages, which inspires program modularity and code reuse. The Python interpreter and also the intensive commonplace library are on the market in supply or binary type at no cost for all major platforms and may be freely distributed. Often, programmers fall soft on with Python thanks to the magnified productivity it provides. Since there's no compilation step, the edit- test-debug cycle is implausibly fast. Debugging Python programs is easy, a bug or dangerous input can ne'er cause a segmentation fault. Instead, once the interpreter discovers an error, it raises an exception. once the program doesn't catch the exception, the interpreter prints a stack trace. A supply level computer programme permits examination of native and world variables, analysis of discretional expressions, setting breakpoints, stepping through the code a line at a time, and then on.

Python is a dynamic programming language which supports several different programming paradigms as follows:

- Procedural programming
- Object oriented programming
- Functional programming

  Standard: Python byte code is executed in the Python interpreter (similar to Java) platform independent code

- Syntax is clear, easy to read and learn (almost pseudo code)
- Common language
- Intuitive object-oriented programming
- Full modularity, hierarchical packages
- Comprehensive standard library for many tasks
- Big community
- Simply extendable via C/C++, wrapping of C/C++ libraries
- Focus: Programming speed

4.2.2.2 INTRODUCTION TO PYTHON

Guido van Rossum began functioning on Python within the late 1980s, as a successor to the fundamental principle programming language, and initial discharged it in 1991 as Python 0.9.0. Python two.0 was released in 2000 and introduced new features, akin to list comprehensions and a trash collection system victimization reference count and was discontinued with version 2.7.18 in 2020.

Python 3.0 was released in 2008 and was a serious revision of the language that's not fully backward-compatible and far Python 2 code doesn't run unqualified on Python 3. Python 2.0 was released on sixteen Gregorian calendar month 2000, with several major new options, together with a cycle-detecting refuse collector and support for Unicode.

Python three.0 was discharged on 3 Gregorian calendar month two008. it had been a serious revision of the language that's not fully backward-compatible. several of its major features were backported to Python 2.6.x and 2.7.x version series. Releases of Python 3 embody the 2to3 utility, that automates (at least partially) the interpretation of Python 2 code to Python 3. Python 2.7's end-of- life date was at the start set at 2015 then delayed to 2020 out of concern that an outsized body of existing code couldn't simply be forward-ported to Python 3. No a lot of security patches or different enhancements are going to be discharged for it. With Python 2's end-of-life, solely Python 3.6.x and later are supported. Python 3.9.2 and 3.8.8 were speeded up as all versions of Python (including 2.7) had security issues, resulting in potential remote code execution and internet cache poisoning.

DESIGN PHILOSOPHY AND FEATURES

The standard library has 2 modules (itertools and functools) that implement practical tools borrowed from Haskell and normal ML. instead of having all of its practicality designed into its core, Python was designed to be extremely extensile (with modules). This compact modularity has created it significantly standard as a method of adding programmable interfaces to existing applications. Python attempts for a simpler, less-cluttered syntax and synchronic linguistics whereas giving developers a alternative in their writing methodology. Python developers strive to avoid premature optimization, and reject patches to non-critical elements of the CPython reference implementation that may provide marginal will increase in speed at the price of clarity. once speed is important, a

Python technologist will move time-critical functions to extension modules written in languages reminiscent of C, or use PyPy, a just- in-time compiler. Cython is additionally available, that interprets a Python script into C and makes direct C-level API calls into the Python interpreter.

SYNTAX AND SEMANTICS, INDENTATION

Python is supposed to be a simply clear language. Its format is visually uncluttered, and it usually uses English keywords wherever different languages use punctuation. not like several other languages, it doesn't use crisp brackets to delimit blocks, and semicolons once statements are allowed however are rarely, if ever, used. it's fewer syntactical exceptions and special cases than C or Pascal. Python uses whitespace indentation, instead of curly brackets or keywords, to delimit blocks.

TYPING

Python makes use of duck typing and has typed gadgets however untyped variable names. Type constraints aren't checked at assemble time; rather, operations on an item may also fail, signifying that the given item isn't always of a appropriate type. Despite being dynamically-typed, Python is strongly-typed, forbidding operations that aren't well-defined (for example, including a range of to a string) in preference to silently trying to make feel of them. Python permits programmers to outline their very own sorts the use of training, which can be most usually used for item-orientated programming. New times of training are built through calling the class (for example, SpamClass () or EggsClass ()), and the training are times of the metaclass type (itself an example of itself), permitting metaprogramming and reflection.

LIBRARIES

Python’s giant standard library, usually cited united of its greatest strengths, provides tools suited to several tasks. For Internet-facing applications, many standard formats and protocols corresponding to MIME and hypertext transfer protocol are supported. It includes modules for making graphical user interfaces, connecting to relative databases, generating pseudorandom numbers, arithmetic with arbitrary- precision decimals, manipulating regular expressions, and unit testing. Some components of the quality library are lined by specifications (for example, the net Server entranceway Interface (WSGI) implementation wsgiref follows life 333), however most modules are not. they're given by their code, internal documentation, and check suites. However, as a result of most of the quality library is cross-platform Python code, solely a number of modules want sterilization or redaction for variant implementations. As of March 2021, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 290,000 packages with a large vary of functionality, including: Automation

- Data analytics
- Databases
- Documentation
- Graphical user interfaces
- Image processing
- Machine learning
- Mobile App
- Multimedia
- Computer Networking

DEVELOPMENT ENVIRONMENTS

Most Python implementations (including CPython) embody a read–eval–print loop (REPL), allowing them to perform as a instruction interpreter that the user enters statements consecutive and receives results immediately. Different shells, as well as IDLE and IPython, add further skills like improved auto-completion, session state retention and syntax highlighting. still as normal desktop integrated development environments, there are internet browser-based IDEs; Sage science (intended for developing science and math-related Python programs); PythonAnywhere, a browser-based IDE and hosting environment; and cover IDE, an advertisement Python IDE action scientific computing.

4.2.2.3 ANACONDA NAVIGATOR - JUPYTER NOTEBOOK

Anaconda may be a distribution of the Python associate degreed R programming languages for scientific computing (data science, machine learning applications, large-scale knowledge processing, prophetical analytics, etc.), that aims to change package management and deployment. The distribution includes data-science packages appropriate for Windows, Linux, and macOS. it's developed and maintained by boa, Inc., that was based by Peter Wang and Travis Oliphant in 2012. As a boa, Inc. product, it is additionally referred to as boa Distribution or boa Individual Edition, whereas different merchandise from the corporate are boa Team Edition and Anaconda Enterprise Edition, each of which aren't free. The subsequent applications are on the market by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QtConsole
- Spyder

- Glue
- Orange
- RStudio
- Visual Studio Code

Jupyter Notebook (formerly IPython Notebooks) may be a net-based interactive machine setting for making Jupyter notebook documents. The "notebook" term will informally build respect to many alternative entities, chiefly the Jupyter web application, Jupyter Python web server, or Jupyter document format counting on context. A Jupyter Notebook document is a JSON document, following a versioned schema, containing an ordered list of input/output cells which might contain code, text (using Markdown), mathematics, plots and made media, sometimes ending with the ".ipynb" extension. A Jupyter Notebook will be born-again to variety of open commonplace output formats (HTML, presentation slides, LaTeX, PDF, Restructured Text, Markdown, Python) through "Download As" within the net interface, via the nbconvert library or "jupyter nbconvert" command interface during a shell. To change visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publically on the market notebook document, convert it to markup language on the fly and show it to the user.

CHAPTER 5

CHAPTER 5

SYSTEM DESIGN

5.1 ARCHITECTURE DIAGRAM

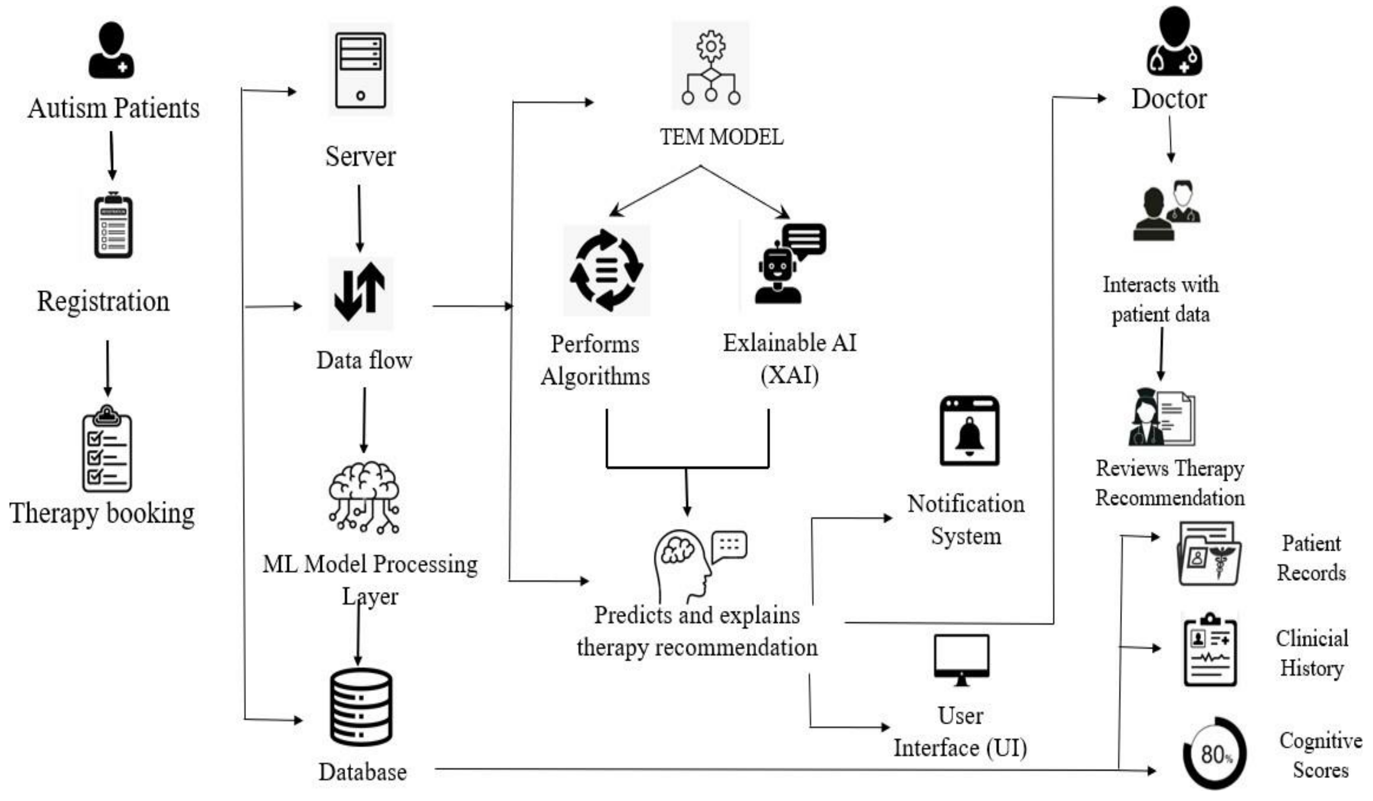


Fig 1 Architecture Diagram

5.1 UML DIAGRAMS

USECASE

DIAGRAM

A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases. A Use Case describes a sequence of actions that provided something of unmeasurable value to an actor and is drawn as a horizontal ellipse. An actor is a person, organization or external system that plays a role in one or more interaction with the system.

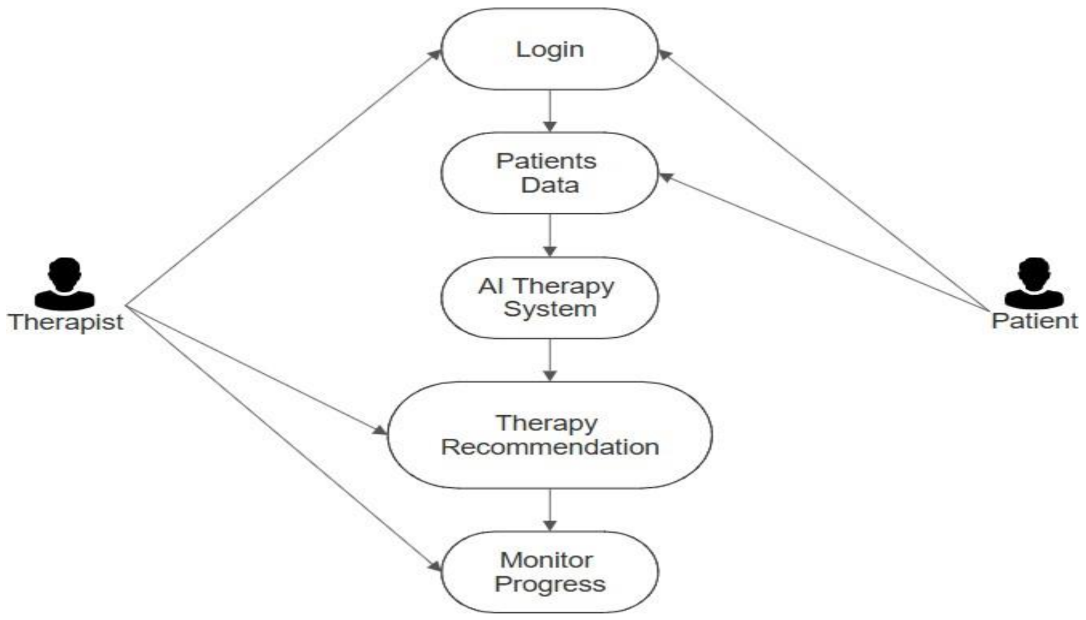


Fig 2 Use case Diagram

5.1.1 SEQUENCE DIAGRAM

A Sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence diagrams are sometimes called event diagrams, event sceneries and timing diagram.

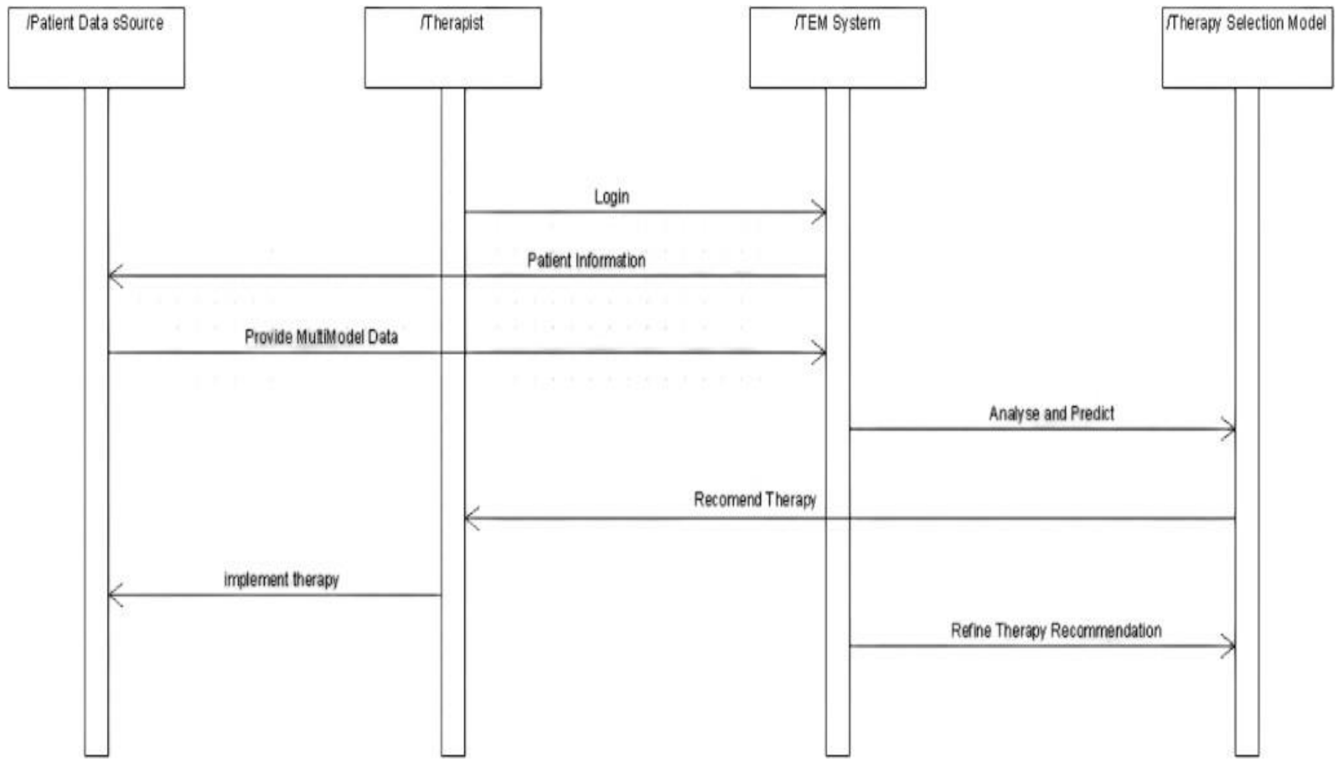


Fig 3 Sequence Diagram

5.1.2 CLASS DIAGRAM

A Class diagram in the Unified Modelling Language is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

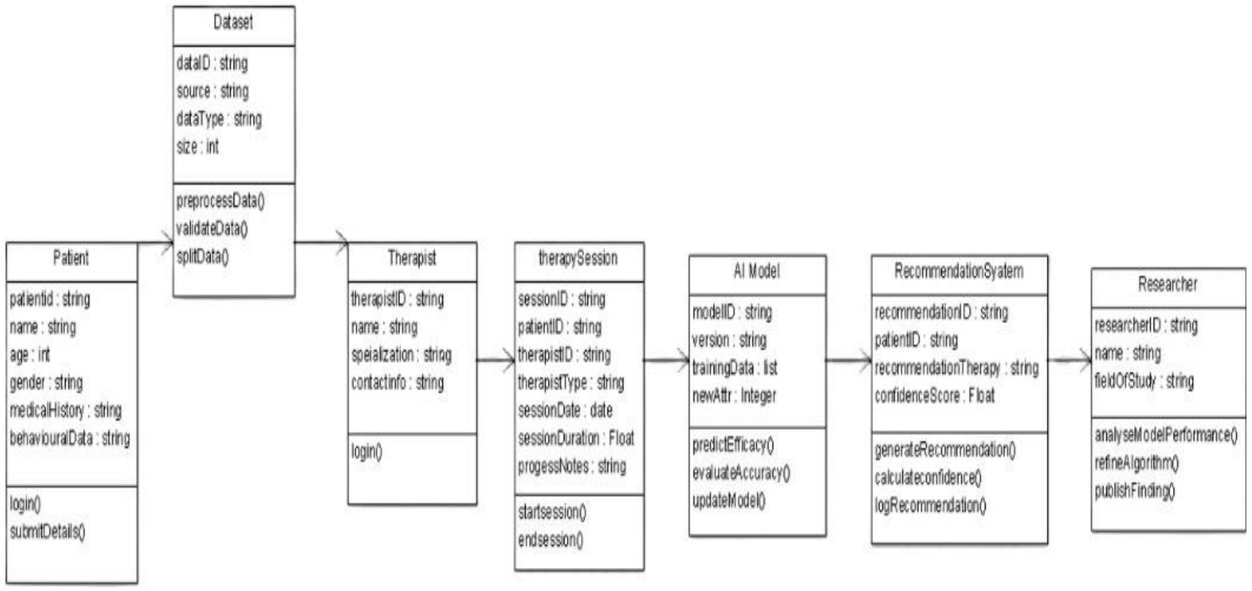


Fig 4 Class Diagram

5.1.3 ACTIVITY DIAGRAM

Activity diagram is a graphical representation of workflows of stepwise activities and actions with support for choice, iteration and concurrency. An activity diagram shows the overall flow of control.

- Rounded rectanglesrepresent activities.
- Diamonds representdecisions.
- Bars represent the start or end of concurrent activities.
- An encircled circle represents the end of the workflow.
- A black circle represents the start of the workflow

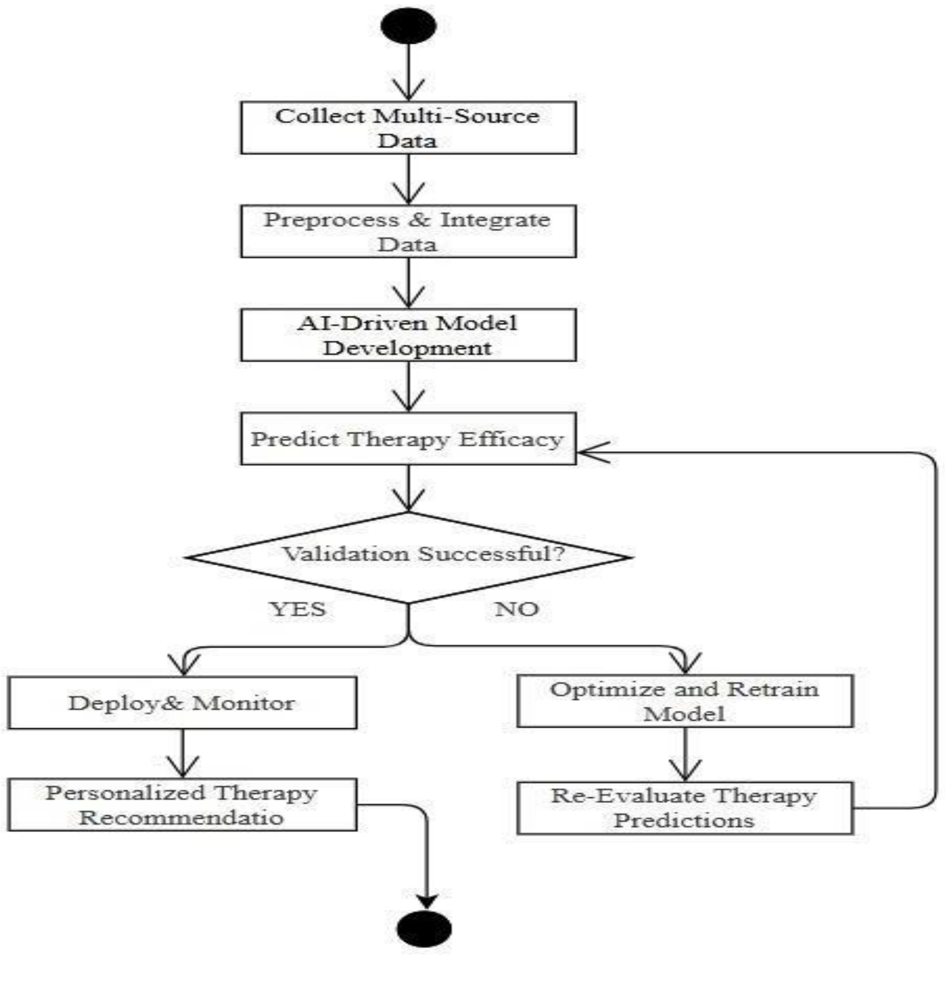
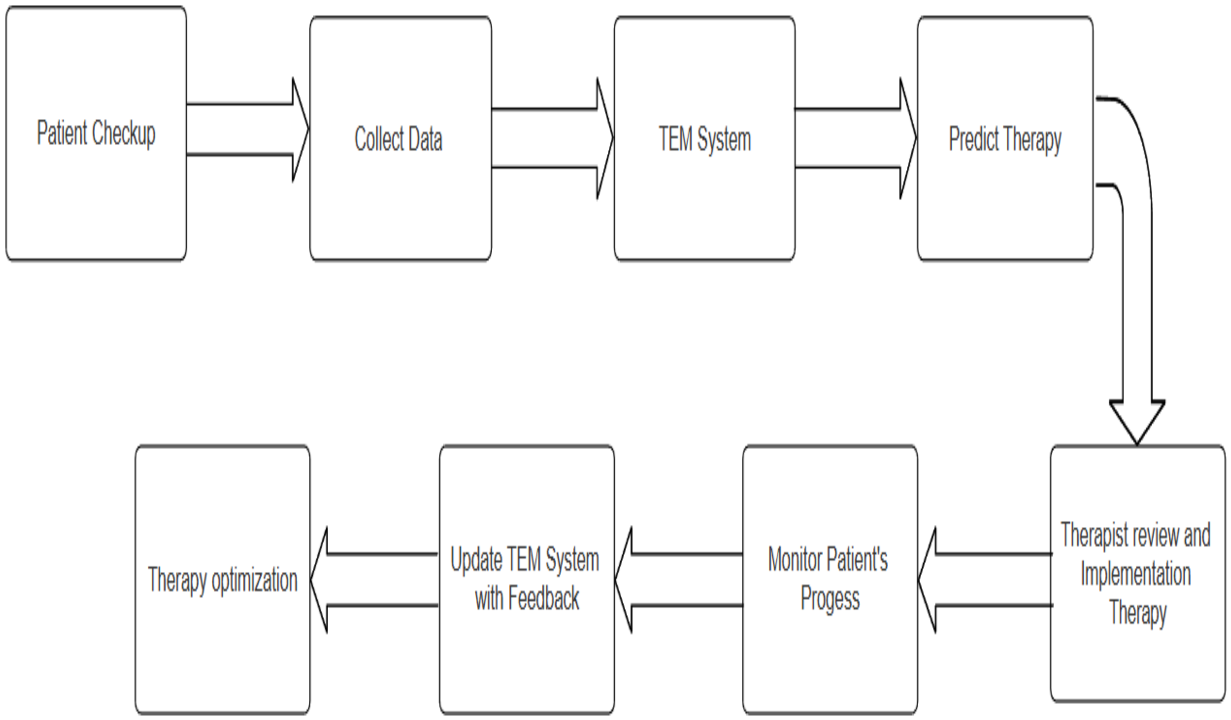

Collect Multi-Source Data
Preprocess & Integrate Data
AI-Driven Model Development
Predict Therapy Efficacy
Validation Successful?
YES
NO
Deploy& Monitor
Optimize and Retrain Model
Personalized Therapy Recommendatio
Re-Evaluate Therapy Predictions

Fig 5 Activity Diagram

5.1.1 WORK FLOWDIAGRAM

A workflow diagram (also known as a workflow) provides a graphic overview of the business process. Using standardized symbols and shapes, the workflow shows step by step how your work is completed from start to finish.



Patient Checkup
Collect Data
TEM System
Predict Therapy
Therapist review and Implementation Therapy
Monitor Patient's Progess
Update TEM System with Feedback
Therapy optimization

Fig 6 Work Flow Diagram

CHAPTER 6

CHAPTER 6
DATA COLLECTION AND PREPROCESSING

6.1 DATA COLLECTION

The dataset for this project was collected from the Figshare website, a provider of open research repository infrastructure whose solutions help organizations and researchers share, showcase and manage their research. Specifically, the dataset contains autism patient data providing accurate information on patient description and their characteristics. The dataset was obtained from a reliable source and is publicly available for research and analysis purposes.

6.2 DATASET DESCRIPTION

The dataset consists of a structured tabular format, typically stored in a CSV (Comma Separated Values) file, containing rows and columns of autism patient data. Each row represents a unique patient entry, while each column corresponds to a specific attribute or variable related to patient characteristics and performance metrics. Key fields included in the dataset are as follows:

- Age: Age category of the patient.
- Gender: Gender of the patient.
- Level of ASD: Clinical severity of Autism Spectrum Disorder.
- Behavior: Categorizes behavioral tendencies.
- Planned Therapy: The type of therapy recommended.

Various patient clinical features such as Speak Verbally, Follow Instructions, Maintain Interaction, Socialize with Other Children, Eye Contact, Role

Playing, Facial Expression, Look at Pointed Toys, Respond When Called, Keep Attention,

6.3 FEATURE ENGINEERING

6.3.1 SELECTION OF FEATURES

In this section, we carefully select the features from the dataset that are most relevant to enhance the predictive accuracy of the therapy efficacy model. Guided by machine learning-based statistical techniques, we prioritize features that exhibit strong associations with therapy outcomes. Attributes related to communication skills, social behaviors, cognitive responses, and clinical classifications are considered essential indicators of therapy suitability. Features demonstrating high correlation with the therapy plan or parent objectives, or known to influence ASD symptom management, are retained. Redundant or low-impact features may be removed or merged to simplify model complexity and improve performance.

6.3.2 FEATURE DEFINITION

Once the features are selected, we define each feature in detail, specifying its meaning, value range, and preprocessing considerations. Beyond the basic attributes like patient age and gender, we introduce more nuanced metrics such as Eye Contact, Role Playing, Facial Expression to provide a comprehensive understanding of the patient and their autism. Let's delve into the significance and interpretation of each selected variable:

- **Age** — The age group or range in which the child is categorized, influencing therapy suitability and developmental expectations.

- **Gender** — The biological gender of the child, which may correlate with therapy responses and developmental patterns.
- **Level of ASD** — The clinically diagnosed severity level of Autism Spectrum Disorder (Level 1: mild, Level 2: moderate, Level 3: severe) defining therapy intensity needs.
- **Speak Verbally** — Indicates whether the child can communicate using spoken language, important for determining speech-based therapies.
- **Follow Instructions** — Ability to comprehend and accurately act upon verbal or non-verbal instructions provided by caregivers or therapists.
- **Maintain Interaction** — The child’s ability to engage and sustain meaningful interpersonal interactions over time.
- **Socialize with Other Children** — Reflects whether the child interacts positively or regularly with peers in social settings.
- **Eye Contact** — Measures the ability to maintain appropriate eye contact during communication, a critical aspect of social responsiveness.
- **Role Playing** — Indicates the child’s capacity and interest in participating in pretend-play or imaginative role-based games.
- **Facial Expression** — Ability to convey emotions, moods, or reactions through facial gestures and expressions.
- **Understand Others' Feelings** — The child’s awareness and empathy towards the emotions or feelings of others in their environment.
- **Look at Pointed Toys** — Assesses responsiveness to joint attention cues, specifically when someone points at an object or toy.
- **Respond When Called** — Measures consistency in responding by name or to personal addresses, a fundamental social behavior.
- **Keep Attention** — Ability to maintain focus on tasks or stimuli over sustained periods without distraction.

- **Interest in Gadgets** — Indicates the child's attraction to electronic devices or digital gadgets, which can affect therapy planning.
- **Behaviour** — Describes behavioral tendencies such as tantrums, obsessive behaviors, sensitivity issues, or hyperactivity.
- **Parents' Objective** — The specific developmental or behavioral improvements parents seek for their child through therapy interventions.
- **Planned Therapy** — The type of therapeutic intervention recommended (like occupational, behavioral, speech therapy), forming the target output of the predictive model.

These complex metrics may require additional explanation to ensure clarity and consistency in their interpretation and utilization within our analysis.

6.3.3 FEATURE ANALYSIS

After defining the features, we conduct a comprehensive analysis to understand their distribution, interrelationships, and potential influence on the target variable (Planned Therapy). Utilizing various visualizations such as histograms, bar charts, and correlation heatmaps, we explore the dataset to uncover meaningful patterns, trends, and associations among patient attributes and therapy outcomes. Furthermore, we employ machine learning-based techniques to compute feature importance scores, enabling us to identify and prioritize the most significant factors that contribute to predicting effective therapy plans for children with Autism Spectrum Disorder (ASD)

This in-depth process allows us to gain valuable insights into the patient-specific and behavioral characteristics that drive therapy recommendations. By systematically selecting, defining, and analyzing relevant features, we aim to build accurate, interpretable, and clinically useful predictive models that effectively capture the complex dynamics of therapy planning in ASD management.

6.4 DATA CLEANING AND TRANSFORMATION:

Prior to analysis, the dataset underwent a thorough data cleaning and transformation process to ensure data quality and consistency. This process involved the following steps:

- Handling Missing Values: Any missing values in the dataset were identified and addressed through techniques such as imputation or removal, depending on the impact on the analysis.
- Dealing with Outliers: Outliers in the dataset were identified and treated using appropriate methods to prevent them from skewing the results of the analysis.
- Encoding Categorical Variables: Categorical variables such as patient age and ASD levels were encoded into numerical format to facilitate modeling.
- Feature Scaling: Numerical features in the dataset were scaled to ensure uniformity and prevent variables with larger magnitudes from dominating the analysis.
- Data Validation: The integrity of the dataset was verified through validation checks to detect and correct any inconsistencies or errors in the data.
- By performing these data cleaning and transformation steps, the dataset was prepared for further analysis and modeling, ensuring the accuracy and reliability of the results obtained.

CHAPTER 7

CHAPTER 7

7. EXPLORATORY DATA ANALYSIS (EDA)

7.1 UNIVARIANT ANALYSIS

In this section, we conduct univariate analysis to explore the distribution and characteristics of individual variables within the dataset. Through techniques such as histograms, box plots, and summary statistics, we gain insights into the central tendency, dispersion, and shape of each variable's distribution. Univariate analysis provides a foundational understanding of the data and highlights any anomalies or patterns that may warrant further investigation.

Here we are exploring three main univariant analysis. They are

- Histogram - represented by a set of rectangles, adjacent to each other, where each bar represents a kind of data
- Boxplot - graph that gives a visual indication of how a data set’s 25th percentile, 50th percentile, 75th percentile, minimum, maximum and outlier values are spread out and compare to each other
- Descriptive Statistics - summarize and organize characteristics of a data set. A data set is a collection of responses or observations from a sample or entire population.

7.1.1 HISTOGRAM

A histogram is a graphical representation of the distribution of data. It Visualizes the distribution of each variable separately to understand the spread and shape of the data. The histogram is represented by a set of rectangles, adjacent to each other, where each bar represents a kind of data. Statistics is a stream of mathematics that is applied in various fields. When numerals are repeated in statistical data, this repetition is known as Frequency and which can be written in the form of a table, called a frequency distribution. A Frequency distribution can be shown graphically by using different types of graphs and a Histogram is one among them.

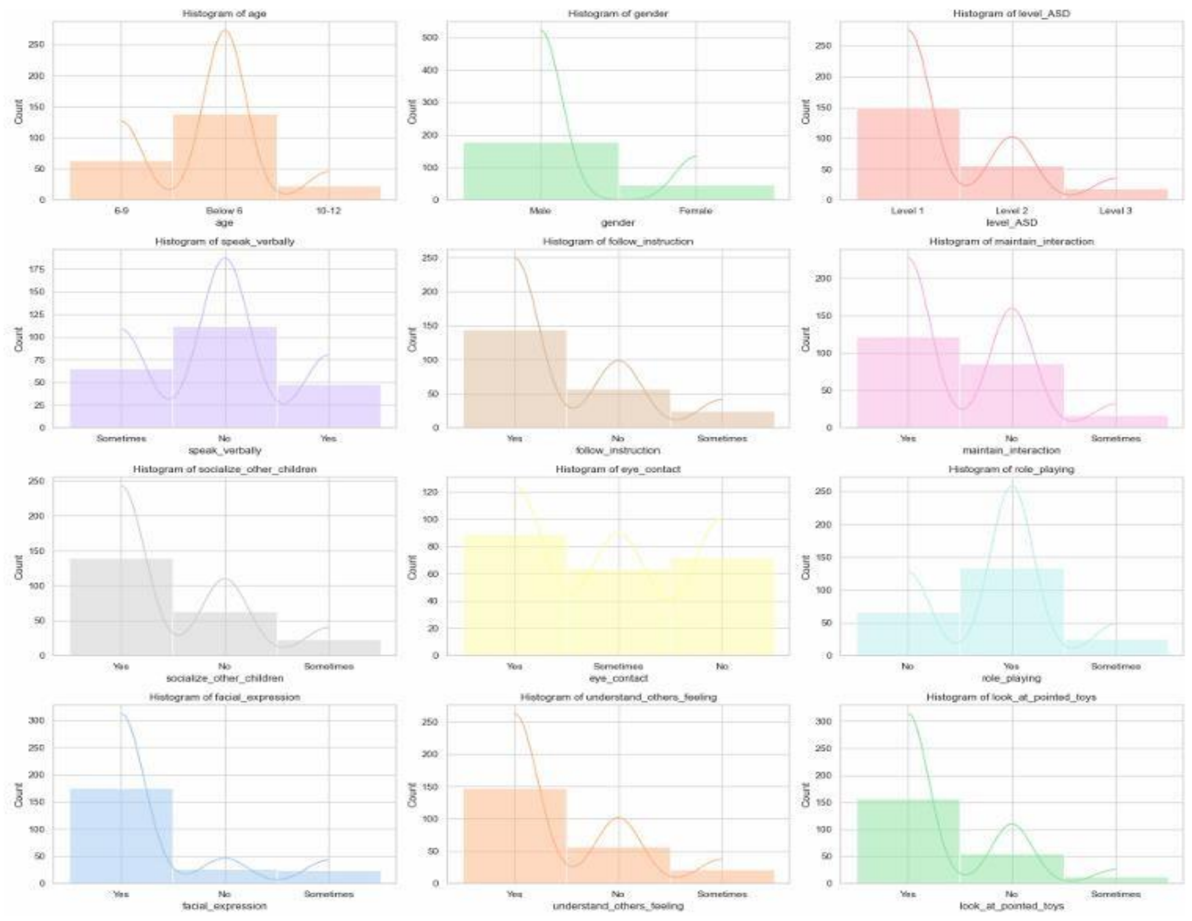


Fig 7 Histogram

7.1.2 BOXPLOT

A boxplot is a graph that gives a visual indication of how a data set’s 25th percentile, 50th percentile, 75th percentile, minimum, maximum and outlier values are spread out and compare to each other. Boxplots are drawn as a box with a vertical line down the middle, and has horizontal lines attached to each side. Display the five-number summary (minimum, first quartile median, third quartile, maximum) and identify potential outliers for each variable.

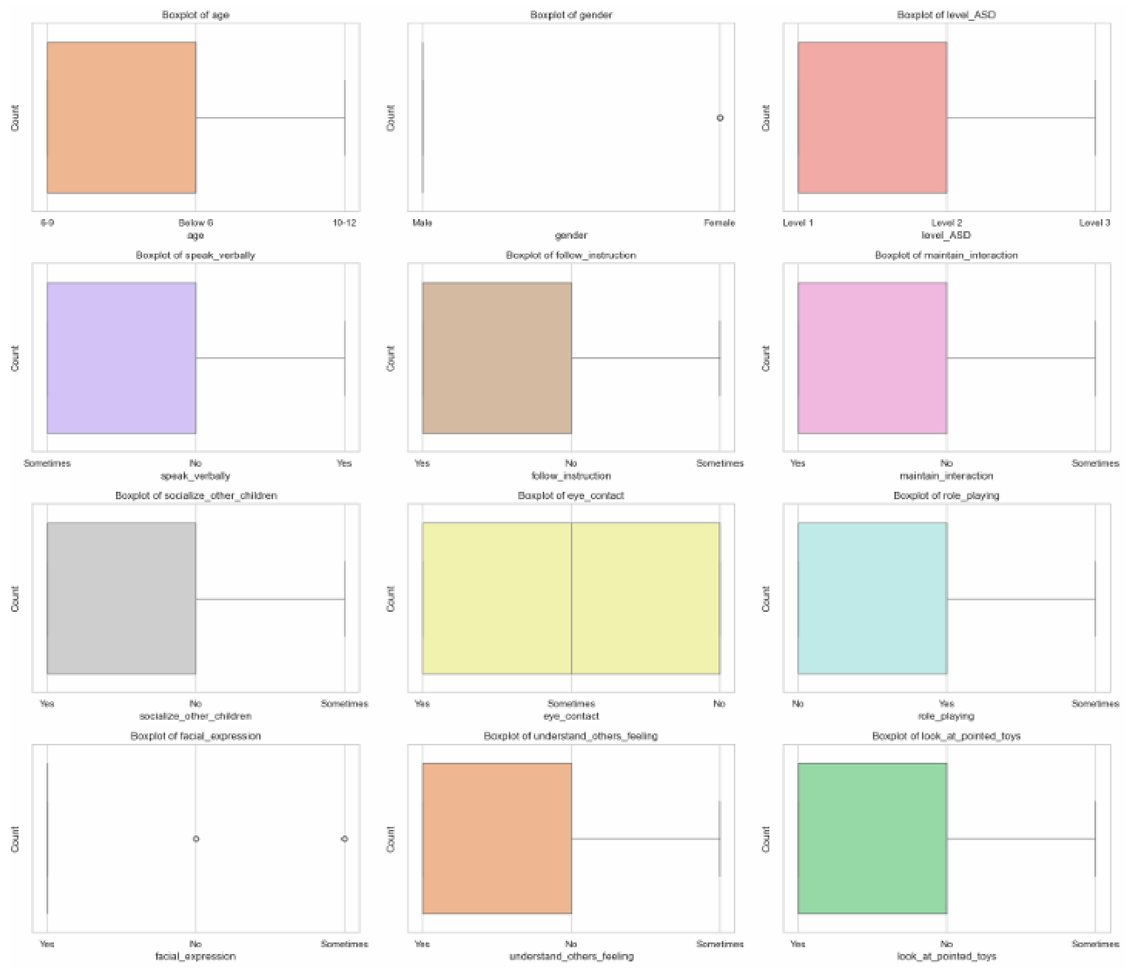
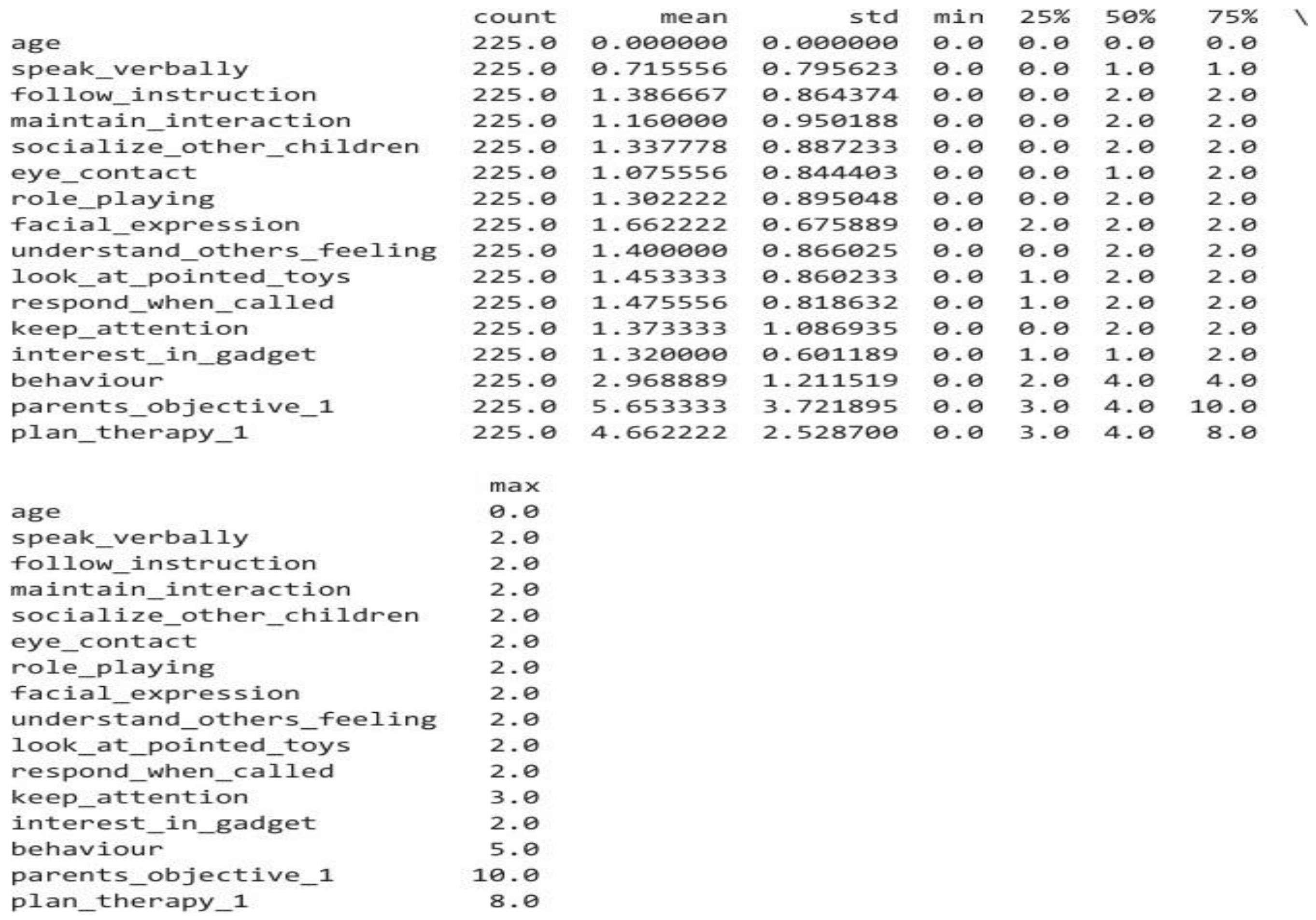


Fig 8 Boxplot

7.1.3 DESCRIPTIVE STATISTICS

Descriptive statistics summarize and organize characteristics of a data set. A data set is a collection of responses or observations from a sample or entire population. In quantitative research, after collecting data, the first step of statistical analysis is to describe characteristics of the responses, such as the average of one variable or the relation between two variables. The next step is inferential statistics, which help you decide whether your data confirms or refutes your hypothesis and whether it is generalizable to a larger population. It Calculates summary statistics such as mean, median, mode, standard deviation, and range for each variable individually.



```
                            count      mean       std  min  25%  50%   75%  \
age                         225.0  0.000000  0.000000  0.0  0.0  0.0   0.0
speak_verbally              225.0  0.715556  0.795623  0.0  0.0  1.0   1.0
follow_instruction          225.0  1.386667  0.864374  0.0  0.0  2.0   2.0
maintain_interaction        225.0  1.160000  0.950188  0.0  0.0  2.0   2.0
socialize_other_children    225.0  1.337778  0.887233  0.0  0.0  2.0   2.0
eye_contact                 225.0  1.075556  0.844403  0.0  0.0  1.0   2.0
role_playing                225.0  1.302222  0.895048  0.0  0.0  2.0   2.0
facial_expression           225.0  1.662222  0.675889  0.0  2.0  2.0   2.0
understand_others_feeling   225.0  1.400000  0.866025  0.0  0.0  2.0   2.0
look_at_pointed_toys        225.0  1.453333  0.860233  0.0  1.0  2.0   2.0
respond_when_called         225.0  1.475556  0.818632  0.0  1.0  2.0   2.0
keep_attention              225.0  1.373333  1.086935  0.0  0.0  2.0   2.0
interest_in_gadget          225.0  1.320000  0.601189  0.0  1.0  1.0   2.0
behaviour                   225.0  2.968889  1.211519  0.0  2.0  4.0   4.0
parents_objective_1         225.0  5.653333  3.721895  0.0  3.0  4.0  10.0
plan_therapy_1              225.0  4.662222  2.528700  0.0  3.0  4.0   8.0

                             max
age                          0.0
speak_verbally               2.0
follow_instruction           2.0
maintain_interaction         2.0
socialize_other_children     2.0
eye_contact                  2.0
role_playing                 2.0
facial_expression            2.0
understand_others_feeling    2.0
look_at_pointed_toys         2.0
respond_when_called          2.0
keep_attention               3.0
interest_in_gadget           2.0
behaviour                    5.0
parents_objective_1         10.0
plan_therapy_1               8.0
```

Fig 9 Descriptive Statistics

7.2 BIVARIANT ANALYSIS

Bivariate analysis is one of the statistical analysis where two variables are observed. One variable here is dependent while the other is independent. These variables are usually denoted by X and Y. So, here we analyse the changes occured between the two variables and to what extent.

Here we are exploring three main univariant analysis. They are

- Heatmap - graphical representation of data using colors to visualize the value of the matrix
- Correlation Matrix - correlation between all the possible pairs of values in a matrix format.

7.2.1 HEATMAP

Heatmap is defined as a graphical representation of data using colors to visualize the value of the matrix. In this, to represent more common values or higher activities brighter colors basically reddish colors are used and to represent less common or activity values, darker colors are preferred. Heatmap is also defined by the name of the shading matrix. By observing how cell colors change across each axis, you can observe if there are any patterns in value for one or both variables. The variables plotted on each axis can be of any type, whether they take on categorical labels or numeric values. Cell colorings can correspond to all manner of metrics, like a frequency count of points in each bin, or summary statistics like mean or median for a third variable. One way of thinking of the construction of a heatmap is as a table or matrix, with color encoding on top of the cells. In certain applications, it is also possible for cells to be colored based on non-numeric values.

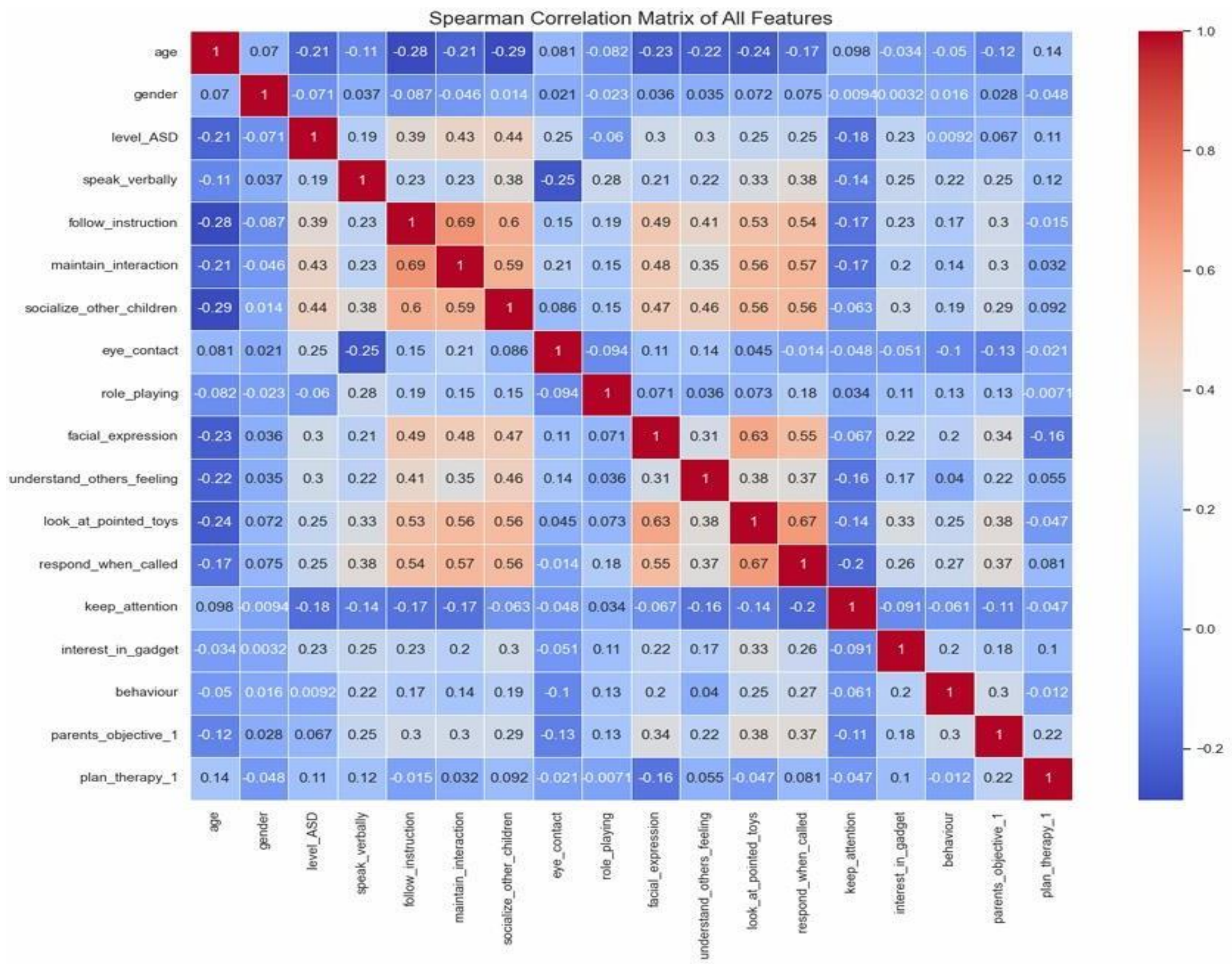
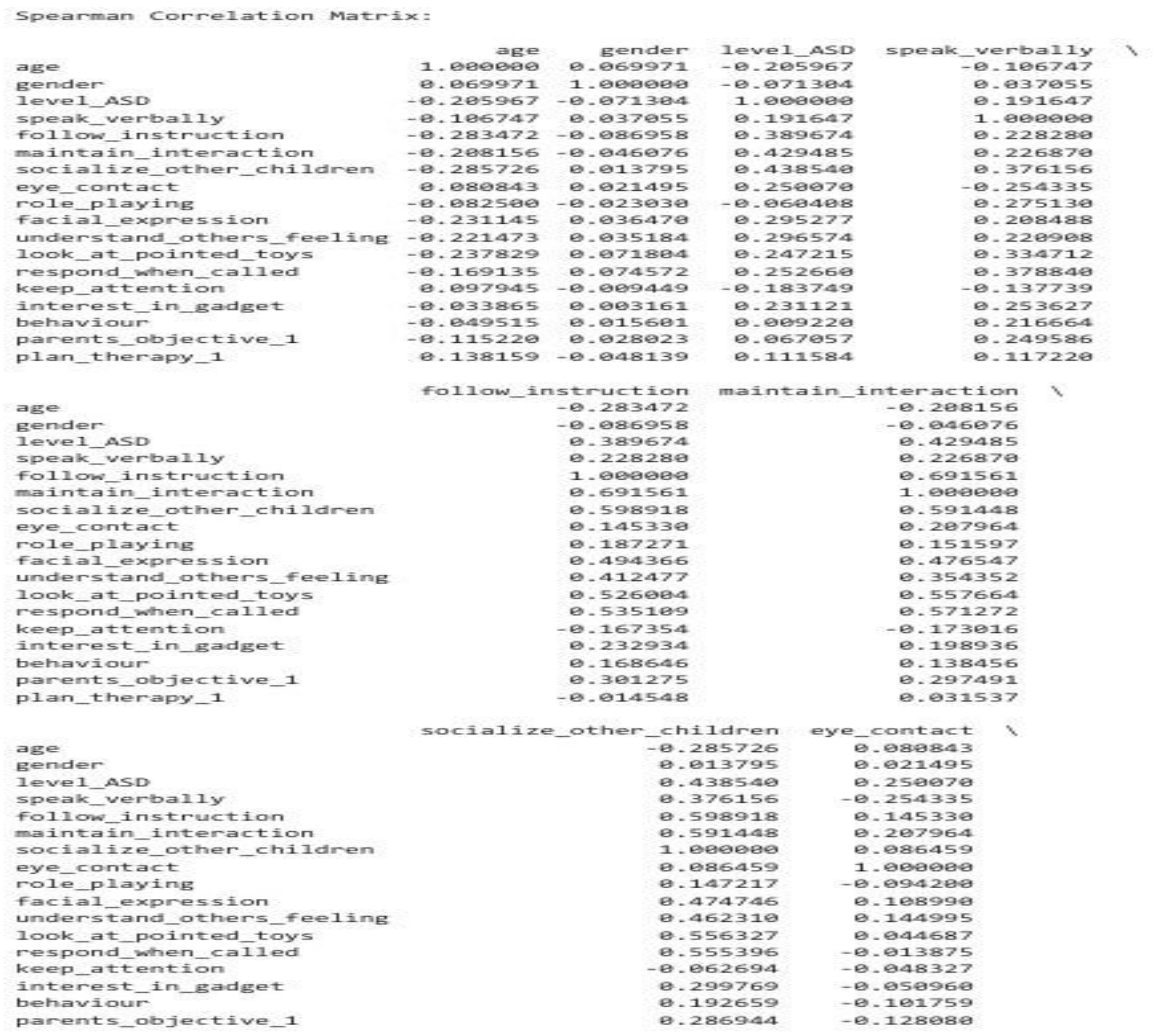


Fig 10 Heatmap

7.2.2 CORRELATION MATRIX

The correlation matrix is a matrix that shows the correlation between variables. It gives the correlation between all the possible pairs of values in a matrix format. Correlation coefficients range from -1 to 1, where:1 indicates a perfect positive linear relationship, -1 indicates a perfect negative linear relationship,0 indicates no linear relationship.



```
Spearman Correlation Matrix:

                                age    gender  level_ASD  speak_verbally  \
age                        1.000000  0.069971  -0.205967       -0.106747
gender                     0.069971  1.000000  -0.071304        0.037055
level_ASD                 -0.205967 -0.071304   1.000000        0.191647
speak_verbally            -0.106747  0.037055   0.191647        1.000000
follow_instruction        -0.283472 -0.086958   0.389674        0.228280
maintain_interaction      -0.208156 -0.046076   0.429485        0.226870
socialize_other_children  -0.285726  0.013795   0.438540        0.376156
eye_contact                0.080843  0.021495   0.250070       -0.254335
role_playing              -0.082500 -0.023030  -0.060408        0.275130
facial_expression         -0.231145  0.036470   0.295277        0.208488
understand_others_feeling -0.221473  0.035184   0.296574        0.220908
look_at_pointed_toys      -0.237829  0.071804   0.247215        0.334712
respond_when_called       -0.169135  0.074572   0.252660        0.378840
keep_attention             0.097945 -0.009449  -0.183749       -0.137739
interest_in_gadget        -0.033865  0.003161   0.231121        0.253627
behaviour                 -0.049515  0.015601   0.009220        0.216664
parents_objective_1       -0.115220  0.028023   0.067057        0.249586
plan_therapy_1             0.138159 -0.048139   0.111584        0.117220

                           follow_instruction  maintain_interaction  \
age                                 -0.283472             -0.208156
gender                              -0.086958             -0.046076
level_ASD                            0.389674              0.429485
speak_verbally                       0.228280              0.226870
follow_instruction                   1.000000              0.691561
maintain_interaction                 0.691561              1.000000
socialize_other_children             0.598918              0.591448
eye_contact                          0.145330              0.207964
role_playing                         0.187271              0.151597
facial_expression                    0.494366              0.476547
understand_others_feeling            0.412477              0.354352
look_at_pointed_toys                 0.526004              0.557664
respond_when_called                  0.535109              0.571272
keep_attention                      -0.167354             -0.173016
interest_in_gadget                   0.232934              0.198936
behaviour                            0.168646              0.138456
parents_objective_1                  0.301275              0.297491
plan_therapy_1                      -0.014548              0.031537

                           socialize_other_children  eye_contact  \
age                                       -0.285726     0.080843
gender                                     0.013795     0.021495
level_ASD                                  0.438540     0.250070
speak_verbally                             0.376156    -0.254335
follow_instruction                         0.598918     0.145330
maintain_interaction                       0.591448     0.207964
socialize_other_children                   1.000000     0.086459
eye_contact                                0.086459     1.000000
role_playing                               0.147217    -0.094200
facial_expression                          0.474746     0.108990
understand_others_feeling                  0.462310     0.144995
look_at_pointed_toys                       0.556327     0.044687
respond_when_called                        0.555396    -0.013875
keep_attention                            -0.062694    -0.048327
interest_in_gadget                         0.299769    -0.050960
behaviour                                  0.192659    -0.101759
parents_objective_1                        0.286944    -0.128080
```

Fig 11 Correlation Matrix

7.3 MULTIVARIANT ANALYSIS

The following figures reveals that the attribute "socialize with other children" emerged as the most influential factor, exhibiting the highest correlation coefficient of 0.414951. This suggests a moderately strong positive relationship between a child's ability to socialize with peers and the likelihood or type of therapy plan recommended. In other words, children who exhibit challenges or strengths in socializing with other children are more likely to have therapy plans tailored accordingly. Since social interaction is a critical developmental milestone, especially in the context of autism-related therapies, this finding emphasizes the importance of evaluating and considering social behavior when designing intervention strategies.

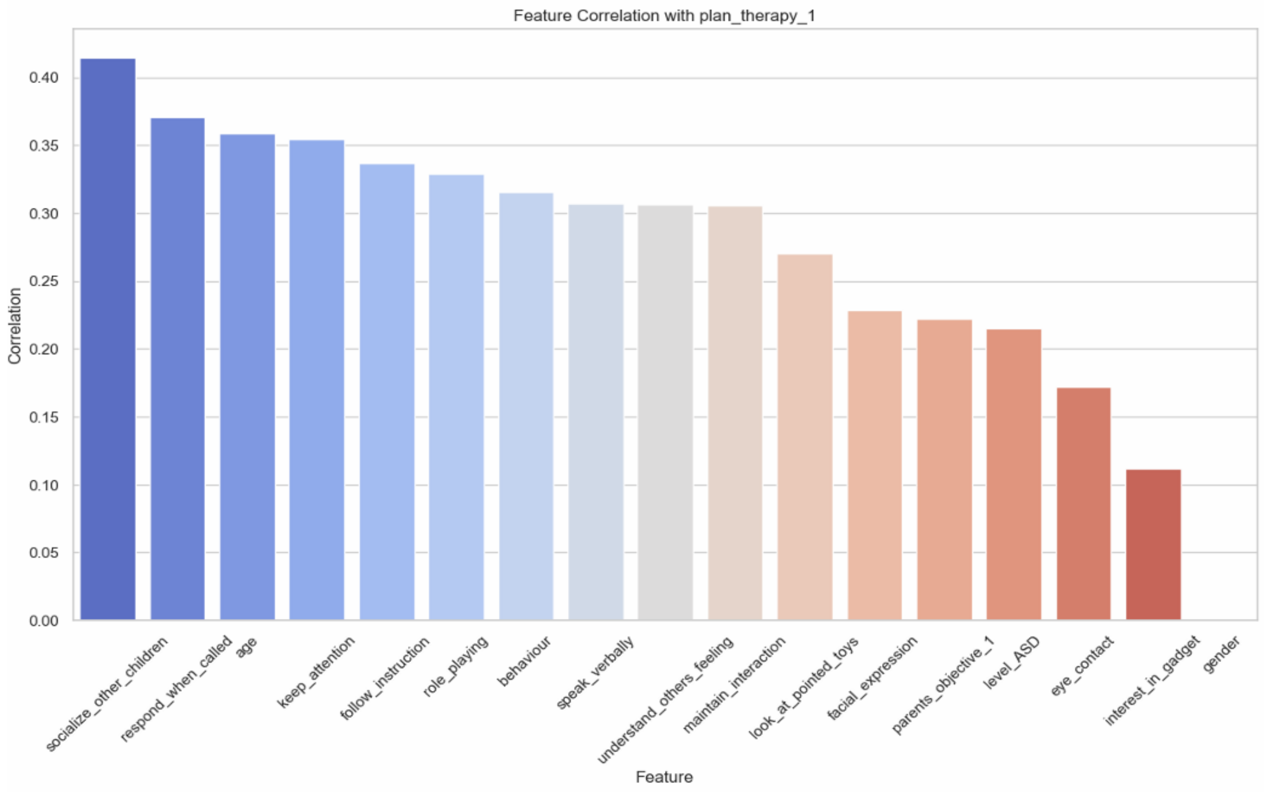


Fig 12 Correlation with Recommended Therapy

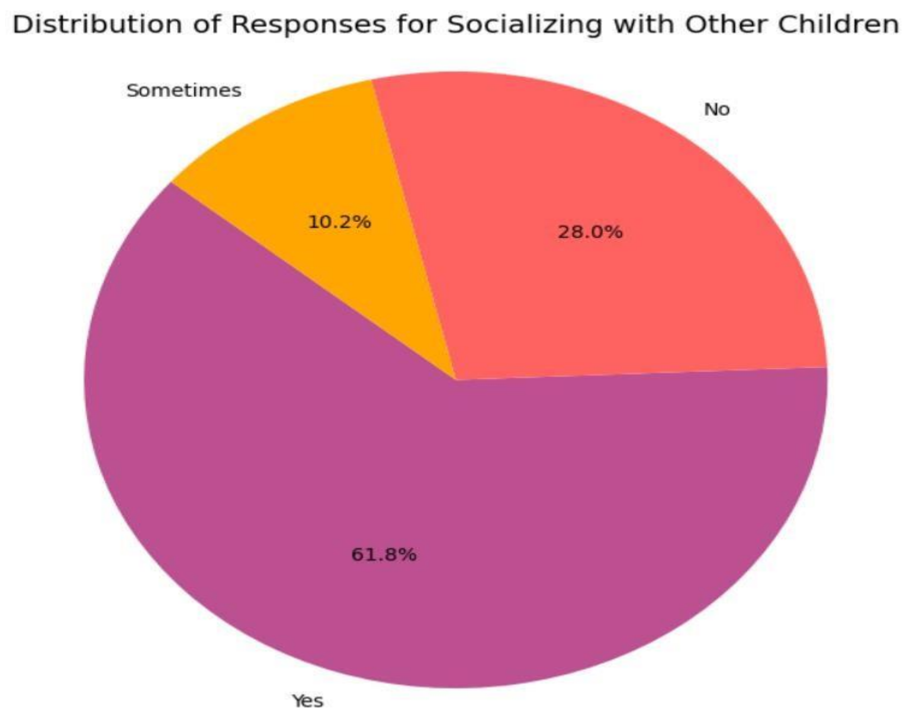


Fig 13 Count of Response for "Socialize with other children"

The distribution of recommended therapy plans was visualized to understand how different therapy types are prescribed across the dataset. The count plot reveals that Occupational Therapy and Speech Therapy are the most commonly recommended interventions for children, followed by therapies like Behavior Therapy, Sensory Integration, and Attention Therapy. This forms an important foundation for further multivariate analysis, as it allows us to explore how multiple independent variables collectively influence the type of therapy recommended.

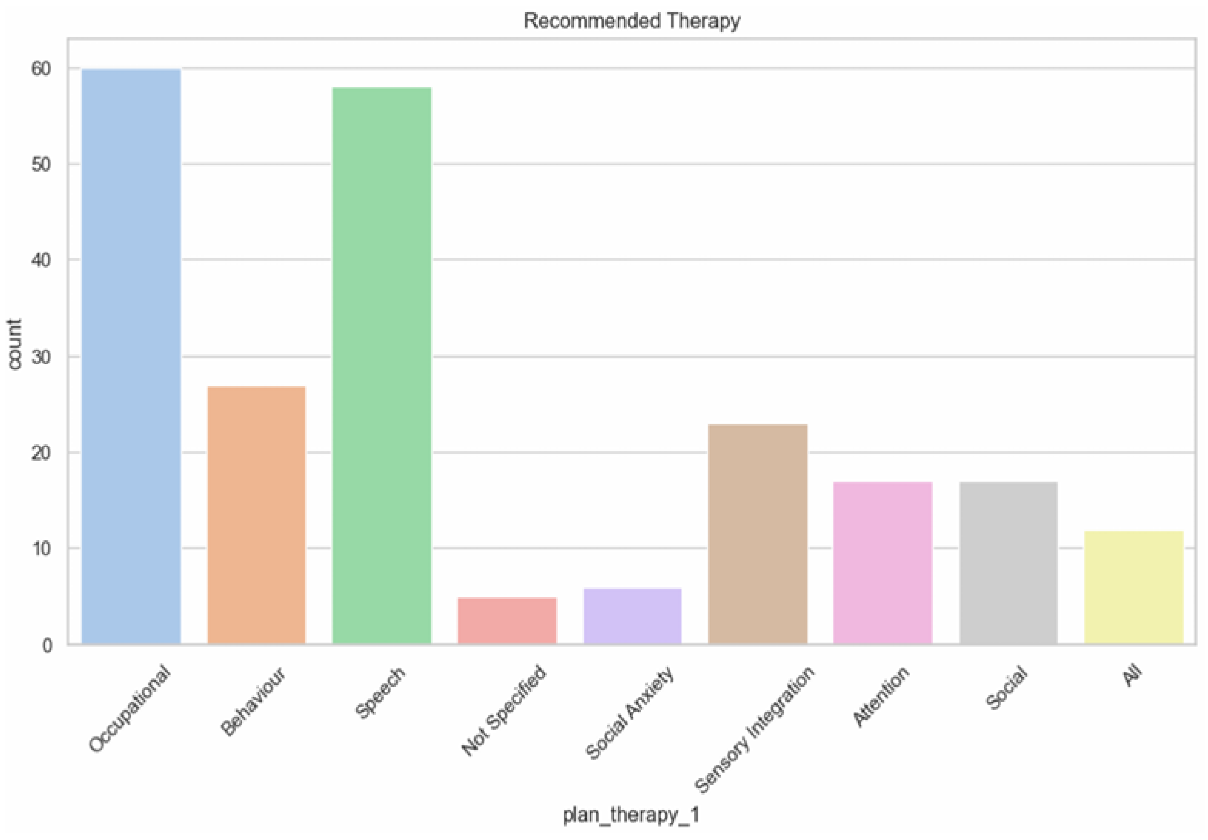


Fig 14 Count of Recommended Therapy

This clustered bar plot illustrates the relationship between recommended therapy plans and the attribute "socialize with other children", categorized as *Yes*, *No*, and *Sometimes*. It's evident that children who socialize well with others (Yes) predominantly receive Speech Therapy and Occupational Therapy, with Speech Therapy being the highest at nearly 50 cases. Conversely, children who struggle to socialize (No) are more evenly distributed across various therapies, though Speech and Occupational therapies still lead. Interestingly, those categorized as "Sometimes" socializing have a relatively lower and more scattered distribution, with a noticeable presence in All therapies combined and Occupational Therapy.

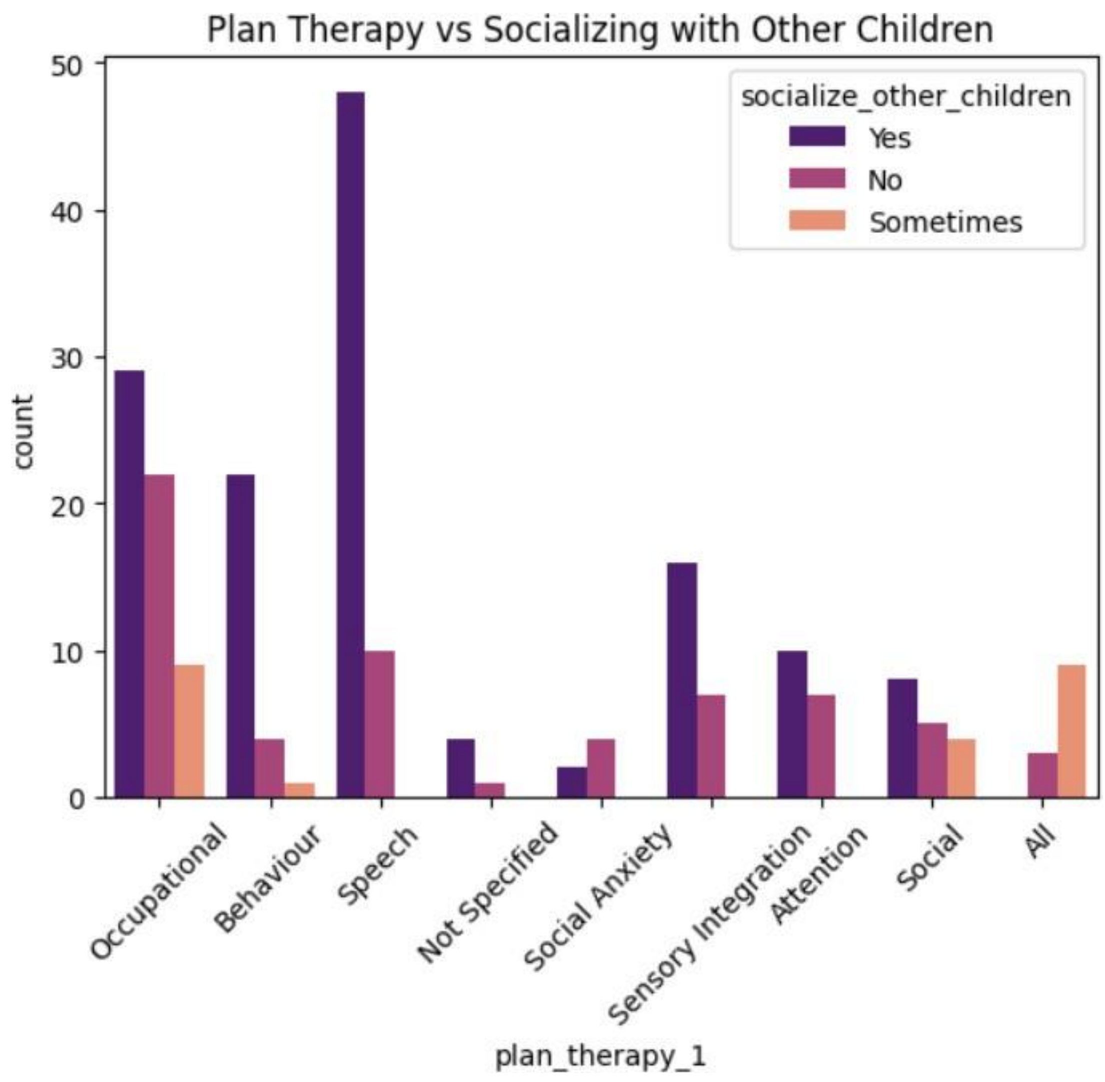


Fig 15 Distribution of Recommended Therapies by Socializing ability

CHAPTER 8

CHAPTER 8

EXPERIMENTAL ANALYSIS

8.1 ALGORITHMS USED

8.1.1 ENSEMBLE LEARNING

Ensemble learning is a machine learning paradigm that combines multiple models to improve the performance and predictive accuracy of a system. The idea is that by aggregating the outputs of several models, the ensemble can often outperform individual models. The primary goal of ensemble learning is to leverage the strengths of various models while mitigating their weaknesses, thereby increasing the overall accuracy and stability of predictions. This two-page content provides an in-depth look at ensemble algorithms, their types, and their applications. An ensemble algorithm refers to a machine learning technique where multiple models (often referred to as "base learners") are trained to solve the same problem, and their outputs are combined to make a final prediction. The basic intuition behind ensemble methods is that multiple weak learners can be combined to create a strong learner. This is often achieved by either averaging or voting on the predictions made by the individual models. The most commonly used ensemble algorithms include Bagging, Boosting, and Stacking each of which uses a different approach to combine multiple models.

8.1.2 RANDOM FOREST CLASSIFIER

Random Forest Classifier is a powerful ensemble learning method used for classification tasks. It builds multiple decision trees during training and combines their outputs to improve accuracy and reduce overfitting. Each tree is trained on a random subset of the data and features (a technique known as bagging), ensuring diversity among the trees. The final prediction is made based on the majority vote from all individual trees. Random Forest is robust to noise and can handle high-dimensional data well. However, it can be less interpretable than simpler models. In this context, the Random Forest Classifier can be used to predict whether an NBA player falls into a high-salary or low-salary category based on attributes like age, points per game (PPG), rebounds (REB), steals (STL), and other performance metrics. Random Forest Classifier is a robust and versatile ensemble learning method that builds a collection (or "forest") of decision trees and aggregates their predictions to perform classification tasks. It is based on the concept of bagging (bootstrap aggregating), where each tree is trained on a random subset of the data and a random subset of features. This introduces diversity and reduces overfitting, making Random Forest more stable than a single decision tree.

Key characteristics:

- **Ensemble method**: Combines multiple decision trees.
- **Majority voting**: Final prediction based on the most common output from all trees.
- **Handles overfitting**: Better generalization than a single decision tree.
- **Feature importance**: Helps identify which features most influence the classification.

Random Forest Classifier is a robust ensemble learning algorithm that combines the outputs of multiple decision trees to make more accurate and stable predictions. Each tree in the forest is trained on a randomly selected subset of the training data (with replacement) using a technique called bagging (Bootstrap Aggregating), and at each split in a tree, a random subset of features is considered. This randomness introduces diversity among the trees, which reduces overfitting and improves generalization. In classification tasks, the final prediction is determined by majority voting—the class that receives the most votes from the individual trees is selected as the output.One of the key advantages of Random Forest is its ability to handle high-dimensional datasets and capture complex feature interactions without heavy preprocessing or feature scaling. It is also relatively resistant to noise and outliers, and provides estimates of feature importance, helping users understand which variables influence the outcome most. However, the model may become computationally expensive with a large number of trees or when applied to very large datasets.

8.1.3 XGBOOST CLASSIFIER

XGBoost (eXtreme Gradient Boosting) is a high-performance, tree-based ensemble algorithm that uses gradient boosting to create a strong classifier from many weak decision-tree learners. Each successive tree is trained to correct the residual errors of the existing ensemble, and the model is optimized by minimizing a regularized objective function that balances training loss with model complexity:

Obj=i=1∑n ℓ(yi,y^i(t))+k=1∑t Ω (fk),Ω(f)=γT+21λj=1∑Twj2

Where,

- ℓ\ellℓ is a differentiable loss (e.g., logistic),
- fkf_kfk is the kkk-th tree with TTT leaves and weights wjw_jwj,
- γ and λ\lambdaλ are regularization hyper-parameters that discourage deep or excessively weighted trees.

XGBoost Classifier (Extreme Gradient Boosting) is a highly efficient and scalable implementation of gradient boosting algorithms, widely used in machine learning competitions and real-world applications due to its exceptional performance. Unlike traditional ensemble methods, XGBoost builds decision trees sequentially, where each new tree corrects the residual errors of the previous ones using gradient descent on a defined loss function. It incorporates regularization terms to control overfitting, making it more robust than other boosting techniques. XGBoost supports parallel processing, missing value handling, and column subsampling, which enhance both speed and accuracy. In classification problems, such as predicting salary tiers of NBA players, XGBoost excels at capturing non-linear interactions and complex patterns in the data. Additionally, it provides tools for interpreting the model, such as feature importance scores and SHAP integration, allowing insights into which player attributes most strongly influence salary predictions. Its flexibility, scalability, and high accuracy make it one of the top choices for classification tasks involving structured/tabular data.

8.1.4 SUPPORT VECTOR MACHINE

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks, but it is particularly effective for classification. The core idea of SVM is to find the optimal hyperplane that best separates the data points of different classes with the maximum margin, ensuring better generalization on unseen data. In a binary classification task, SVM seeks to maximize the distance between the hyperplane and the support vectors, which are the closest data points from each class.

If the data is not linearly separable, SVM uses a technique called the kernel trick to project data into a higher-dimensional space where a linear separator can be found.Mathematical Representation:

The decision function is:

f(x)=sign(w·x+b)f(x) = \text{sign}(w \cdot x + b)f(x)=sign(w·x+b)

where:

- www is the weight vector (orientation of the hyperplane),
- xxx is the feature vector,
- bbb is the bias term,
- The model aims to maximize the margin 2‖w‖\frac{2}{\|w\|}‖w‖2 while minimizing classification errors.

Support Vector Machine (SVM) is a powerful and versatile supervised learning algorithm widely used for classification tasks, particularly when the data is high-dimensional or the decision boundary is not clearly linear. The primary goal of SVM is to find the optimal hyperplane that separates data points from different classes with the maximum margin, ensuring better generalization to unseen data. For datasets that are not linearly separable, SVM uses kernel functions—such as linear, polynomial, and radial basis function (RBF) kernels—to project data into a higher-dimensional space where a linear separator can be found. This makes SVM extremely effective at handling non-linear and complex relationships between features.

8.1.5 SHAP (Shapley Additive Explanations)

SHAP (Shapley Additive Explanations) is a unified framework for interpreting machine learning model predictions by assigning each feature a contribution value based on cooperative game theory. The method is grounded in the concept of Shapley values, which originate from game theory and were developed to fairly distribute payouts among players depending on their contributions to the total payout. In SHAP, the "players" are the input features, and the "payout" is the model’s output (prediction). For each prediction, SHAP calculates how much each feature increased or decreased the predicted value compared to a baseline (typically the average prediction across the dataset). This allows users to understand not only which features are most important globally across the model but also how specific features drive individual predictions. SHAP provides several desirable properties, such as local accuracy (the prediction equals the sum of all SHAP values plus the base value), missingness (features not in the model have zero contribution), and consistency (if a feature’s contribution increases in a newer model, its SHAP value doesn’t decrease).One of SHAP’s most powerful advantages is its model-agnostic nature, meaning it can be applied to any machine learning model, including linear models, tree-based ensembles like Random Forest or XGBoost, and deep learning models like LSTMs.

For tree-based models, SHAP includes a highly efficient algorithm called TreeSHAP, which significantly reduces computation time. SHAP enables both local interpretation, where users can break down a single prediction to see the influence of each feature, and global interpretation, where aggregate SHAP values across all predictions highlight the most influential features overall. The visualizations SHAP provides—such as summary plots, force plots, dependence plots, and waterfall plots—are intuitive and make it easier for stakeholders to understand complex models. In critical applications like healthcare, finance, or player salary prediction in sports analytics, SHAP plays an important role in promoting transparency, trust, and fairness by revealing why a model made a particular decision.

For instance, in an LSTM-based model predicting NBA player salaries, SHAP can identify whether features like minutes played, shooting accuracy, or number of assists positively or negatively influenced a specific prediction.Despite its strengths, SHAP can be computationally intensive, especially for large datasets and complex models. However, due to its theoretical soundness and interpretability capabilities, it has become one of the most trusted tools in the field of Explainable AI (XAI). In modern machine learning workflows, especially where accountability and transparency are critical, SHAP is increasingly adopted not just by researchers but also by data science teams in industry.

8.1.6 LIME (Local Interpretable Model-agnostic Explanations)

It is a model interpretability technique designed to explain the predictions of complex machine learning models in a way that is both simple and human-understandable. The key idea behind LIME is that while a model may behave in a complex, nonlinear way globally, its behavior can often be approximated locally around a specific prediction by a much simpler, interpretable model such as linear regression or a decision tree. LIME operates by first selecting a prediction to explain, then generating a set of perturbed samples by slightly altering the input data around the point of interest. The original model is then used to predict outcomes for these new samples, and an interpretable model is trained

on this new dataset to approximate the local decision boundary. The resulting explanation highlights the most influential features and their direction of impact (positive or negative) on that specific prediction.LIME is model-agnostic, meaning it can be applied to any classifier or regressor, including black-box models such as neural networks, random forests, XGBoost, and support vector machines. Its flexibility and simplicity make it a popular tool for understanding model predictions in domains like finance, healthcare, image classification, and text processing.For example, in a model predicting NBA player salaries, LIME could explain why a particular player is predicted to earn a high salary by
pointing to specific contributing factors like high shooting accuracy, low turnovers, and strong assist numbers. LIME is especially useful in scenarios where individual decision accountability is important, such as credit scoring or medical diagnoses, where stakeholders must be able to trust and verify the reasoning behind a model's decision.A key benefit of LIME is its local focus, which provides explanations that are tailored to individual predictions rather than trying to generalize the entire model behavior.

This makes LIME especially useful for debugging models, validating outcomes, or providing explanations to non-technical audiences. However, LIME does have some limitations. It can be unstable due to its reliance on random sampling, meaning that different runs can sometimes produce slightly different explanations. Additionally, because it only explains predictions locally, it may not provide a full picture of the model's global behavior or feature importance across the dataset. Still, its strength lies in its accessibility and ease of use, and it remains a valuable tool in the toolkit for explainable artificial intelligence (XAI).In deep learning models like LSTM, which are often considered "black-box" due to their complex architecture, LIME can be used to interpret predictions for specific sequences. For example, in a time-series model predicting stock trends or player performance, LIME can identify which time steps or features contributed most significantly to a particular prediction. This helps bridge the gap between high-performance models and real-world interpretability needs, making machine learning models more transparent and trustworthy.

8.1.7 Bayesian optimization

Bayesian Optimization is a powerful and efficient technique for optimizing complex, expensive-to-evaluate functions, particularly in the field of machine learning for hyperparameter tuning. Unlike traditional methods such as grid search or random search that sample blindly, Bayesian Optimization uses a probabilistic model to make intelligent decisions about where to evaluate next. It is especially useful when the objective function is non-convex, noisy, lacks an analytical expression, or is computationally expensive to evaluate—like training deep learning models such as LSTMs, CNNs, or ensemble methods. At the core of Bayesian Optimization is a surrogate model, typically a Gaussian Process (GP), which estimates the relationship between hyperparameters and the performance of the model. This surrogate model is updated with each new evaluation to become increasingly accurate over time.

To choose the next set of hyperparameters to test, Bayesian Optimization uses an acquisition function, such as Expected Improvement (EI), Probability of Improvement (PI), or Upper Confidence Bound (UCB). These functions balance the trade-off between exploration (testing new, uncertain regions of the search space) and exploitation (focusing on areas that have already shown promising results). This strategy allows Bayesian Optimization to find near-optimal solutions with far fewer iterations compared to brute-force methods.

As a result, it has become a preferred method for tuning hyperparameters in models where training is time-consuming or resource-intensive.In the context of machine learning, Bayesian Optimization is commonly used to tune critical hyperparameters like learning rate, number of hidden layers, number of LSTM units, dropout rate, regularization parameters, and more. For example, in an LSTM model used to predict NBA player salaries based on historical performance data, Bayesian Optimization can automatically discover the most effective set of hyperparameters that maximize prediction accuracy or minimize loss on validation data. This removes the need for manual tuning and significantly improves the model’s performance and generalization ability.Beyond hyperparameter tuning, Bayesian Optimization is also used in domains like experimental design, robotics, reinforcement learning, and even neural architecture search (NAS).

One of its main strengths lies in being sample-efficient—it can reach high performance with relatively few evaluations, making it ideal for costly models. However, its efficiency comes at the cost of increased computational overhead in maintaining and updating the surrogate model, especially in high-dimensional spaces or with large datasets. Nevertheless, libraries like Optuna, Scikit-Optimize, and BayesianOptimization in Python have made this technique more accessible and easy to integrate into modern machine learning workflows.In summary, Bayesian Optimization is an advanced and intelligent approach to optimizing machine learning models, particularly well-suited for scenarios where training is expensive, and model performance is sensitive to hyperparameters. Its ability to make informed decisions about where to sample next makes it a valuable tool in building accurate, efficient, and robust predictive models.

8.1.8 LSTM (Long Short-Term Memory)

LSTM (Long Short-Term Memory) is a specialized type of Recurrent Neural Network (RNN) designed to effectively model and learn from sequential data. Traditional RNNs often struggle with long-term dependencies due to the vanishing gradient problem, which causes them to forget information as the sequence length increases. LSTM networks address this issue through the use of a **memory cell** and three key **gates**—the **input gate**, **forget gate**, and **output gate**—which regulate the flow of information. These gates allow LSTMs to selectively remember or forget information over long sequences, making them ideal for tasks where context and order matter, such as time-series forecasting, natural language processing, and speech recognition.LSTMs are particularly powerful in applications involving **temporal patterns**, such as predicting stock prices, weather conditions, or player performance over time. For example, in sports analytics, an LSTM model can be used to predict an NBA player's future salary or performance metrics based on their past game statistics, capturing patterns and trends that unfold over weeks or seasons.

LSTMs can also be stacked into multiple layers (stacked LSTM) or combined with convolutional layers (CNN-LSTM) to handle more complex sequence data or spatial-temporal dependencies.One of the major advantages of LSTM is its ability to handle varying sequence lengths and retain relevant information from earlier time steps while ignoring irrelevant data, all within a single model. This makes it far more effective than traditional machine learning models or simple RNNs when dealing with time-dependent data. Furthermore, LSTM networks can be enhanced with **dropout**, **batch normalization**, or **attention mechanisms** to improve regularization, stability, and performance on complex tasks.However, LSTMs are computationally more expensive than simpler models, both in training time and memory usage, due to their complex internal structure. They may also require careful hyperparameter tuning (e.g., learning rate, number of units, number of layers), which can be optimized using methods like Bayesian Optimization**.** Despite these challenges, LSTM remains one of the most popular and effective models for sequential prediction problems.In modern deep learning workflows, LSTMs are often used alongside explainability tools such as **SHAP** and **LIME** to better understand the model's decision-making process and gain insight into which parts of the sequence were most influential in the prediction. This is especially important in domains such as healthcare, finance, or sports, where transparency and trust in model outputs are crucial.

CHAPTER 9

CHAPTER 9

PERFORMANCE ANALYSIS OF PROPOSED APPROACH

Discussion On Results

The evaluation of classification models for NBA player salary classification involved the use of ensemble and individual algorithms, including Random Forest Classifier, XGBoost Classifier, Support Vector Machine (SVM), and a Voting Classifier that combined these models. Each algorithm was trained and rigorously assessed using key performance metrics such as accuracy, F1-score, precision, and recall, as summarized in Table 1. Among all models, the Voting Classifier demonstrated the highest performance, achieving the best accuracy and F1-score, thereby showcasing its ability to leverage the strengths of multiple classifiers for improved predictive capability. This superior performance makes the Voting Classifier the most reliable choice for accurately classifying the type of therapy based on patient-related features.

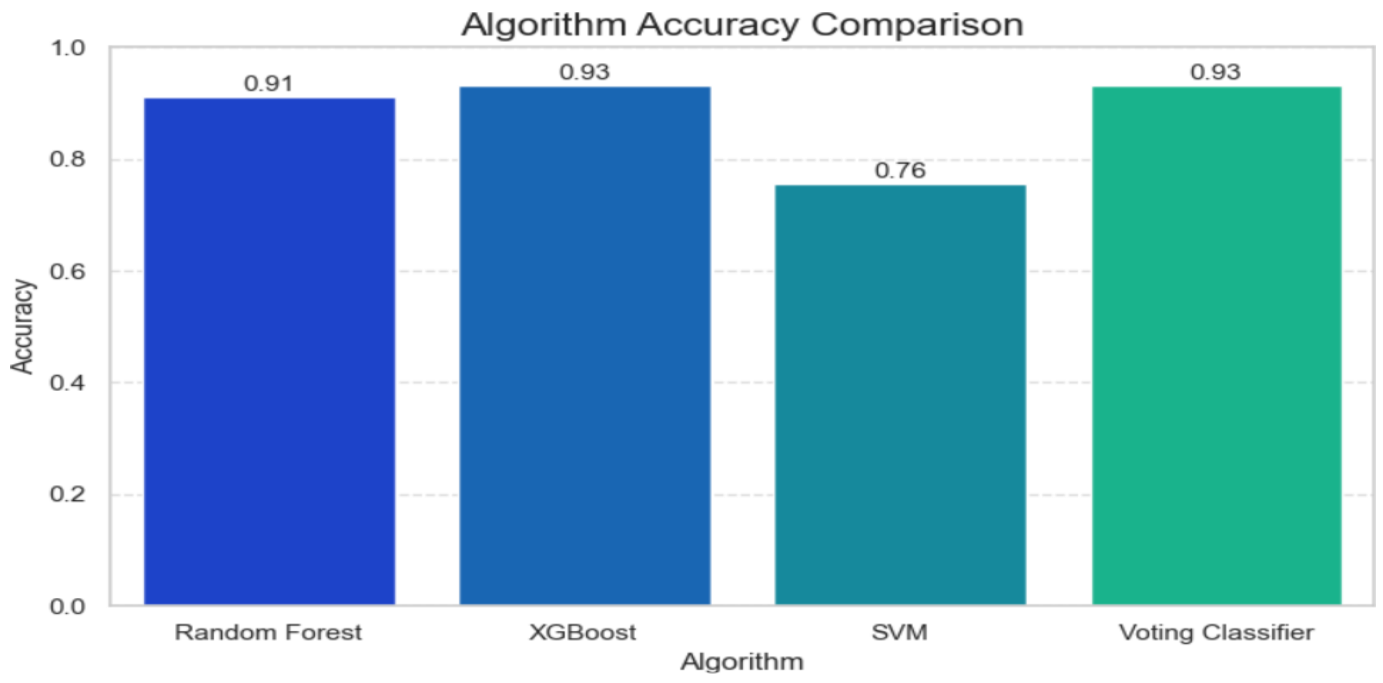


Fig 16 Accuracy comparison between algorithms

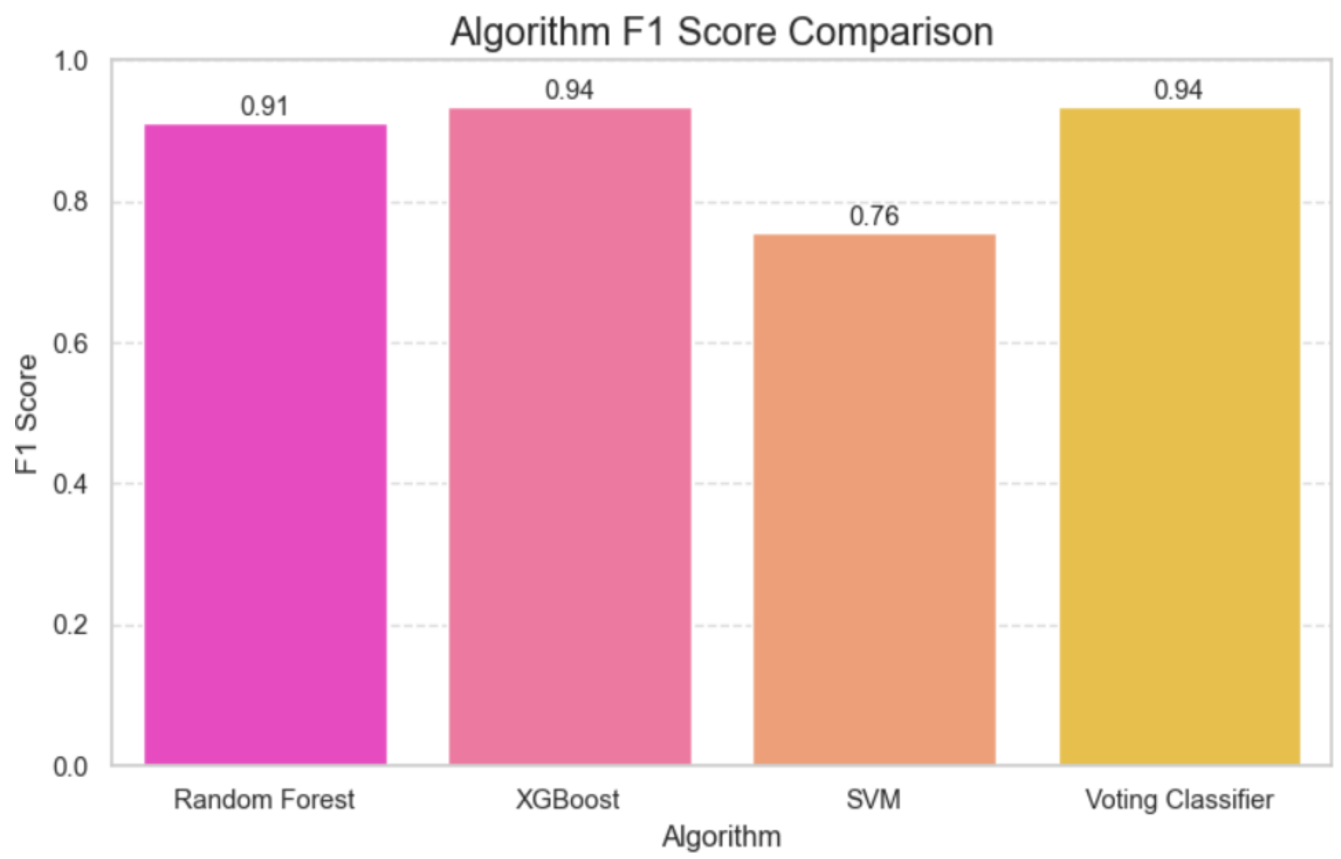


Fig 17 F1 Score comparison between algorithms

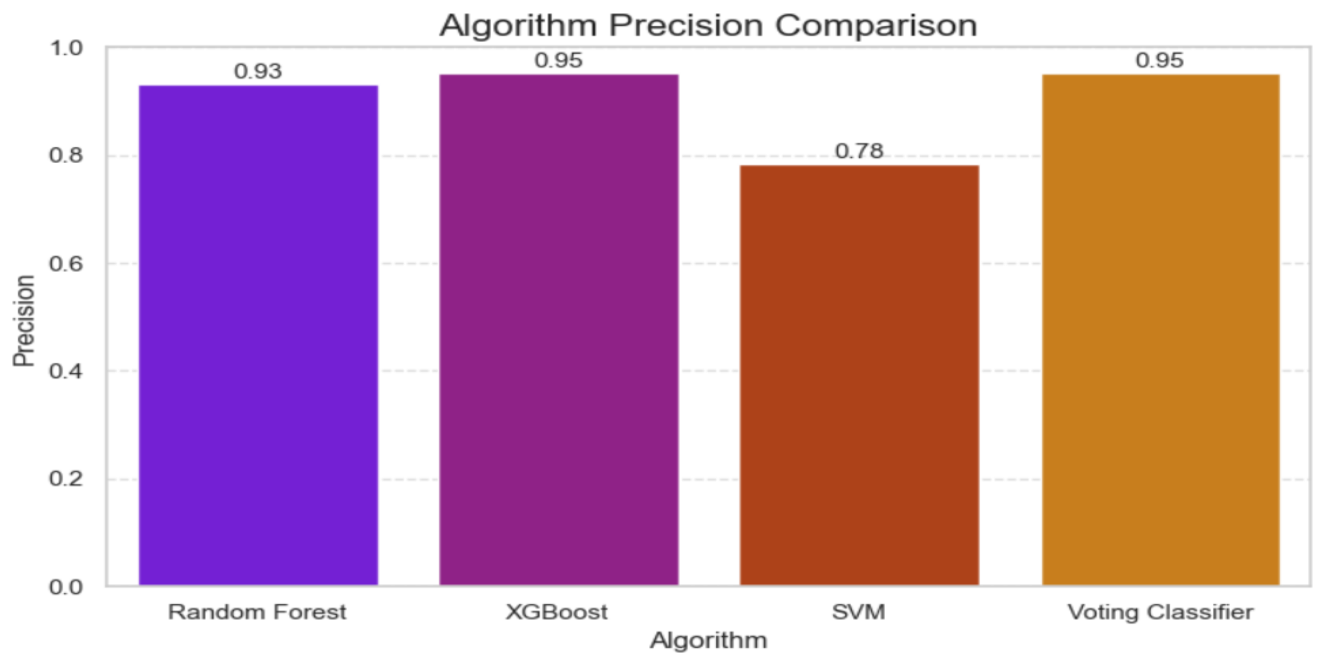


Fig 18 Recall comparison between algorithms

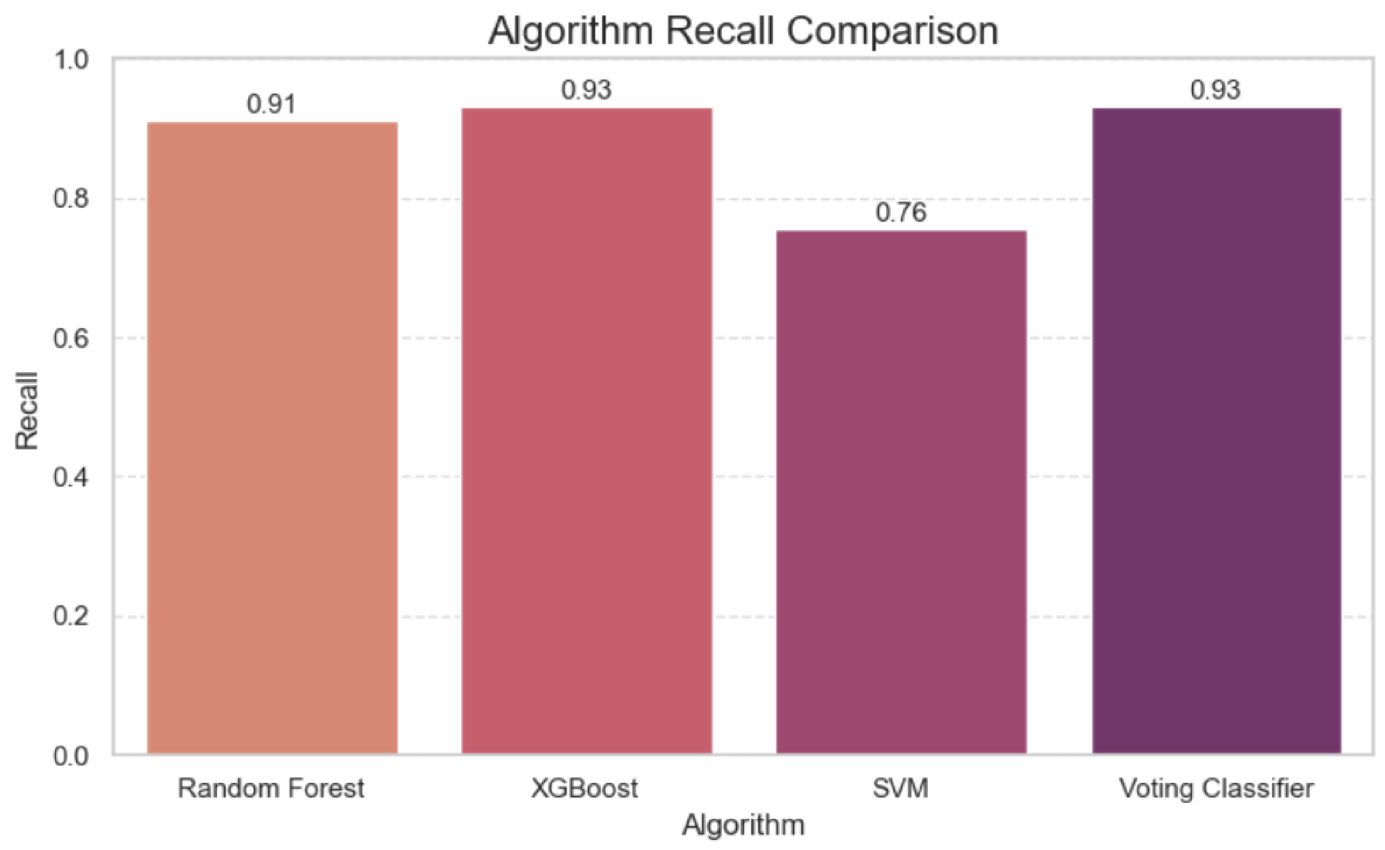


Fig 19 Precision comparison between algorithms

Table 1

Evaluation between different algorithms

Algorithms	**Accuracy**	**F1 Score**	**Recall**	**Precision**
Random Forest	0.91	0.91	0.93	0.91
XGBoost	0.93	0.94	0.95	0.93
SVM	0.76	0.76	0.78	0.76
Voting Classifier	0.93	0.94	0.95	0.93

CHAPTER 10

CHAPTER 10
CONCLUSION AND FUTURE SCOPE

10.1 CONCLUSION

The system aims to provide valuable insights into the effectiveness of various classification models for predicting personalized therapy recommendations for individuals with Autism Spectrum Disorder (ASD), highlighting the growing importance of data-driven methodologies in modern healthcare decision-making. The training and evaluation results for multiple classification models applied to the therapy prediction task revealed notable differences in their performance. The Random Forest Classifier demonstrated strong predictive capability, achieving an accuracy of 91.11%, with an F1 Score of 91.26%, precision of 93.19%, and recall of 91.11%, indicating its consistent ability to handle diverse and complex patient data. XGBoost Classifier slightly outperformed Random Forest, recording the highest accuracy of 93.33%, along with an F1 Score of 93.58%, precision of 95.19%, and recall of 93.33%. This underscores XGBoost's efficiency in structured healthcare data and its capability to model intricate relationships between patient behaviours and therapy outcomes effectively.

On the other hand, the Support Vector Machine (SVM) model demonstrated moderate performance, with an accuracy of 75.56%, F1 Score of 75.54%, precision of 78.31%, and recall of 75.56%, suggesting limitations in addressing the complexities within high-dimensional, overlapping feature spaces inherent in ASD-related datasets. Notably, the Voting Classifier, which combined Random Forest, XGBoost, and SVM, matched the best-performing individual model (XGBoost), achieving an accuracy of 93.33%, F1 Score of 93.58%, precision of 95.19%, and recall of 93.33%. This demonstrates the ensemble model's strength in leveraging complementary patterns captured by its constituent algorithms, delivering reliable and balanced classification outcomes.

10.2 Future scope

For future enhancements, the Therapy Efficacy Model (TEM) can be improved by incorporating additional machine learning techniques like Neural Networks to further refine predictions. Integrating multimodal data, such as genetic, brain imaging, and physiological data, could provide a more comprehensive understanding of ASD, leading to more accurate therapy recommendations. The system could also adopt reinforcement learning for real-time adaptation of therapy strategies based on patient progress. Expanding the platform to mobile devices and incorporating Natural Language Processing (NLP) would enhance user interaction, allowing for seamless communication between patients and doctors. Additionally, virtual or augmented reality could be explored to create engaging, interactive therapy exercises. By utilizing federated learning, the model can be trained on decentralized data while maintaining privacy, ensuring scalability and robustness. Lastly, ensuring multilingual support would expand accessibility, making the system more inclusive and available globally.

APPENDICES

APPENDIX-A

SOURCE CODE

```
import numpy as np
import pandas as pd

#laod the dataset
dataset = pd.read_excel('Final Autism Dataset.xlsx')

#display first five rows
dataset.head(5)

#Display the number of rows and columns
dataset.shape

# Check for missing/null values
dataset.isnull().sum()

# Transpose the summary statistics for better readability
summary_statistics = dataset.describe().transpose()

# Display the result
print(summary_statistics)

dataset.nunique()

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))
sns.countplot(x='gender', data=dataset, palette='coolwarm')
plt.title('Gender Distribution')
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(x='level_ASD', data=dataset, palette='viridis')
plt.title('Autism Level Distribution')
plt.show()
plt.figure(figsize=(12, 6))
sns.countplot(x='speak_verbally', data=dataset, palette='Set2')
plt.title('Speech Ability')
plt.show()

plt.figure(figsize=(12, 6))
sns.countplot(x='plan_therapy_1', data=dataset, palette='pastel')
plt.title('Recommended Therapy')
plt.xticks(rotation=45)
plt.show()

# Create the countplot
plt.figure(figsize=(8, 5))
sns.countplot(x='socialize_other_children', data=dataset, palette='viridis')

# Add title and labels
plt.title('Count of Responses for "Socialise with Other Children"', fontsize=14)
plt.xlabel('Response')
plt.ylabel('Count')

# Show the plot
plt.show()

therapy_autism_ct = pd.crosstab(dataset['level_ASD'], dataset['plan_therapy_1'])
print("\nAutism Level vs Therapy Plan:\n", therapy_autism_ct)

# Set up Seaborn theme
sns.set(style="whitegrid")

# Number of attributes (columns)
columns = dataset.columns

# Set up a grid for plots
num_cols = 3
num_rows = -(-len(columns) // num_cols)  # Ceiling division

# Set figure size
plt.figure(figsize=(18, num_rows * 4))

# Loop through columns and plot histograms
for i, col in enumerate(columns, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.histplot(dataset[col], kde=True, color=sns.color_palette("pastel")[i % 10])
    plt.title(f'Histogram of {col}')


    plt.xlabel(col)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()

# Set up Seaborn theme
sns.set(style="whitegrid")

# Number of attributes (columns)
columns = dataset.columns

# Set up a grid for plots
num_cols = 3
num_rows = -(-len(columns) // num_cols)  # Ceiling division

# Set figure size
plt.figure(figsize=(18, num_rows * 4))

# Loop through columns and plot histograms
for i, col in enumerate(columns, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.boxplot(x=dataset[col], color=sns.color_palette("pastel")[i % 10])
    plt.title(f'Boxplot of {col}')
    plt.xlabel(col)
    plt.ylabel('Count')

plt.tight_layout()
plt.show()

from scipy.stats import chi2_contingency, spearmanr

target_col = "plan_therapy_1"

df_encoded = dataset.apply(lambda x: pd.factorize(x)[0] if x.dtype == 'object' else
x)

def cramers_v(x, y):
    confusion_matrix = pd.crosstab(x, y)
    chi2 = chi2_contingency(confusion_matrix)[0]
    n = confusion_matrix.sum().sum()
    phi2 = chi2 / n
    r, k = confusion_matrix.shape
    phi2corr = max(0, phi2 - ((k-1)*(r-1))/(n-1))


    r_corr = r - ((r-1)**2)/(n-1)
    k_corr = k - ((k-1)**2)/(n-1)
    return np.sqrt(phi2corr / min((k_corr-1), (r_corr-1)))

correlation_results = {}

for col in df_encoded.columns:
    if col != target_col:
        if df_encoded[col].nunique() > 10:
            correlation, _ = spearmanr(df_encoded[col], df_encoded[target_col])
        else:
            correlation = cramers_v(df_encoded[col], df_encoded[target_col])

        correlation_results[col] = correlation

correlation_df = pd.DataFrame(list(correlation_results.items()),
columns=["Feature", "Correlation"])
correlation_df = correlation_df.sort_values(by="Correlation", ascending=False)

print("\nCorrelation of All Features with Target Column:\n")
print(correlation_df)

plt.figure(figsize=(15, 8))
sns.barplot(x=correlation_df["Feature"], y=correlation_df["Correlation"],
palette="coolwarm")
plt.xticks(rotation=45)
plt.ylabel("Correlation")
plt.title(f"Feature Correlation with {target_col}")
plt.show()

dataset_encoded = dataset.apply(lambda x: pd.factorize(x)[0] if x.dtype == 'object'
else x)

# Compute Spearman correlation matrix
```

```
spearman_corr_matrix = dataset_encoded.corr(method='spearman')

# Display the correlation matrix
print("Spearman Correlation Matrix:\n")
print(spearman_corr_matrix)

import seaborn as sns
import matplotlib.pyplot as plt

# Compute Spearman correlation matrix


correlation_matrix = df_encoded.corr(method='spearman')

# Set up the figure
plt.figure(figsize=(16, 12))

# Plot the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5,
square=True, cbar=True)

# Title
plt.title('Spearman Correlation Matrix of All Features', fontsize=16)

# Display the heatmap
plt.tight_layout()
plt.show()

from sklearn.model_selection import train_test_split

important_features = [
    "socialize_other_children",
    "respond_when_called",
    "age",
    "keep_attention",
    "follow_instruction",
    "role_playing",
    "behaviour",
    "speak_verbally",
    "understand_others_feeling",
    "maintain_interaction",
    "look_at_pointed_toys",
    "facial_expression",
    "parents_objective_1",
    "level_ASD",
    "eye_contact"
]

dataset.info()

dataset["age"] = pd.to_numeric(dataset["age"], errors="coerce")     # Convert to
numeric

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()


for col in dataset.columns:
    dataset[col] = le.fit_transform(dataset[col])  # Convert categorical to numbers

dataset = pd.get_dummies(dataset, columns=["gender", "level_ASD"],
drop_first=True)

dataset.head()

dataset.isnull().sum()

important_features = [
    "socialize_other_children",
    "respond_when_called",
    "age",
    "keep_attention",
    "follow_instruction",
    "role_playing",    "behaviour",
    "speak_verbally",
    "understand_others_feeling",
    "maintain_interaction",
    "look_at_pointed_toys",
    "facial_expression",
    "parents_objective_1",
    "level_ASD_1",
    "level_ASD_2",
    "eye_contact"
]


X = dataset.drop(['plan_therapy_1'],axis=1)
y = dataset["plan_therapy_1"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier


from sklearn.linear_model import LogisticRegression
from sklearn.utils.class_weight import compute_class_weight
from imblearn.over_sampling import SMOTE

# 1. Prepare your dataset
X = X.copy()
for col in X.columns:
    if X[col].dtype == 'bool':
        X[col] = X[col].astype(int)

X_np = X.to_numpy().astype(float)
y_np = y.to_numpy()  # assuming y is a pandas Series

# 2. Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_np)
```

```
# 3. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_np, test_size=0.2,
random_state=42, stratify=y_np)

# 4. SMOTE for oversampling
smote = SMOTE(random_state=42, k_neighbors=2)
X_train_sm, y_train_sm = smote.fit_resample(X_train, y_train)

# 5. Compute class weights (optional since SMOTE balances it, but still useful)
classes = np.unique(y_train_sm)
weights = compute_class_weight(class_weight='balanced', classes=classes,
y=y_train_sm)
class_weight_dict = dict(zip(classes, weights))

# 6. Define models with class_weight
rf = RandomForestClassifier(n_estimators=100, random_state=42,
class_weight='balanced')
svm = SVC(probability=True, kernel='linear', random_state=42,
class_weight='balanced')
xgb = XGBClassifier(n_estimators=100, use_label_encoder=False,
eval_metric='mlogloss', random_state=42)

# 7. Voting classifier
voting_model = VotingClassifier(
    estimators=[('rf', rf), ('xgb', xgb), ('svm', svm)],
    voting='soft'

)

# 8. Train
voting_model.fit(X_train_sm, y_train_sm)

# 9. Predict
y_pred_train = voting_model.predict(X_train_sm)
y_pred_test = voting_model.predict(X_test)

# 10. Evaluate
print("Training Accuracy:", accuracy_score(y_train_sm, y_pred_train))
print("Testing Accuracy:", accuracy_score(y_test, y_pred_test))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_test))
print("\nClassification Report:\n", classification_report(y_test, y_pred_test))

from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, StackingClassifier
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

rf = RandomForestClassifier(n_estimators=100, random_state=42)
xgb = XGBClassifier(n_estimators=100, use_label_encoder=False,
eval_metric='mlogloss', random_state=42)
svm = SVC(probability=True, kernel='linear', random_state=42)

meta_classifier = LogisticRegression()

stacked_model = StackingClassifier(
    estimators=[('rf', rf), ('xgb', xgb), ('svm', svm)],
    final_estimator=meta_classifier
)

stacked_model.fit(X_train, y_train)

train_pred = stacked_model.predict(X_train)
test_pred = stacked_model.predict(X_test)

train_acc = accuracy_score(y_train, train_pred)
test_acc = accuracy_score(y_test, test_pred)
conf_matrix = confusion_matrix(y_test, test_pred)
class_report = classification_report(y_test, test_pred)



print(f"Training Accuracy: {train_acc:.4f}")
print(f"Testing Accuracy: {test_acc:.4f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense, Input, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = X.copy()
for col in X.columns:
    if X[col].dtype == 'bool':
        X[col] = X[col].astype(int)

X_np = X.to_numpy()  # or X.values

X_np = X_np.astype(float)  # ensure float for StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_np)

X_reshaped = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))

X_train, X_test, y_train, y_test = train_test_split(X_reshaped, y, test_size=0.2,
random_state=42)

y_train = to_categorical(y_train, num_classes=9)
y_test = to_categorical(y_test, num_classes=9)

input_layer = Input(shape=(X_reshaped.shape[1], X_reshaped.shape[2]))

lstm_out = LSTM(128, return_sequences=False)(input_layer)
lstm_out = Dropout(0.3)(lstm_out)

dense_out = Dense(64, activation="relu")(lstm_out)
dense_out = Dropout(0.3)(dense_out)
```

```
output_layer = Dense(9, activation="softmax")(dense_out)

optimizer = Adam(learning_rate=0.0005)
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(loss='categorical_crossentropy',                    optimizer=optimizer,
metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,
y_test))

loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import LSTM, Dense, Input, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils.class_weight import compute_class_weight

# 1. Preprocess
X = X.copy()
for col in X.columns:
    if X[col].dtype == 'bool':
        X[col] = X[col].astype(int)

X_np = X.to_numpy().astype(float)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_np)

# 2. Reshape for LSTM
X_reshaped = X_scaled.reshape((X_scaled.shape[0], 1, X_scaled.shape[1]))

# 3. Train-test split
X_train,   X_test,   y_train_raw,   y_test_raw   =   train_test_split(X_reshaped,   y,
test_size=0.2, random_state=42)
```

```
# 4. Compute class weights
import numpy as np
from collections import Counter
from sklearn.utils.class_weight import compute_class_weight


classes = np.unique(y_train_raw)
class_weights = compute_class_weight(class_weight='balanced', classes=classes,
y=y_train_raw)
class_weights_dict = dict(zip(classes, class_weights))
print("Class weights:", class_weights_dict)

# 5. One-hot encode targets
y_train = to_categorical(y_train_raw, num_classes=9)
y_test = to_categorical(y_test_raw, num_classes=9)

# 6. Build LSTM Model
input_layer = Input(shape=(X_reshaped.shape[1], X_reshaped.shape[2]))
lstm_out = LSTM(128, return_sequences=False)(input_layer)
lstm_out = Dropout(0.3)(lstm_out)
dense_out = Dense(64, activation="relu")(lstm_out)
dense_out = Dropout(0.3)(dense_out)
output_layer = Dense(9, activation="softmax")(dense_out)

optimizer = Adam(learning_rate=0.0005)
model = Model(inputs=input_layer, outputs=output_layer)
model.compile(loss='categorical_crossentropy',                    optimizer=optimizer,
metrics=['accuracy'])

# 7. Train with class weights
model.fit(X_train, y_train, epochs=100, batch_size=32, validation_data=(X_test,
y_test), class_weight=class_weights_dict)

# 8. Evaluate
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {accuracy:.4f}")

from sklearn.metrics import f1_score, precision_score, recall_score,
accuracy_score
```

```
models = {
    'Random Forest': rf,
    'XGBoost': xgb,
    'SVM': svm,
    'Voting Classifier': voting_model
}

# Dictionary to hold the results
results = {}

for name, model in models.items():
    # Train the model
    model.fit(X_train_sm, y_train_sm)

    # Predict
    y_pred = model.predict(X_test)

    # If predictions are 2D, convert them back to 1D
    if len(y_pred.shape) > 1 and y_pred.shape[1] > 1:
        y_pred = np.argmax(y_pred, axis=1)

    # Calculate metrics
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='weighted')
    prec = precision_score(y_test, y_pred, average='weighted')
    rec = recall_score(y_test, y_pred, average='weighted')

    # Store the results
    results[name] = {
        'Accuracy': acc,
        'F1 Score': f1,
        'Precision': prec,
        'Recall': rec
    }

# Print results nicely
for model_name, metrics in results.items():
    print(f"\nModel: {model_name}")
    for metric_name, value in metrics.items():
        print(f"{metric_name}: {value:.4f}")
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# Predict on test data
y_pred_test = voting_model.predict(X_test)

# Accuracy
voting_accuracy = accuracy_score(y_test, y_pred_test)

# Precision
voting_precision = precision_score(y_test, y_pred_test, average='weighted')  # or
'macro' depending on your needs

# Recall
voting_recall = recall_score(y_test, y_pred_test, average='weighted')

# F1-Score
voting_f1 = f1_score(y_test, y_pred_test, average='weighted')

# Print results
print("Voting Classifier Performance on Test Data:")
print(f"Accuracy  : {voting_accuracy:.4f}")
print(f"Precision : {voting_precision:.4f}")
print(f"Recall    : {voting_recall:.4f}")
print(f"F1-Score  : {voting_f1:.4f}")

# Extract models and accuracies
models = list(results.keys())
accuracies = [results[model]['Accuracy'] for model in models]

# Create barplot
plt.figure(figsize=(8, 5))
sns.barplot(x=models, y=accuracies, palette='winter')

# Styling
plt.title('Algorithm Accuracy Comparison', fontsize=16)
plt.xlabel('Algorithm', fontsize=12)
plt.ylabel('Accuracy', fontsize=12)
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
# Show exact accuracy values on top of bars
for i, value in enumerate(accuracies):
    plt.text(i, value + 0.01, f"{value:.2f}", ha='center', fontsize=11)

plt.tight_layout()
plt.show()

# Extract models and accuracies
models = list(results.keys())
accuracies = [results[model]['F1 Score'] for model in models]

# Create barplot
plt.figure(figsize=(8, 5))
sns.barplot(x=models, y=accuracies, palette='spring')

# Styling
plt.title('Algorithm F1 Score Comparison', fontsize=16)
plt.xlabel('Algorithm', fontsize=12)
plt.ylabel('F1 Score', fontsize=12)
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show exact accuracy values on top of bars
for i, value in enumerate(accuracies):
    plt.text(i, value + 0.01, f"{value:.2f}", ha='center', fontsize=11)

plt.tight_layout()
plt.show()




# Extract models and accuracies
models = list(results.keys())
accuracies = [results[model]['Recall'] for model in models]

# Create barplot
plt.figure(figsize=(8, 5))
sns.barplot(x=models, y=accuracies, palette='flare')
```

```
# Styling
plt.title('Algorithm Recall Comparison', fontsize=16)
plt.xlabel('Algorithm', fontsize=12)
plt.ylabel('Recall', fontsize=12)
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show exact accuracy values on top of bars
for i, value in enumerate(accuracies):
    plt.text(i, value + 0.01, f"{value:.2f}", ha='center', fontsize=11)

plt.tight_layout()
plt.show()




# Extract models and accuracies
models = list(results.keys())
accuracies = [results[model]['Precision'] for model in models]

# Create barplot
plt.figure(figsize=(8, 5))
sns.barplot(x=models, y=accuracies, palette='gnuplot')

# Styling
plt.title('Algorithm Precision Comparison', fontsize=16)
plt.xlabel('Algorithm', fontsize=12)
plt.ylabel('Precision', fontsize=12)
plt.ylim(0, 1)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Show exact accuracy values on top of bars
for i, value in enumerate(accuracies):
    plt.text(i, value + 0.01, f"{value:.2f}", ha='center', fontsize=11)

plt.tight_layout()
plt.show()
```

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
from werkzeug.utils import secure_filename
import sqlite3
import numpy as np
import pandas as pd
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
import json
import logging
import os


app = Flask(__name__)
app.secret_key = '123'
UPLOAD_FOLDER = 'static/uploads'

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['DATABASE'] = 'data.db'

database = 'data.db'

def init_db():
    with sqlite3.connect(database) as conn:
        cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS users (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT NOT NULL,
                    mail TEXT NOT NULL UNIQUE,
                    password TEXT NOT NULL,
                    profile_pic TEXT)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS doctor (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    username TEXT NOT NULL,
                    mail TEXT NOT NULL UNIQUE,
                    password TEXT NOT NULL,
                    profile_pic TEXT,
                    specialization TEXT NOT NULL,
                    medical_register_id TEXT NOT NULL UNIQUE)''')
##    cursor.execute("DROP TABLE IF EXISTS user_profile;")

    cursor.execute('''CREATE TABLE IF NOT EXISTS user_profile (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    mail TEXT NOT NULL UNIQUE,
                    age INT NOT NULL,
                    speak_verbal TEXT NOT NULL,
                    follow_instruction TEXT NOT NULL,
                    maintain_interaction TEXT NOT NULL,
                    socialize_other_children TEXT NOT NULL,
                    eye_contact TEXT NOT NULL,
                    role_playing TEXT NOT NULL,
                    facial_expression TEXT NOT NULL,
                    understand_others_feelings TEXT NOT NULL,
                    look_at_pointed_toys TEXT NOT NULL,
                    respond_when_called TEXT NOT NULL,
                    keep_attention TEXT NOT NULL,
                    interest_in_gadget TEXT NOT NULL,
                    behaviour TEXT NOT NULL,
                    parent_objective TEXT NOT NULL,
                    gender TEXT NOT NULL,
                    level_ASD_1 TEXT NOT NULL,
                    level_ASD_2 TEXT NOT NULL,
                    status INT NOT NULL
                    )''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS user_profiles (
                    email TEXT PRIMARY KEY,
                    name TEXT NOT NULL,
                    profile_data TEXT)''')


    cursor.execute('''CREATE TABLE IF NOT EXISTS suggestions (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    email TEXT NOT NULL,
                    prediction TEXT NOT NULL,
                    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
                    suggestion TEXT NOT NULL)''')

    cursor.execute('''CREATE TABLE IF NOT EXISTS
user_prediction_suggestion (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    email TEXT NOT NULL,
                    profile_data TEXT NOT NULL,
                    prediction TEXT NOT NULL,
                    suggestion TEXT NOT NULL)''')



    conn.commit()
    conn.close()

def get_profile_pic(filename):
    if not filename:
        # fallback to first available image or default
        upload_folder = os.path.join(app.root_path, 'static/uploads')
        available_pics = [f for f in os.listdir(upload_folder) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
        fallback = available_pics[0] if available_pics else 'default.png'
        return url_for('static', filename='uploads/' + fallback)
    return url_for('static', filename='uploads/' + filename)

def get_user(email):
    with sqlite3.connect(app.config['DATABASE']) as conn:
        conn.row_factory = sqlite3.Row
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE mail = ?", (email,))
        row = cursor.fetchone()
        return dict(row) if row else None

with open('stacked_model.pkl', 'rb') as f:
    stacked_model = pickle.load(f)



@app.route('/')
def index():
    return render_template('index.html')

@app.route('/doctor_details')
def doctor_details():
    return render_template('doctor_details.html')
```

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        print("Form Data:", request.form)
        print("Files:", request.files)

        username = request.form.get('username')
        mail = request.form.get('mail')
        password = request.form.get('password')
        profile_pic = request.files.get('profile_pic')

        if not username or not mail or not password or not profile_pic:
            flash('All fields are required.', 'danger')
            return redirect(url_for('register'))

        # Save profile picture
        if profile_pic and profile_pic.filename:
            filename = secure_filename(profile_pic.filename)
            profile_pic_path = os.path.join(app.config['UPLOAD_FOLDER'],
filename)
            profile_pic.save(profile_pic_path)
            profile_pic_url = filename  # 🔥 Store only the filename
        else:
            profile_pic_url = None

        conn = sqlite3.connect(app.config['DATABASE'])
        cursor = conn.cursor()
        try:
            cursor.execute("INSERT INTO users (username, mail, password,
profile_pic) VALUES (?, ?, ?, ?)",
                           (username, mail, password, profile_pic_url))
            conn.commit()
            flash('Registration successful! Please login.', 'success')
            return redirect(url_for('login'))
        except sqlite3.IntegrityError:
            flash('Username or Email already exists. Try a different one.', 'danger')
        finally:
            conn.close()

    return render_template('register.html')
```

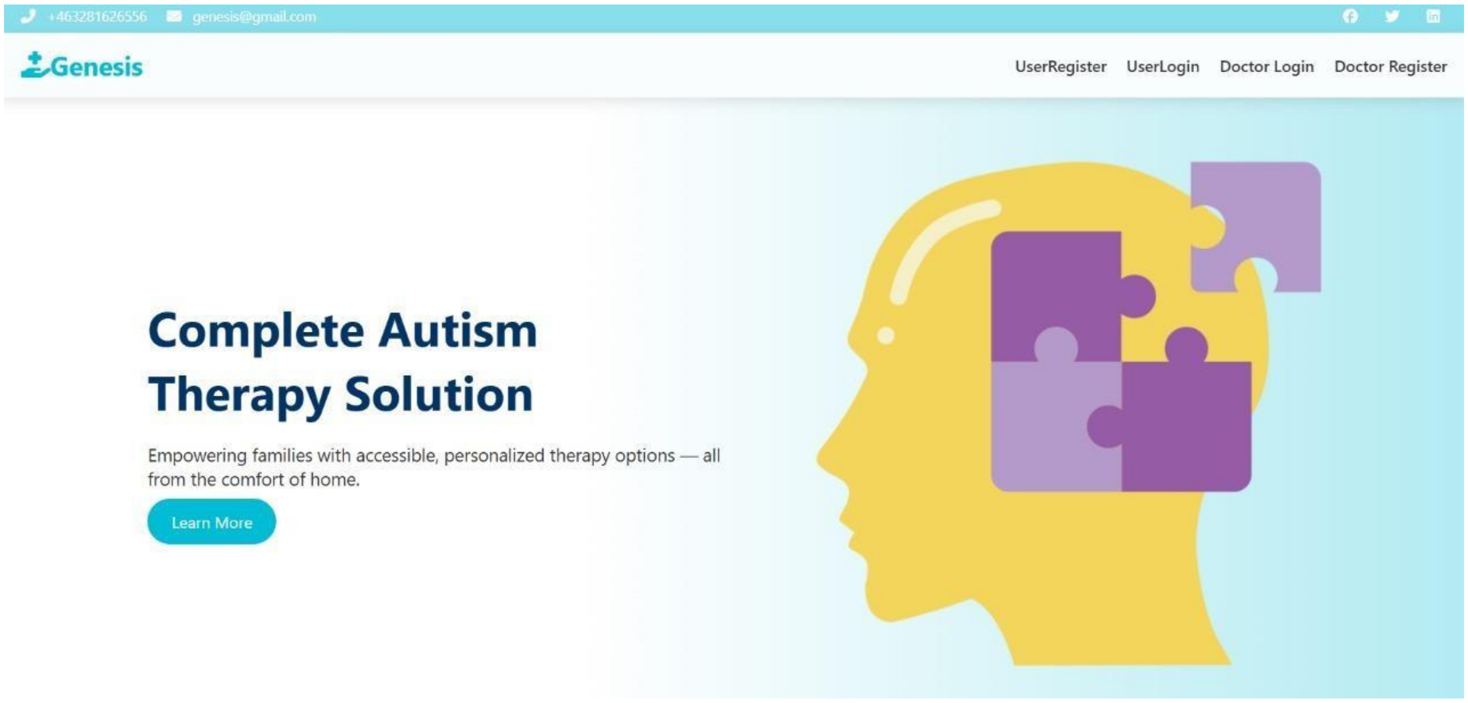
```
@app.route('/login', methods=['GET', 'POST'])
def login():
    global username
    if request.method == 'POST':
        username = request.form['mail']
        password = request.form['password']

        conn = sqlite3.connect(database)
        cursor = conn.cursor()
        cursor.execute("SELECT * FROM users WHERE mail = ? AND password
= ?", (username, password,))
        user = cursor.fetchone()
        conn.close()

        if user:
            return redirect(url_for('dashboard', email=username))
        else:
            flash('Invalid credentials. Please try again.', 'danger')

    return render_template('login.html')


@app.route('/doctor_register', methods=['GET', 'POST'])
def doctor_register():
    if request.method == 'POST':
        username = request.form['username']
        mail = request.form['mail']
        password = request.form['password']
        specialization = request.form['specialization']
        medical_register_id = request.form['medical_register_id']
        profile_pic = request.files['profile_pic']


        if profile_pic and profile_pic.filename:
            filename = secure_filename(profile_pic.filename)
            profile_pic_path = os.path.join(app.config['UPLOAD_FOLDER'],
filename)
            profile_pic.save(profile_pic_path)
            profile_pic_url = url_for('static', filename='uploads/' + filename)
        else:
            profile_pic_url = None
        conn = sqlite3.connect(app.config['DATABASE'])
        cursor = conn.cursor()
        try:
            cursor.execute("INSERT INTO doctor (username, mail, password, 
profile_pic, specialization, medical_register_id) VALUES (?, ?, ?, ?, ?, ?)",
                           (username, mail, password, profile_pic_url, specialization, 
medical_register_id))
            conn.commit()
            flash('Registration successful! Please login.', 'success')
            return redirect(url_for('doctor_login'))
        except sqlite3.IntegrityError:
            flash('Username, Email, or Medical Register ID already exists. Try a 
different one.', 'danger')
        finally:
            conn.close()

    return render_template('doctor_register.html')


@app.route('/doctor_login', methods=['GET', 'POST'])
def doctor_login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']

        with sqlite3.connect(app.config['DATABASE']) as conn:
            cursor = conn.cursor()
            cursor.execute("SELECT password FROM doctor WHERE username = 
?", (username,))
            result = cursor.fetchone()


        if result and result[0] == password:
            session['username'] = username
            return redirect(url_for('doctor_dashboard'))
        else:
            flash("Invalid credentials. Try again.", "danger")

    return render_template('doctor_login.html')

@app.route('/doctor_dashboard')
```

```
def doctor_dashboard():
    username = session.get('username') or request.args.get('username')

    if not username:
        flash("Username not found, please log in again.", "danger")
        return redirect(url_for("doctor_login"))

    session['username'] = username

    # Fetch profile pic, specialization, and medical_register_id from the database
    with sqlite3.connect(app.config['DATABASE']) as conn:
        cursor = conn.cursor()
        cursor.execute("""
            SELECT profile_pic, specialization, medical_register_id
            FROM doctor
            WHERE username = ?
        """, (username,))
        result = cursor.fetchone()

    if result:
        profile_pic, specialization, medical_register_id = result

        # Store values in session
        session['profile_pic'] = profile_pic or None
        session['specialization'] = specialization
        session['medical_register_id'] = medical_register_id  # Store
medical_register_id

    # If there's no profile picture in the session, use default or first available pic
    if not session.get('profile_pic'):
        upload_folder = os.path.join(app.root_path, 'static/uploads')
        available_pics = [f for f in os.listdir(upload_folder) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
        if available_pics:
            session['profile_pic'] = url_for('static', filename='uploads/' +
available_pics[0])
        else:
            session['profile_pic'] = None

    # Render the doctor dashboard template with the medical_register_id
    return render_template(
        'doctor_dashboard.html',
        username=session['username'],
        profile_pic=session['profile_pic'],
        specialization=session.get('specialization'),
        medical_register_id=session.get('medical_register_id'),  # Pass
medical_register_id to the template
    )


@app.route('/dashboard')
def dashboard():
    email = request.args.get('email')
    user = get_user(email)  # Returns a dict

    if user is None:
        flash("User not found. Please log in again.", "danger")
        return redirect(url_for("login"))

    profile_pic = user.get('profile_pic')
    username = user.get('username')

    if not profile_pic:
        # fallback to any image or default
        upload_folder = os.path.join(app.root_path, 'static/uploads')
        available_pics = [f for f in os.listdir(upload_folder) if
f.lower().endswith(('.png', '.jpg', '.jpeg'))]
        if available_pics:
            profile_pic = url_for('static', filename='uploads/' + available_pics[0])
        else:
            profile_pic = url_for('static', filename='uploads/default.png')
    else:
        profile_pic = url_for('static', filename='uploads/' + profile_pic)  # ✅ Only
prepend here

    user_dict = {
        'username': username,
        'mail': email,
        'profile_pic': profile_pic
    }

    return render_template(
        'dashboard.html',
        email=email,
        username=username,
        user=user_dict
    )


def get_username_from_email(email):
    conn = sqlite3.connect(database)
    cursor = conn.cursor()
    cursor.execute("SELECT username FROM users WHERE mail = ?", (email,))
    user = cursor.fetchone()
    conn.close()
    return user[0] if user else None

@app.route('/profile', methods=['GET', 'POST'])
def profile():
    global username

    if request.method == 'POST':
        # Extract form data
        name = request.form.get('name')
        age = request.form.get('age')
        speak_verbal = request.form.get('speak')
        follow_instruction = request.form.get('follow')
        maintain_interaction = request.form.get('interaction')
        socialize_other_children = request.form.get('socialize_other_children')
        eye_contact = request.form.get('eye_contact')
        role_playing = request.form.get('role_playing')
        facial_expression = request.form.get('facial_express')
        understand_others_feelings = request.form.get('other_feel')
        look_at_pointed_toys = request.form.get('look_at_points')
        respond_when_called = request.form.get('respond')
        keep_attention = request.form.get('keep_attention')
        interest_in_gadget = request.form.get('gadgets')
        behaviour = request.form.get('behavior')
        parent_objective = request.form.get('parent_object')
        gender = request.form.get('gender')
        level_asd_1 = request.form.get('level_1')
        level_asd_2 = request.form.get('level_2')
        print(level_asd_2,"level")
```

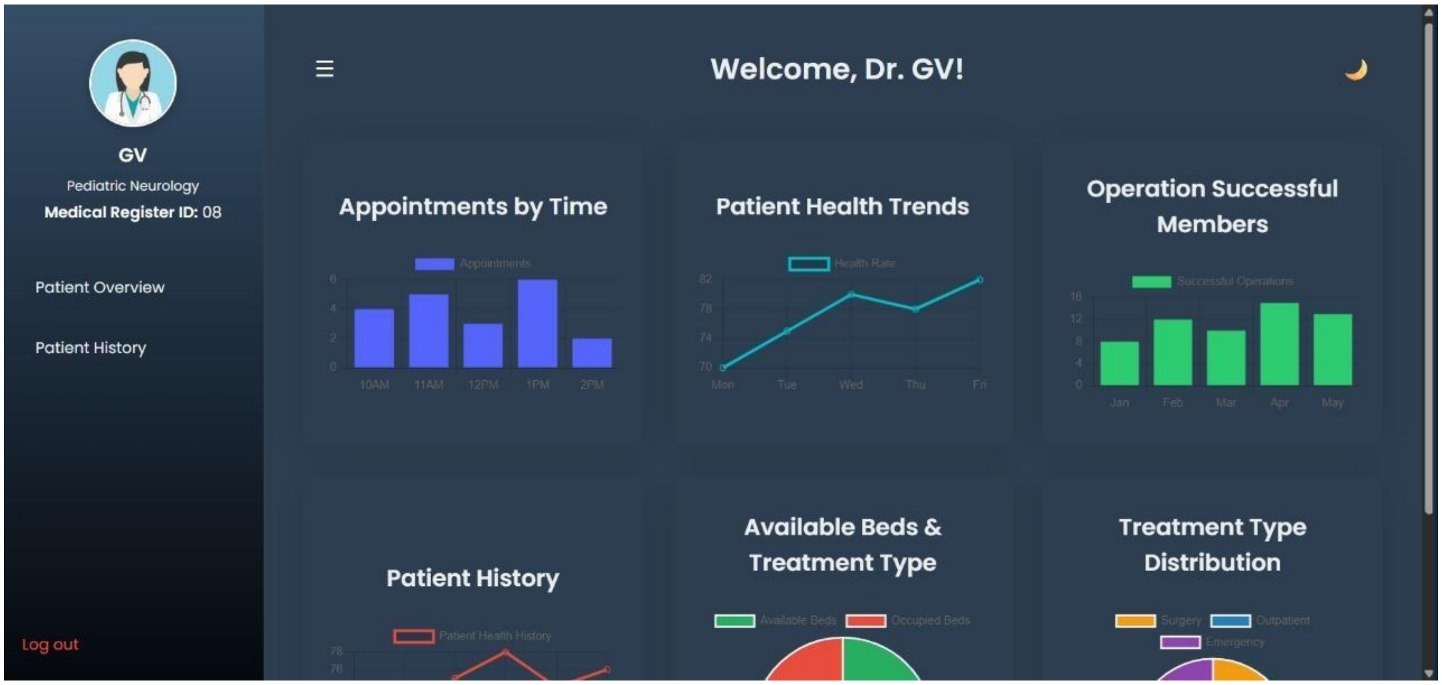
```
##        level_asd_3 = request.form.get('level_3')

        # Validate required fields
        if not name or not age or not gender:
            flash('Name, Age, and Gender are required fields.', 'danger')
            return redirect(url_for('profile'))
        # Database operations
        with sqlite3.connect(database, timeout=10) as conn:
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM user_profile WHERE mail = ?",
(username,))
            existing_profile = cursor.fetchone()
            print(existing_profile,"profile")

            if existing_profile:
                cursor.execute("""
                    UPDATE user_profile
                    SET age=?, speak_verbal=?, follow_instruction=?,
maintain_interaction=?,
                        socialize_other_children=?, eye_contact=?, role_playing=?,
                        faical_expression=?, understand_others_feelings=?,
                        look_at_poined_toys=?, respond_when_called=?,
keep_attention=?,
                        interest_in_gadget=?, behaviour=?, parent_objective=?,
                        gender=?, level_ASD_1=?, level_ASD_2=?, status=?
                    WHERE mail=?
                """, (age, speak_verbal, follow_instruction, maintain_interaction,
                      socialize_other_children, eye_contact, role_playing,
                      facial_expression, understand_others_feelings,
                      look_at_pointed_toys, respond_when_called, keep_attention,
                      interest_in_gadget, behaviour, parent_objective,
                      gender, level_asd_1, level_asd_2, 0, username))
                flash('Profile updated successfully.', 'info')
            else:
                cursor.execute("""
                    INSERT INTO user_profile (mail, age, speak_verbal,
follow_instruction,
                                              maintain_interaction, socialize_other_children,
                                              eye_contact, role_playing, facial_expression,
                                              understand_others_feelings, look_at_pointed_toys,
                                              respond_when_called, keep_attention,
                                    interest_in_gadget, behaviour, parent_objective,
                                    gender, level_ASD_1, level_ASD_2, status)
                    VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
                """, (username, age, speak_verbal, follow_instruction,
                      maintain_interaction, socialize_other_children, eye_contact,
                      role_playing, facial_expression, understand_others_feelings,
                      look_at_pointed_toys, respond_when_called, keep_attention,
                      interest_in_gadget, behaviour, parent_objective,
                      gender, level_asd_1, level_asd_2, 0))
                flash('Profile created successfully.', 'success')


            # Save profile data in user_profiles table
            cursor.execute("""
                INSERT OR REPLACE INTO user_profiles (email, name,
profile_data)
                VALUES (?, ?, ?)""", (username, name, json.dumps({
                    "age": age,
                    "speak_verbal": speak_verbal,
                    "follow_instruction": follow_instruction,
                    "maintain_interaction": maintain_interaction,
                    "socialize_other_children": socialize_other_children,
                    "eye_contact": eye_contact,
                    "role_playing": role_playing,
                    "facial_expression": facial_expression,
                    "understand_others_feelings": understand_others_feelings,
                    "look_at_pointed_toys": look_at_pointed_toys,
                    "respond_when_called": respond_when_called,
                    "keep_attention": keep_attention,
                    "interest_in_gadget": interest_in_gadget,
                    "behaviour": behaviour,
                    "parent_objective": parent_objective,
                    "gender": gender,
                    "level_asd_1": level_asd_1,
                    "level_asd_2": level_asd_2,
##                  "level_asd_3": level_asd_3,
                })))

            conn.commit()

        return redirect(url_for('profile'))
```

```
    return render_template('profile.html')

@app.route('/user_profiles')
def user_profiles():
    conn = sqlite3.connect('data.db')
    cursor = conn.cursor()
    cursor.execute("SELECT * FROM user_profile")
    users = cursor.fetchall()
    conn.close()
    return render_template('user_profiles.html', users=users)


import logging
logging.basicConfig(level=logging.DEBUG)

@app.route('/predict', methods=['POST'])
def predict():
    logging.info("Form data received: %s", request.form)

    # Ensure the expected form field names match those used in the HTML form
    expected_features = [f'field{i}' for i in range(18)]  # Adjusted for 'field0',
'field1', ... 'field17'

    user_data_dict = {}
    user_data_values = []
    errors = []

    # Validate inputs
    for feature in expected_features:
        value = request.form.get(feature)
        user_data_dict[feature] = value

        if not value or value.strip() == "":
            error_msg = f"Missing value for {feature}."
            errors.append(error_msg)
            logging.error(error_msg)
        else:
            try:
                user_data_values.append(float(value))
            except ValueError:
                error_msg = f"Invalid input for {feature}. Please enter a valid number."
            errors.append(error_msg)
            logging.error(error_msg)

    # If errors exist, flash them and redirect **only after validation**
    if errors:
        for error in errors:
            flash(error, "danger")
        logging.info("Redirecting to profile due to validation errors.")
        return redirect(url_for('profile'))  # Redirect **only once**

    # Proceed to prediction if no errors
    user_email = request.form.get('email')
    user_id = request.form.get('user_id')

    features = np.array(user_data_values).reshape(1, -1)

    try:
        prediction = stacked_model.predict(features)[0]
        logging.info("Prediction successful. Predicted class index: %s", prediction)
    except Exception as e:
        logging.error("Error during prediction: %s", str(e))
        flash("An error occurred while generating the prediction.", "danger")
        return redirect(url_for('profile'))

    class_names = ['All', 'Attention', 'Behaviour', 'Not Specified',
                   'Occupational', 'Sensory Integration', 'Social',
                   'Social Anxiety', 'Speech']

    predicted_class = class_names[int(prediction)]
    logging.info("Final predicted class: %s", predicted_class)

    # Store prediction in database
    try:
        conn = sqlite3.connect('data.db')
        cursor = conn.cursor()
        cursor.execute("INSERT INTO user_prediction_suggestion (email,
profile_data, prediction, suggestion) VALUES (?, ?, ?, ?)",
                       (user_email, json.dumps(user_data_dict), predicted_class, ""))
        conn.commit()
        conn.close()
        logging.info("Prediction stored in database successfully.")
    except Exception as e:
        logging.error("Error inserting into database: %s", str(e))

    # Render the correct results page
    return render_template('prediction_result.html',
                           prediction=predicted_class,
                           user_data=user_data_values,
                           email=user_email)

@app.route('/submit_suggestion', methods=['POST'])
def submit_suggestion():
    suggestion = request.form.get('suggestion')
    prediction = request.form.get('prediction')
    email = request.form.get('email')

    print(f"Suggestion: {suggestion}, Prediction: {prediction}, Email: {email}")  #
Debugging output

    if suggestion and email and prediction and suggestion.strip():
        try:
            conn = sqlite3.connect('data.db')
            cursor = conn.cursor()

            # Insert new suggestion
            cursor.execute("INSERT INTO suggestions (email, prediction,
suggestion) VALUES (?, ?, ?)",
                           (email, prediction, suggestion))

            conn.commit()  # Commit the transaction
            print("Suggestion stored successfully.")  # Confirmation message
        except Exception as e:
            print(f"Error occurred: {e}")  # Log the error
        finally:
            conn.close()  # Ensure the connection is closed
    else:
        print("Suggestion was empty or invalid.")

    return redirect(f"/my_suggestions?email={email}")

@app.route('/my_suggestions')
def my_suggestions():
    email = request.args.get('email')
    conn = sqlite3.connect('data.db')
    cursor = conn.cursor()

    # Fetch suggestions for the given email
    cursor.execute("SELECT prediction, suggestion FROM suggestions WHERE
email = ?", (email,))
    suggestions = cursor.fetchall()

    conn.close()

    print(suggestions)  # Log the fetched suggestions for debugging

    return render_template('my_suggestions.html', suggestions=suggestions)

# Function to get a database connection
def get_db_connection():
    conn = sqlite3.connect('data.db')  # Make sure the path is correct
    conn.row_factory = sqlite3.Row  # Allows for row-based dictionary access
    return conn

@app.route('/user_history', methods=['GET'])
def user_history():
    conn = get_db_connection()

    try:
        cursor = conn.cursor()

        cursor.execute("""
            SELECT suggestions.email,
                   user_profiles.profile_data,
                   suggestions.prediction,
                   suggestions.suggestion
            FROM suggestions
            LEFT JOIN user_profiles
                ON LOWER(suggestions.email) = LOWER(user_profiles.email)
        """)

        records = cursor.fetchall()

        if not records:
            flash("No prediction or suggestion history found.", "warning")

        return render_template('user_history.html', records=records)

    except sqlite3.Error as e:
        print(f"Database error: {e}")
        flash("An error occurred while fetching data.", "danger")
        return redirect(url_for('doctor_dashboard'))

    finally:
        conn.close()

@app.route('/view_history', methods=['GET'])
def view_history():
    patient_email = request.args.get('email')  # ✅ Get the patient's email from
URL

    if not patient_email:
        flash("No patient selected.", "danger")
        return redirect(url_for('doctor_dashboard'))

    conn = get_db_connection()

    try:
        cursor = conn.cursor()
        cursor.execute("""
            SELECT user_profiles.profile_data,
                   suggestions.prediction,
                   suggestions.suggestion
            FROM suggestions
            LEFT JOIN user_profiles
                ON LOWER(suggestions.email) = LOWER(user_profiles.email)
            WHERE LOWER(suggestions.email) = LOWER(?)
        """, (patient_email,))

        records = cursor.fetchall()

        if not records:
            flash("No history found for this patient.", "warning")

        return render_template('patient_history.html', records=records,
name=patient_email)

    except sqlite3.Error as e:
        print(f"Database error: {e}")
        flash("An error occurred while fetching data.", "danger")
        return redirect(url_for('doctor_dashboard'))

    finally:
        conn.close()

@app.route('/logout')
def logout():
    session.pop('username', None)
    flash('You have been logged out.', 'info')
    return redirect(url_for('login'))

import webbrowser

if __name__== '__main__':
    init_db()
    webbrowser.open('http://127.0.0.1:1050')
    app.run(debug=False,port=1050)
```

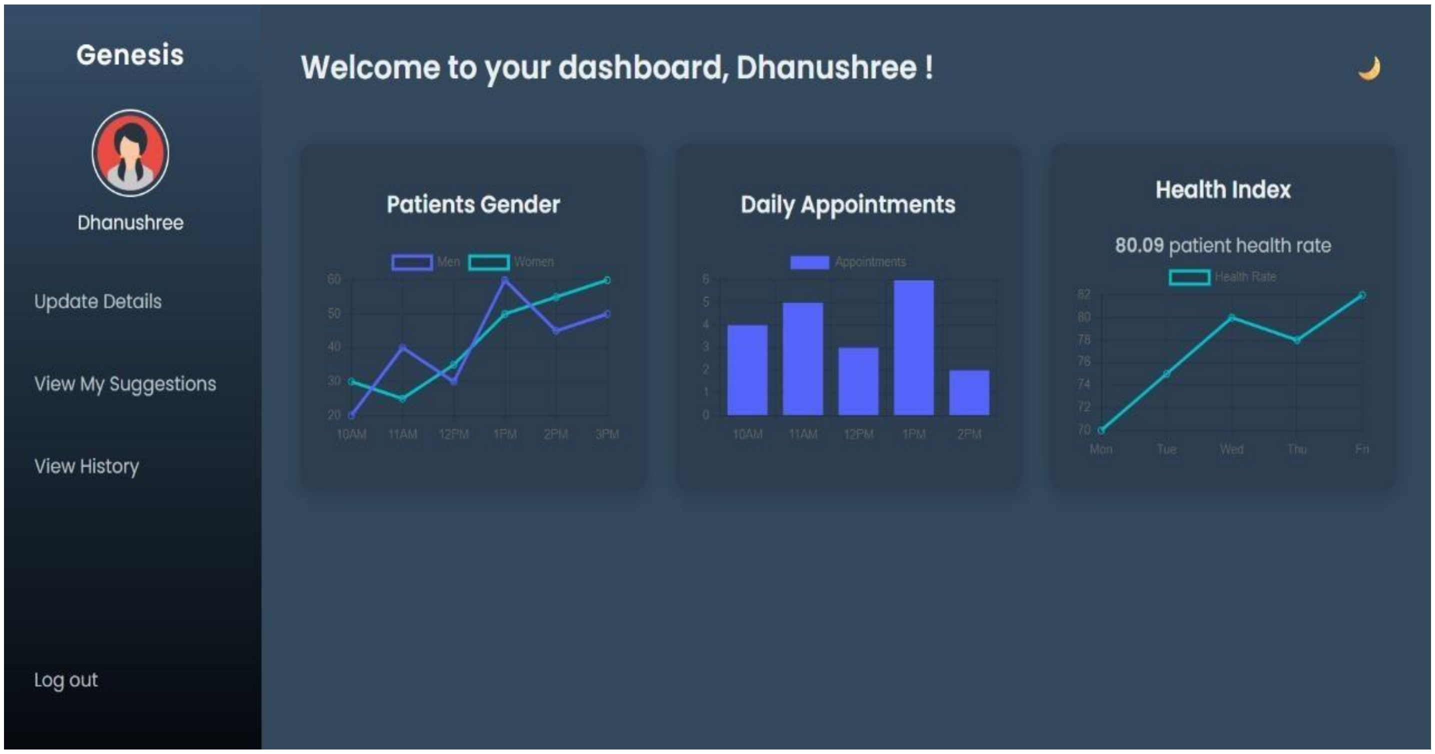
APPENDIX B- SNAPSHOTS


+463281626556
genesis@gmail.com
Genesis
UserRegister
UserLogin
Doctor Login
Doctor Register
Complete Autism Therapy Solution
Empowering families with accessible, personalized therapy options — all from the comfort of home.
Learn More

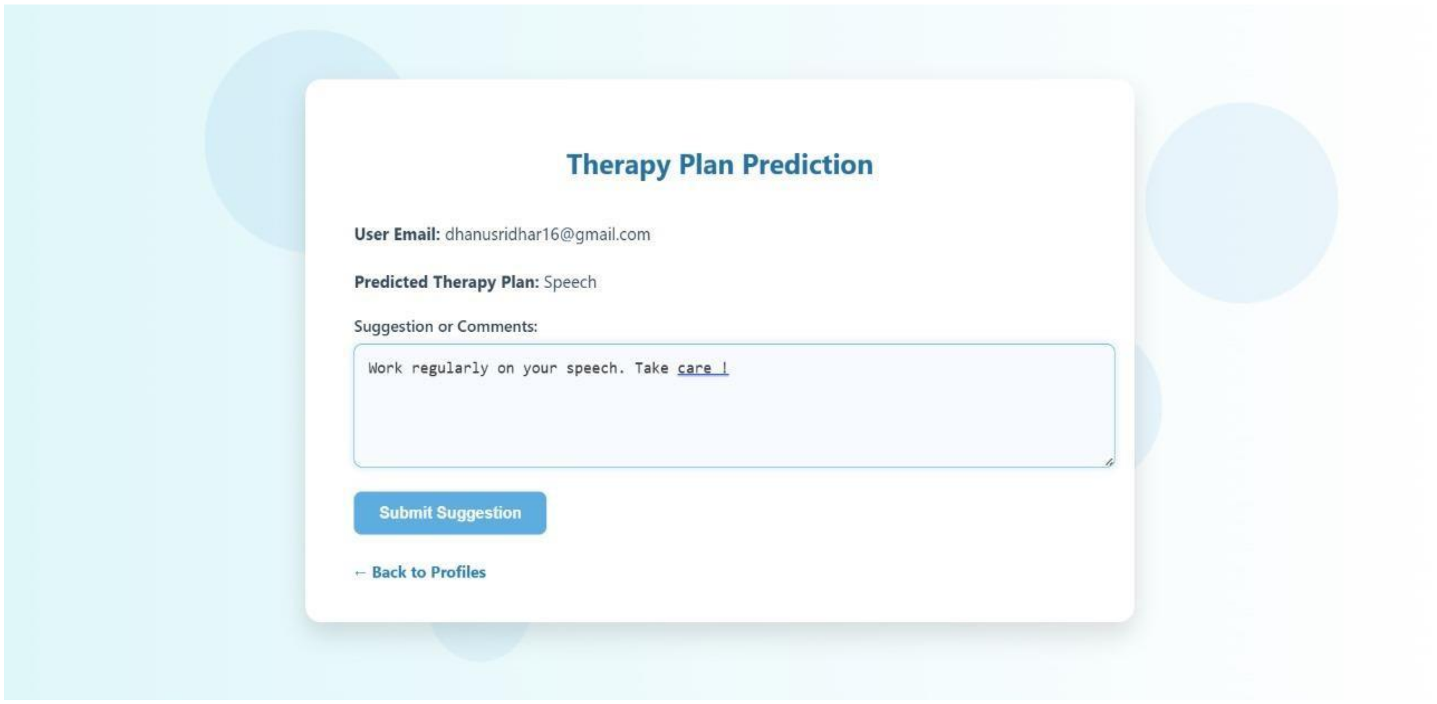
Login Page



Doctor Dashboard



Patient Dashboard



Therapy Plan Prediction

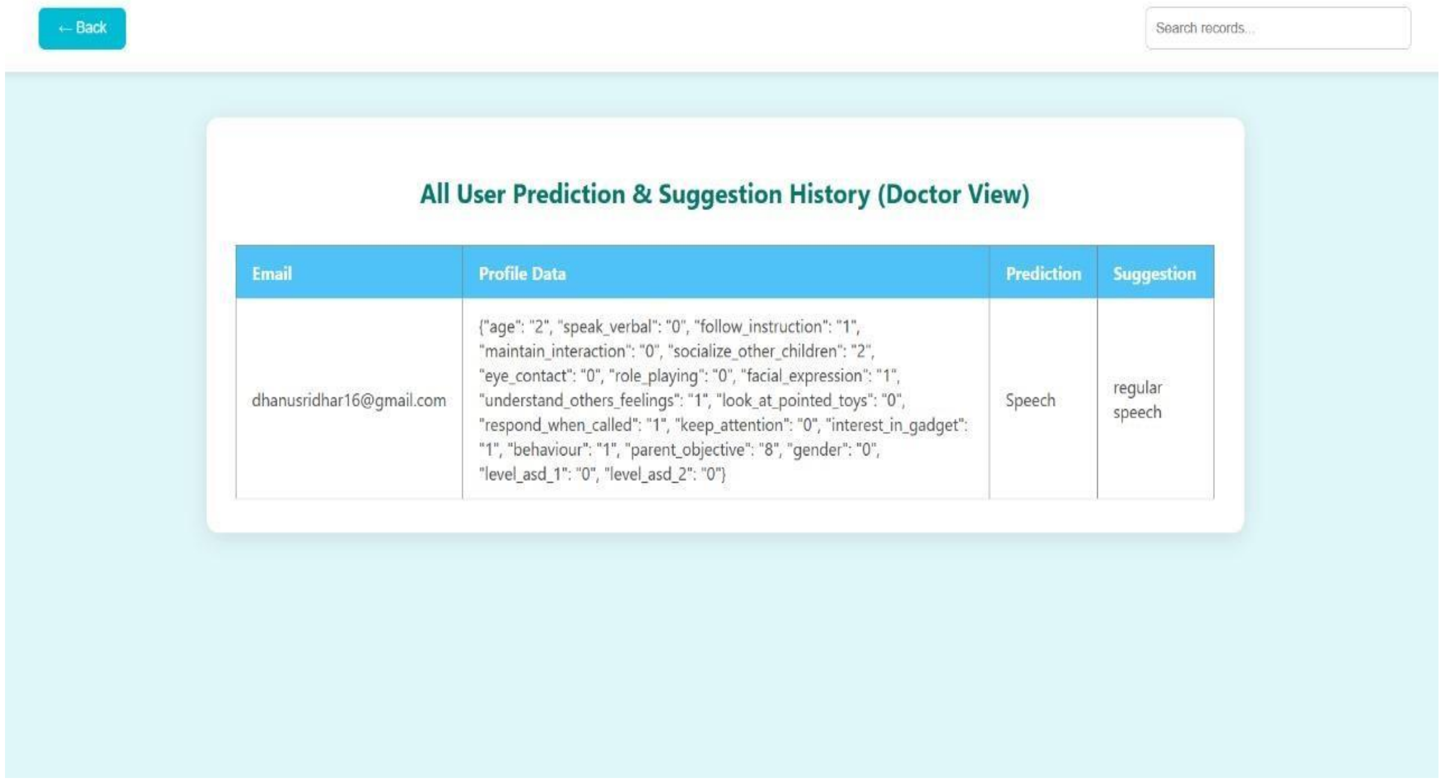
User Email: dhanusridhar16@gmail.com

Predicted Therapy Plan: Speech

Suggestion or Comments:

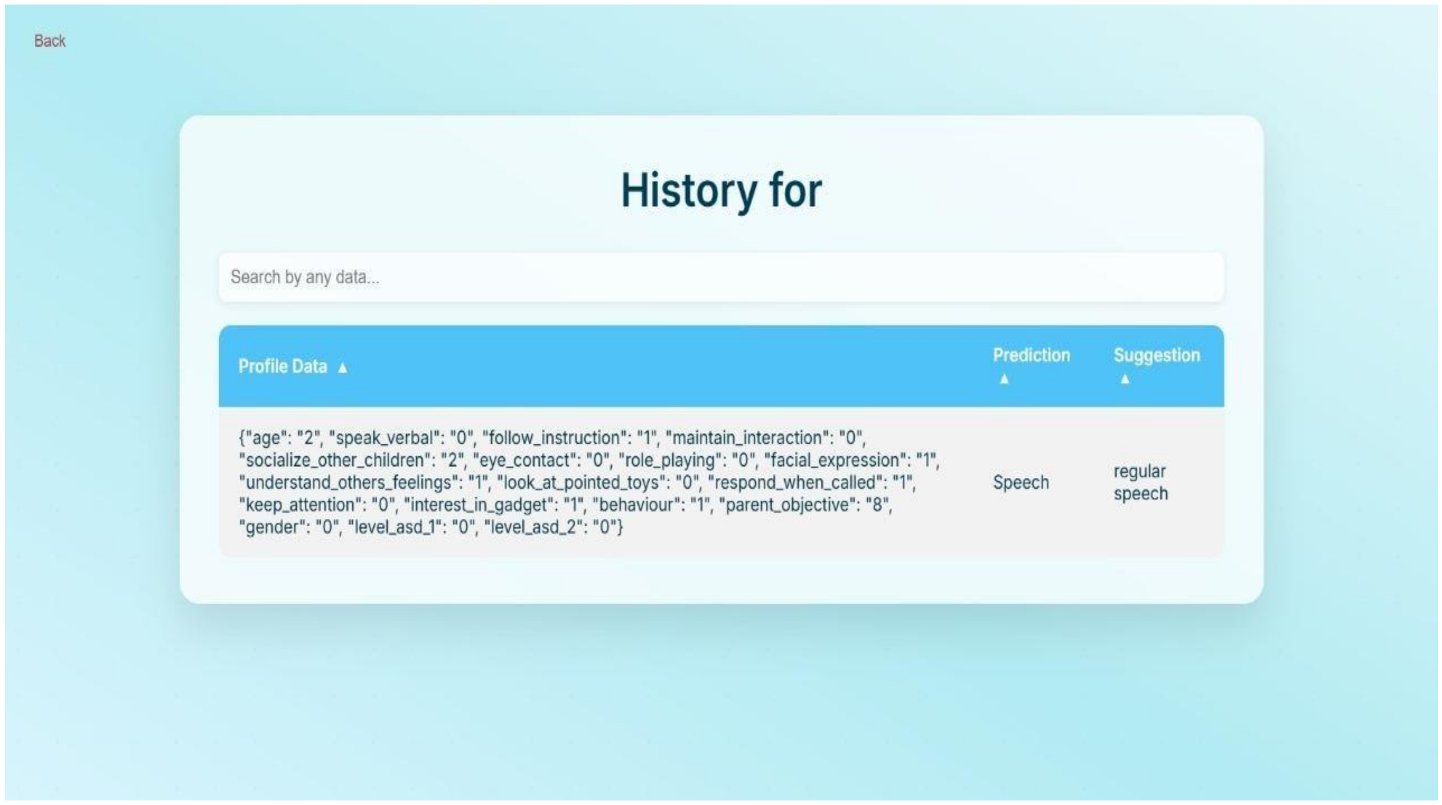
```
Work regularly on your speech. Take care !
```

Therapy Prediction



All User Prediction & Suggestion History (Doctor View)

Email	Profile Data	Prediction	Suggestion
dhanusridhar16@gmail.com	{"age": "2", "speak_verbal": "0", "follow_instruction": "1", "maintain_interaction": "0", "socialize_other_children": "2", "eye_contact": "0", "role_playing": "0", "facial_expression": "1", "understand_others_feelings": "1", "look_at_pointed_toys": "0", "respond_when_called": "1", "keep_attention": "0", "interest_in_gadget": "1", "behaviour": "1", "parent_objective": "8", "gender": "0", "level_asd_1": "0", "level_asd_2": "0"}	Speech	regular speech



History for

Profile Data ▲	Prediction ▲	Suggestion ▲
{"age": "2", "speak_verbal": "0", "follow_instruction": "1", "maintain_interaction": "0", "socialize_other_children": "2", "eye_contact": "0", "role_playing": "0", "facial_expression": "1", "understand_others_feelings": "1", "look_at_pointed_toys": "0", "respond_when_called": "1", "keep_attention": "0", "interest_in_gadget": "1", "behaviour": "1", "parent_objective": "8", "gender": "0", "level_asd_1": "0", "level_asd_2": "0"}	Speech	regular speech

CHAPTER 12

CHAPTER 12

REFERENCES

1. Carroll, L., Braeutigam, S., Dawes, J. M., Krsnik, Z., Kostovic, I., Coutinho, E., Dewing, J. M., Horton, C. A., Gomez-Nicola, D. and Menassa, D. A. (2020) 'Autism spectrum disorders: Multiple routes to, and multiple consequences of, abnormal synaptic function and connectivity', *Neuroscientist*, Vol. 27, No. 1, pp. 10–29. doi: 10.1177/1073858420921378.

2. D'Angelo, E., Mazzarello, P., Prestori, F., Mapelli, J., Solinas, S., Lombardo, P., Cesana, E., Gandolfi, D. and Congi, L. (2011) 'The cerebellar network: From structure to function and dynamics', *Brain Research Reviews*, Vol. 66, Nos. 1–2, pp. 5–15. doi: 10.1016/j.brainresrev.2010.10.002.

3. Ha, S., Sohn, I.-J., Kim, N., Sim, H. J. and Cheon, K.-A. (2015) 'Characteristics of brains in autism spectrum disorder: Structure, function and connectivity across the lifespan', *Experimental Neurobiology*, Vol. 24, No. 4, pp. 273–284. doi: 10.5607/en.2015.24.4.273.

4. Han, J., Jiang, G., Ouyang, G. and Li, X. (2022) 'A Multimodal Approach for Identifying Autism Spectrum Disorders in Children', *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 30, pp. 2003-2011. doi: 10.1109/TNSRE.2022.3192431.

5. Hernandez, L. M., Rudie, J. D., Green, S. A., Bookheimer, S. and Dapretto, M. (2014) 'Neural signatures of autism spectrum disorders: Insights into brain network dynamics', *Neuropsychopharmacology*, Vol. 40, No. 1, pp. 171–189. doi: 10.1038/npp.2014.172.

6. Jain, S., Tripathy, H. K., Mallik, S., Qin, H., Shaalan, Y. and Shaalan, K. (2023) 'Autism Detection of MRI Brain Images Using Hybrid Deep CNN With DM-Resnet Classifier', *IEEE Access*, Vol. 11, pp. 117741–117751. doi: 10.1109/ACCESS.2023.3325701.

7. Kohli, M., Kar, A. K. and Sinha, S. (2022) 'The Role of Intelligent Technologies in Early Detection of Autism Spectrum Disorder (ASD): A Scoping Review', *IEEE Access*, Vol. 10, pp. 104887–104913. doi: 10.1109/ACCESS.2022.3208587.

8. Paoletti, F. P., Simoni, S., Parnetti, L. and Gaetani, L. (2021) 'The contribution of small vessel disease to neurodegeneration: Focus on Alzheimer's disease, Parkinson's disease

and multiple sclerosis', *International Journal of Molecular Sciences*, Vol. 22, No. 9, p. 4958. doi: 10.3390/ijms22094958.

9. Wang, M., Guo, J., Wang, Y., Yu, M. and Guo, J. (2023) 'Multimodal Autism Spectrum Disorder Diagnosis Method Based on DeepGCN', *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 31, pp. 3664–3674. doi: 10.1109/TNSRE.2023.3314516.

10. Yang, R., Ke, F., Liu, H., Zhou, M. and Cao, H.-M. (2021) 'Exploring sMRI Biomarkers for Diagnosis of Autism Spectrum Disorders Based on Multi Class Activation Mapping Models', *IEEE Access*, Vol. 9, pp. 124122–124131. doi: 10.1109/ACCESS.2021.3069211.