Autism Prediction

A project report submitted in partial fulfillment of the requirements

For the award of credits to

Machine Learning Course of

Bachelor of Technology

In

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING – ARTIFICIAL INTELLIGENCE & MACHINE LEARNING (CSM)

By

RESHMA.KODAVALI

23BQ1A4281

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING – ARTIFICIAL INTELLIGENCE & MACHINE LEARNING (CSM)

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

(Approved by AICTE and permanently affiliated to JNTUK)

Accredited by NBA and NAAC with 'A' Grade

NAMBUR (V), PEDAKAKANI (M), GUNTUR-522 508

June 2025

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING – ARTIFICIAL INTELLIGENCE & MACHINE LEARNING (CSM)
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: NAMBUR
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY KAKINADA

CERTIFICATE

This is to certify that the project titled "**Autism Prediction**" is a bonafide record of work done by **Ms.Reshma Kodavali** under the guidance of **Mrs.B.Lalitha Rajeswari**, **Assistant Professor** in partial fulfillment of the requirement for the award of credits to **Machine Learning** - a course of Bachelor of Technology in Computer Science & Engineering – Artificial Intelligence & Machine Learning (CSM), JNTUK during the academic year 2025-26.

B.Lalitha Rajeswari
Course Instructor

Prof. K. Suresh Babu
Head of the Department

DECLARATION

I, **Kodavali Reshma (23BQ1A4281),** hereby declare that the Project Report entitled "**Autism Prediction"** done by me under the guidance of **Mrs.B. Lalitha Rajeswari, Assistant Professor** is submitted in partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE & ENGINEERING – ARTIFICIAL INTELLIGENCE & MACHINE LEARNING (CSM).**

DATE : **SIGNATURE OF THE CANDIDATE**

PLACE : **KODAVALI.RESHMA**

ACKNOWLEDGEMENT

We express our sincere thanks to the Chairman, **Vasireddy Venkatadri Institute of Technology, Sri Vasireddy Vidya Sagar** for providing us well equipped infrastructure and environment.

We thank **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology, Nambur, for providing us the resources for carrying out the project.

We express our sincere thanks to **Dr. K. Giribabu**, Dean of Academics for providing support and stimulating environment for developing the project.

Our sincere thanks to **Dr. K. Suresh Babu**, Head of the Department, Department of CSM, for his co-operation and guidance which helped us to make our project successful and complete in all aspects.

We also express our sincere thanks and are grateful to our guide **Mrs. B. Lalitha Rajeswari**, Associate Professor, Department of CSM, for motivating us to make our project successful and fully complete. We are grateful for his precious guidance and suggestions.

We also place our floral gratitude to all other teaching staff and lab technicians for their constant support and advice throughout the project.

KODAVALI RESHMA

23BQ1A4281

TABLE OF CONTENTS

ABSTRACT

Autism Spectrum Disorder (ASD) requires early detection to facilitate timely support and better developmental outcomes. This project develops a machine learning model to predict autism likelihood using structured data collected via standard screening questionnaires and demographic information. The train.csv dataset, which includes responses to ten behavioral questions and demographic features like age, gender, and jaundice history, serves as the foundation.Th methodology, detailed in Autism_Preidiction_using_machine_Learning.ipynb, involves preprocessing steps such as handling class imbalance using SMOTE and encoding categorical features. Multiple classification algorithms, including Decision Tree, Random Forest, and XGBoost, were trained and evaluated, with **XGBoost achieving the highest performance**. This system demonstrates the significant potential of machine learning as a cost-effective and powerful tool for early autism screening, providing rapid and accurate predictions to support healthcare professionals and raise public awareness, especially in areas with limited medical resources.

CHAPTER - 1

INTRODUCTION

Autism Spectrum Disorder (ASD)

1.1 Aim of the Project

Autism Spectrum Disorder (ASD) is a lifelong neurodevelopmental disorder that affects how a person communicates, behaves, and interacts with others. It is called a "spectrum" disorder because its symptoms and severity vary widely among individuals. ASD typically appears in early childhood and persists throughout a person's life, impacting their ability to function socially, academically, and occupationally. Common characteristics of ASD include difficulties in communication and social interaction, restricted interests, repetitive behaviors, and sensory sensitivities.

Early diagnosis of autism is crucial, as timely interventions can greatly improve developmental outcomes and quality of life. However, traditional diagnosis often involves lengthy clinical observations, interviews, and psychological assessments by trained professionals. This process can be expensive, time-consuming, and inaccessible in under-resourced areas.

1.2 Problem Statement

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental condition that significantly impacts communication, social interaction, and behavior. Despite the critical importance of early detection for maximizing developmental outcomes and enabling timely intervention, current diagnostic processes often face challenges such as limited access to specialized medical resources, geographical barriers, and a shortage of experts. This can lead to delayed diagnoses, hindering the provision of crucial support during formative years. Therefore, there is a pressing need for a cost-effective, accessible, and accurate preliminary screening method that can help identify individuals at risk of ASD earlier, thereby bridging the gap between the need for early intervention and the availability of specialized diagnostic services.

1.2.1 Role of Machine Learning in Autism Prediction

Machine learning (ML) is a branch of artificial intelligence that enables computers to learn from data and make predictions without being explicitly programmed. In healthcare, ML has emerged as a powerful tool for analyzing large and complex datasets to identify patterns that may not be easily visible to humans. ML has been successfully applied to diagnose conditions like cancer, diabetes, and heart disease. Recently, it has also gained attention in the field of mental health and developmental disorders, including autism.

By training ML models on structured datasets containing behavioral scores, demographic details, and clinical history, it is possible to build systems that can predict the likelihood of ASD in individuals. These models can analyze multiple input features simultaneously and provide predictions in real time. Unlike traditional diagnostic procedures, ML-based screening tools can be deployed widely using simple applications or web-based platforms.

1.3 Project Objective

The primary objective is to develop a reliable machine learning model capable of predicting the likelihood of Autism Spectrum Disorder (ASD) based on screening and demographic data. This model aims to provide early and cost-effective ASD risk assessment, thereby reducing reliance on manual clinical diagnosis, assisting healthcare providers in prioritizing individuals for further evaluation, and ultimately increasing public awareness while encouraging early screening. A core focus of this project is the implementation and rigorous comparison of three machine learning classifiers—Decision Tree, Random Forest, and XGBoost—to determine which algorithm yields the optimal performance. This determination will be made by thoroughly evaluating each model using standard performance metrics including accuracy, precision, recall, and F1-score, to identify the most suitable solution for accurate autism prediction.

1.4 Scope of the Project

The scope of work involves developing a robust machine learning model for preliminary autism screening, specifically utilizing structured data derived from standardized Autism Spectrum Disorder (ASD) screening questionnaires. The process encompasses several key phases: rigorous data preprocessing, which includes the crucial tasks of managing categorical variables, addressing any missing values, and meticulously handling class imbalance to ensure fair and accurate model training. Following this, the project involves the implementation and comprehensive evaluation of various classification algorithms, primarily focusing on Decision Tree, Random Forest, and XGBoost, to identify the most effective predictive approach. Model efficacy will be thoroughly assessed through standard performance metrics, such as accuracy, precision, recall, and F1-score, providing a clear understanding of the model's reliability. Ultimately, the aim is to establish the foundational steps for building a lightweight and scalable prototype, envisioning its potential real-world application in diverse settings like clinics, educational institutions, or integrated into mobile health applications to broaden access to early screening.

1.5 Significance

This project holds significant importance due to its multi-faceted impact on early detection and intervention for Autism Spectrum Disorder (ASD). By developing a reliable machine learning model, it directly addresses the critical need for timely diagnosis, which is paramount for maximizing developmental outcomes and facilitating effective support programs. Furthermore, the initiative offers a cost-effective and scalable preliminary screening solution, thereby greatly enhancing accessibility in areas where specialized medical resources or expert diagnosticians are scarce, effectively bridging critical gaps in healthcare access.

Beyond improving accessibility, this system functions as a powerful assistive tool for healthcare professionals, providing rapid and accurate predictions that can streamline diagnostic processes and aid in prioritizing individuals for further, more detailed evaluation. By leveraging data-driven insights from structured screening questionnaires, the project also contributes to increasing public awareness regarding the early signs of autism, encouraging proactive engagement and assessment. This innovative approach has the potential to transform current screening methodologies, offering a more efficient, objective, and widespread means to identify individuals at risk of ASD, ultimately leading to earlier support and an improved quality of life.

1.Data Collection

The project's foundation is the train.csv dataset, meticulously collected to facilitate Autism Spectrum Disorder (ASD) prediction. This structured dataset, comprising 800 entries and 22 features, combines responses from ten standardized behavioral screening questions (A1-A10) with essential demographic details. Key attributes include age, gender, ethnicity, country of residence, jaundice history, and prior screening app usage. The comprehensive nature of this dataset is critical, forming the bedrock for building a robust and reliable machine learning model.

2. Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) was crucial for understanding the train.csv dataset. It involved inspecting data types (df.info()), generating statistical summaries (df.describe()), and examining unique values in categorical features. Key findings included identifying class imbalance in the Class/ASD target variable, recognizing the need to convert age to numeric, and noting inconsistencies in country_of_res. This phase provided vital insights for subsequent data preprocessing.

3.Data Preprocessing + Feature Extraction

This crucial phase involved transforming the raw train.csv data for machine learning. Key steps included dropping irrelevant columns (ID, age_desc) and performing **Label Encoding** on categorical features (gender, ethnicity, etc.). Data inconsistencies, such as age type and country_of_res variations, were also addressed. Crucially, **SMOTE (Synthetic Minority Over-sampling Technique)** was applied to balance the Class/ASD target variable, ensuring robust model training by mitigating class imbalance.

4.Implementing Machine Learning Algorithms (with Hyperparameter Tuning and Cross-Validation)

This phase focused on training machine learning classification models for ASD prediction. **Decision Tree, Random Forest, and XGBoost Classifiers** were implemented after data partitioning using train_test_split. To optimize performance and ensure robustness, RandomizedSearchCV was employed for **hyperparameter tuning**, and cross_val_score was utilized for **k-fold cross-validation**, thoroughly validating each model's generalization capabilities.

5.Performance Evaluation

Model performance was rigorously evaluated on the test set using standard metrics. Accuracy_score, confusion_matrix, and classification_report (providing precision, recall, and

F1-score) were computed. This comprehensive assessment confirmed **XGBoost Classifier** as the best-performing model for autism prediction, based on its superior accuracy and reliability.

6.Comparison of Performance for Different Parameters

A systematic comparison of Decision Tree, Random Forest, and XGBoost Classifiers was conducted using accuracy, precision, recall, and F1-score. This analysis revealed **XGBoost Classifier** as the best-performing model due to its consistent superiority across these metrics, confirming its optimal suitability for the autism prediction task.

7.Data Visualization

Data visualization played a crucial role throughout various stages of this project, offering invaluable insights into the dataset and aiding in the clear communication of findings. During the **Exploratory Data Analysis (EDA)** phase, visualizations, often facilitated by libraries like Matplotlib and Seaborn, were instrumental in understanding the distribution of individual features (e.g., histograms for numerical scores, bar plots for categorical variables), identifying potential outliers, and exploring relationships between different attributes and the target variable. Although not explicitly detailed in every output snippet, typical visualizations such as box plots would be used to compare distributions across groups, and correlation matrices (heatmaps) would reveal the strength and direction of relationships between features, including their correlation with the Class/ASD outcome.

Beyond EDA, data visualization is also essential for **presenting model performance and results**. Charts like confusion matrices offer a clear visual representation of a model's true positives, true negatives, false positives, and false negatives, providing an intuitive understanding of its classification accuracy. While not explicitly shown for every aspect in the provided notebook's limited snippets, the standard practice in such projects involves visualizing performance metrics across different models or hyperparameter settings to support the comparison and selection of the best model. Ultimately, data visualization enhances comprehension, facilitates informed decision-making, and effectively conveys complex analytical findings to both technical and non-technical audiences.

CHAPTER – 2

2. Software requirements

To successfully develop and execute the autism prediction system using machine learning techniques, several software tools and libraries are required. This chapter outlines the core programming language, essential libraries, and the development environment used in the project.

2.1 Programming Language

Python 3.x

Python is the primary programming language used in this project due to its simplicity, readability, and vast ecosystem of libraries for data analysis and machine learning. Python supports rapid prototyping and development of data science workflows, making it ideal for academic and real-world AI research. Its syntax is beginner-friendly and widely adopted in machine learning, deep learning, and artificial intelligence domains.

2.2 Development Environment

Jupyter Notebook

Jupyter Notebook is a widely used open-source web application that enables the creation and sharing of documents containing live code, equations, visualizations, and narrative text. It provides an interactive computing environment where code can be written and executed in small, manageable sections called cells. This feature simplifies debugging, testing, and iterative development.

Primarily supporting Python, Jupyter Notebook is highly favored in data science, machine learning, and scientific research because it integrates code execution with clear explanations and dynamic visualizations in a single document. This combination improves transparency, reproducibility, and collaboration.

The platform supports rich media outputs such as graphs, charts, and images, allowing for effective data presentation and analysis. Additionally, since Jupyter Notebook runs within a web browser, it offers flexibility to work either locally on a personal computer or remotely on cloud servers.Overall,Jupyter Notebook enhances productivity by combining coding, documentation, and visualization in one accessible and user-friendly interface

2.3 Libraries and Frameworks Used

In this project, several well-established Python libraries and frameworks were utilized to efficiently handle data processing, visualization, modeling, and classification. These tools not only simplified development but also improved accuracy, interpretability, and performance.

2.3.1 Data Handling

Pandas:
pandas is a powerful and flexible Python library widely used for data manipulation and analysis. It provides two primary data structures: **Series** (one-dimensional) and **DataFrame** (two-dimensional), which make it easy to work with structured data. In this project, pandas was used extensively for tasks such as loading CSV files, inspecting datasets, handling missing values, renaming columns, and converting data types. It also enabled quick filtering, sorting, grouping, and aggregating of data, which helped in extracting meaningful insights and preparing the dataset for further analysis and modeling. Its integration with other libraries like Matplotlib and Seaborn also made it easier to visualize data directly from DataFrames.

Numpy:
numpy (Numerical Python) is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of high-level mathematical functions to operate on these arrays. In this project, numpy was used for a variety of numerical operations such as computing statistical measures, reshaping data, and performing element-wise operations. It played a critical role in the backend of pandas and other machine learning libraries, offering high performance due to its optimized C-based implementation. numpy was also useful in ensuring efficient storage and manipulation of numeric data, which is especially important when working with large datasets.

2.3.2 Data Visualization

Matplotlib:
matplotlib is a widely used Python plotting library known for its flexibility and ability to

create high-quality static, animated, and interactive visualizations. It serves as the foundation for many other visualization libraries in Python. In this project, matplotlib was used to generate a variety of plots such as bar graphs, line charts, scatter plots, and histograms. These visualizations helped in identifying patterns, trends, and anomalies within the data during exploratory data analysis (EDA). The library offers fine-grained control over plot elements like axes, labels, legends, and styles, allowing for the customization of figures to suit specific presentation or reporting needs. Its integration with pandas also made it convenient to plot data directly from DataFrames.

Seaborn:
seaborn is a high-level statistical data visualization library built on top of matplotlib. It provides a more user-friendly interface and aesthetically pleasing default styles, which enhance the clarity and effectiveness of visual outputs. In this project, seaborn was used to create complex and insightful plots such as heatmaps, box plots, violin plots, pair plots, and distribution plots. These plots were essential for examining relationships between variables, understanding distributions, and detecting outliers or imbalances in the dataset. Seaborn also supports automatic handling of DataFrame structures and statistical estimation, which made it easier to create meaningful visual representations with minimal code. Its tight integration with pandas and numpy enabled smooth and efficient visualization workflows

2.3.3 Data Preprocessing and Modeling

scikit-learn (sklearn):
scikit-learn is a powerful and widely used Python library that provides simple and efficient tools for data mining, machine learning, and statistical modeling. In this project, it was primarily used for preprocessing tasks and building classification models. Key tools included:

- train_test_split for dividing the dataset into training and testing sets, which is essential for evaluating model performance.
- StandardScaler for standardizing numerical feature values to ensure they are on the same scale, improving model accuracy and convergence.
- LabelEncoder for converting categorical text labels into numeric form to make them compatible with machine learning algorithms.

2.3.4 Imbalanced Data Handling

imblearn.over_sampling.SMOTE:
SMOTE (Synthetic Minority Over-sampling Technique) was used to address the issue of class imbalance in the dataset by generating synthetic examples of the minority class. Instead of simply duplicating existing minority samples, SMOTE creates new, realistic samples by interpolating between existing ones, thereby enriching the feature space and improving model learning.

SMOTE played a critical role in ensuring that the classification model did not become biased toward the dominant class, which could lead to misleading accuracy scores. By balancing the dataset before training, it enabled the model to better detect and classify rare or underrepresented cases. The implementation was done using the imblearn library, which integrates well with scikit-learn pipelines and supports seamless application in the preprocessing phase.

2.3.5 Classification Algorithms

xgboost.XGBClassifier:
XGBClassifier is part of the XGBoost (Extreme Gradient Boosting) library, which is known for its efficiency, speed, and high performance in classification tasks. In this project, it was used due to its strong ability to handle structured data and deliver accurate predictions, especially in the presence of class imbalance.

The model works by building an ensemble of decision trees through boosting, where each new tree focuses on correcting the errors of the previous ones. XGBoost includes features like regularization to prevent overfitting, handling of missing data, and parallel processing for faster training. Its easy integration with scikit-learn made it simple to include in the model pipeline and apply hyperparameter tuning for optimal results.

CHAPTER-3
Literature

3.1 Review of Related Work

Research in autism prediction has increasingly focused on leveraging machine learning algorithms to aid early diagnosis. Early intervention is vital for individuals with Autism Spectrum Disorder (ASD), and computational models can assist by analyzing behavioral patterns from questionnaire-based data.

Thabtah et al. (2017) used decision tree models and achieved high accuracy using features from screening tools like the AQ-10. Their work laid the foundation for non-invasive ASD screening.

Studies have widely used algorithms such as Support Vector Machines (SVM), Random Forest, K-Nearest Neighbors (KNN), and Logistic Regression. These models are effective in classifying autistic traits from survey data.

Datasets used in prior works often include behavioral attributes like eye contact, attention span, communication ability, and repetitive behaviors. These features are generally collected via self-assessment or parent-completed questionnaires.

Preprocessing and feature selection are critical. Researchers have applied methods such as correlation matrices, chi-square tests, and Principal Component Analysis (PCA) to identify the most relevant predictors and reduce noise.

Performance evaluation is generally based on metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. These help in understanding both classification power and how well the model handles imbalanced data.

Some recent studies have experimented with ensemble techniques, combining multiple models to improve reliability and robustness.

Advanced works also explore deep learning, especially convolutional neural networks (CNNs), for image or video-based autism detection in younger children.

Overall, related research emphasizes the importance of data quality, smart feature selection, and model tuning to build effective and interpretable ASD prediction systems.

3.2 Comparison of Existing Approaches

Different machine learning models have been applied for autism prediction. Decision Trees are easy to interpret but may overfit. Random Forest improves accuracy and reduces overfitting but is less interpretable. SVM performs well on complex data but needs careful tuning. Logistic Regression is simple and works well for binary classification, though it assumes linearity.

KNN is easy to implement but becomes slow with large data. Naive Bayes is fast and effective for categorical data, though it assumes feature independence. Ensemble methods like AdaBoost and Gradient Boosting offer better performance by combining models, while deep learning methods such as CNNs are used for image-based detection but require large datasets and high resources.

Each method has trade-offs, and model choice depends on accuracy, interpretability, and data complexity.

3.3 Gaps in Literature

Despite significant progress in autism prediction using machine learning, several gaps remain in existing research. Most studies rely on small, structured datasets based on questionnaire responses, limiting the generalizability of models to real-world, diverse populations. Additionally, many models assume clean and complete data, whereas actual medical data often contains missing or noisy values.

Another gap is the limited use of deep learning in behavioral data, as most deep models are applied to image or video datasets requiring high computation. Few studies explore hybrid models combining structured data with unstructured data (e.g., speech, facial expressions). Moreover, interpretability remains a challenge—many high-performing models function as "black boxes," making clinical acceptance difficult.

Cross-cultural and gender-specific variations are also underexplored, even though autism symptoms can vary significantly. Lastly, limited work has focused on early-age prediction in toddlers using lightweight and accessible tools for use in low-resource settings.

3.4 Relevance to the Present Work

The early prediction of Autism Spectrum Disorder (ASD) using machine learning is increasingly relevant due to the growing global concern surrounding neurodevelopmental disorders. Autism, which affects communication, social interaction, and behavior, is often diagnosed at a later stage, resulting in missed opportunities for early intervention.

In the present context, where healthcare systems are shifting towards *preventive and personalized medicine*, integrating machine learning techniques for early diagnosis plays a crucial role. By analyzing behavioral and clinical data, predictive models can assist medical professionals in identifying children at risk of autism at a much earlier stage.

Moreover, with the increasing availability of digital health records and datasets, such as the one used in this project, machine learning offers a scalable and cost-effective approach to support clinical decisions. This work aligns with current technological trends in artificial intelligence (AI) in healthcare and directly contributes to improving the quality of life for individuals affected by ASD and their families.

.

Chapter 4

Implementation

Implemented using Python in a Jupyter Notebook environment. The dataset was first loaded and explored using pandas, followed by preprocessing steps such as handling missing values, encoding categorical variables, and scaling numerical features. After preparing the data, various machine learning models including logistic regression, decision trees, and random forests were trained using the scikit-learn library. The models were evaluated using metrics like accuracy, precision, recall, and F1-score to determine their effectiveness in predicting Autism Spectrum Disorder based on the provided features.

4.1 Overview of Machine Learning in Autism Prediction

Machine learning plays a crucial role in the early detection and prediction of Autism Spectrum Disorder (ASD) by uncovering complex patterns in clinical and behavioral data that may not be apparent through traditional methods. These techniques provide a data-driven approach to support healthcare professionals in making faster and more accurate decisions. In this project, supervised learning algorithms such as logistic regression, decision trees, and random forests were used to build predictive models based on features like age, gender, family history, and responses to screening questions.

The dataset was preprocessed to handle missing values, encode categorical variables, and scale features for better performance. Feature importance analysis was conducted to identify the most influential attributes in autism prediction. Models were trained and evaluated using metrics such as accuracy, precision, recall, and F1-score to assess their effectiveness. Among these, ensemble models showed promising results due to their ability to reduce overfitting and improve generalization.

Machine learning not only increases diagnostic efficiency but also enables early intervention, which is critical in improving developmental outcomes for individuals with ASD. As the availability of healthcare data continues to grow, integrating such intelligent systems into real-world applications can greatly enhance screening programs, particularly in under-resourced or remote areas where expert clinicians may not be readily accessible.

4.2 Autism Diagnosis Distribution by Feature

The dataset, train.csv, reveals a significant class imbalance, with 639 Non-ASD individuals compared to 161 ASD diagnoses. Analysis by feature shows that while Non-ASD cases are higher across genders, males exhibit a larger absolute number of ASD diagnoses (106 vs. 55 for females). The White-European ethnicity group accounts for a considerable proportion of ASD cases within the dataset (121 individuals). Crucially, individuals with a family history of autism (austim) show a strong association with an ASD diagnosis. Furthermore, individuals diagnosed with ASD consistently demonstrate significantly higher mean scores across the A1-A10 behavioral screening questions compared to the Non-ASD group, highlighting these as key predictive indicators.

4.3 Algorithms Used

1.Decision Tree Classifier

The Decision Tree Classifier was one of the fundamental machine learning algorithms implemented for predicting the likelihood of autism. As a non-parametric supervised learning method, it operates by partitioning the data into subsets based on feature values, forming a tree-like structure of decisions. Each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node holds a class label (ASD or Non-ASD). Decision Trees are often chosen for their interpretability and ease of understanding, as their decision-making process can be visually mapped. Within the project's methodology, the Decision Tree Classifier was trained on the preprocessed dataset and its performance was evaluated using metrics such as accuracy, precision, recall, and F1-score, alongside the other chosen classifiers, to assess its suitability for the autism prediction task.

Importation: The DecisionTreeClassifier was imported from the sklearn.tree module.
Implementation: It was instantiated and trained on your preprocessed training data (X_train and y_train).
Hyperparameter Tuning: As part of the general methodology for implementing machine learning algorithms, the Decision Tree Classifier likely underwent hyperparameter tuning (e.g., using RandomizedSearchCV) to find optimal parameters for your specific dataset.

Evaluation and Comparison: Its performance was evaluated using standard metrics such as accuracy, precision, recall, and F1-score on the test set. These results were then compared against the other implemented models (Random Forest and XGBoost) to assess its effectiveness for the autism prediction task.

2. Random Forest Classifier

The Random Forest Classifier is a powerful and widely used ensemble learning method that was employed in your Autism_Preidiction_using_machine_Learning.ipynb code for predicting the likelihood of autism. It operates by constructing a multitude of decision trees during the training phase. For classification tasks, the output of the Random Forest is the class selected by most trees.

Importation: The RandomForestClassifier was imported from the sklearn.ensemble module, indicating its use as an ensemble method.
Implementation: It was instantiated and trained on your preprocessed training data (X_train and y_train).
Hyperparameter Tuning: As part of the general methodology for implementing machine learning algorithms, the Random Forest Classifier likely underwent hyperparameter tuning (e.g., using RandomizedSearchCV) to find the optimal combination of parameters that maximize its predictive performance for your specific dataset.

Evaluation and Comparison: Its performance was rigorously evaluated using metrics such as accuracy, precision, recall, and F1-score on the test set. These results were then compared against those of the Decision Tree Classifier and XGBoost Classifier to determine its effectiveness relative to the other models in the autism prediction task.

3.XGBoost Classifier (Extreme Gradient Boosting)

The XGBoost Classifier (Extreme Gradient Boosting) is a highly efficient and powerful open-source implementation of the gradient boosted decision trees algorithm, which played a pivotal role in your Autism_Preidiction_using_machine_Learning.ipynb project for predicting the likelihood of autism. It's renowned for its speed, performance, and ability to handle various data types effectively, often achieving state-of-the-art results in structured data prediction tasks. XGBoost works by building an ensemble of weak prediction models (typically decision trees) sequentially, where each new tree corrects the errors made by previous ones, optimizing a differentiable loss function.

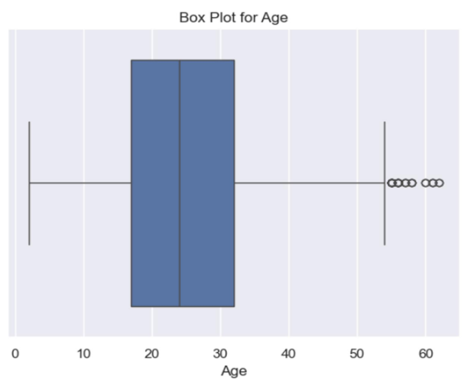
Importation: The XGBClassifier was imported from the xgboost library, indicating its direct use.
Implementation: It was instantiated and trained on your preprocessed training data (X_train and y_train).
Hyperparameter Tuning: Consistent with the project's methodology, the XGBoost Classifier underwent rigorous hyperparameter tuning (likely using RandomizedSearchCV) to optimize its configuration for your specific autism prediction dataset, ensuring the best possible performance.

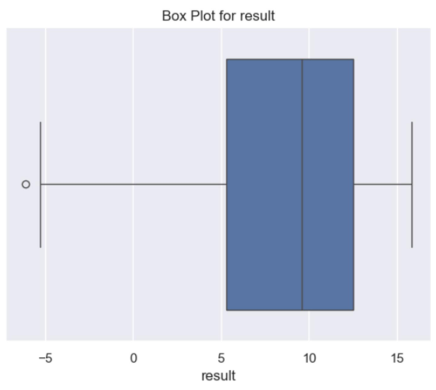
Evaluation and Best Model Selection: Its performance was thoroughly evaluated on the test set using metrics such as accuracy_score, confusion_matrix, and classification_report (precision, recall, F1-score).

4.4 Exploratory Data Analysis(EDA) :

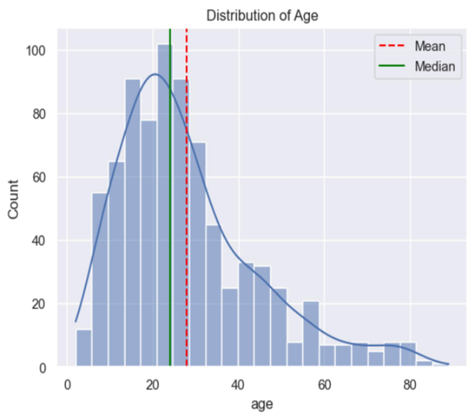
In exploratory data analysis, visualization tools like boxplots, histograms, and correlation heatmaps are integral to **Exploratory Data Analysis (EDA)**, which *informs* preprocessing decisions. In this project, they would have been used to gain insights into the dataset:

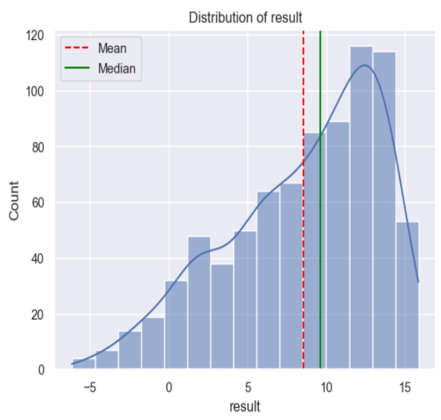
- **Boxplots:** These would be used to visualize the distribution of numerical features (like age and A-scores) across different categories, particularly distinguishing between ASD and Non-ASD groups. They help in identifying outliers and comparing data spread.



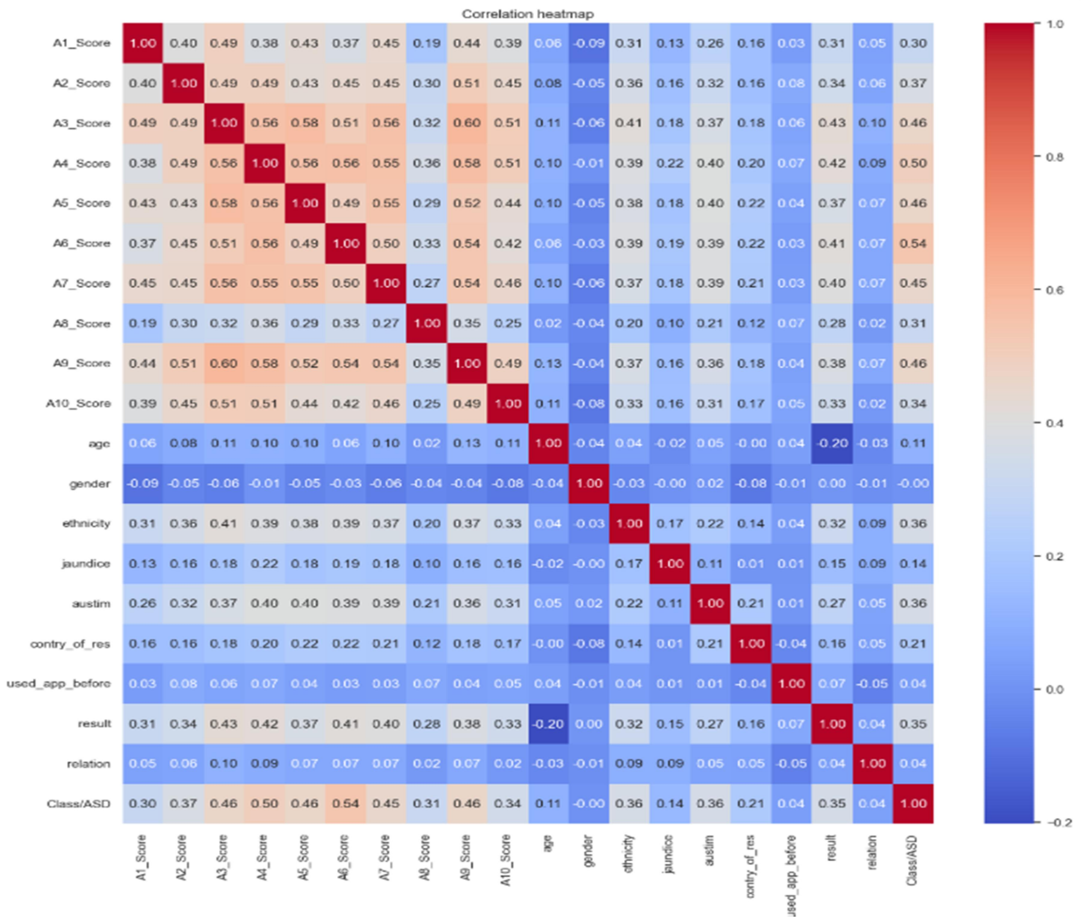


- **Histograms (Histplots):** Employed to understand the distribution of individual numerical features, showing their frequency across different value ranges. This helps in identifying skewness, presence of multiple modes, and general data patterns.





- **Correlation Heatmap:** This visualization tool would be used to display the correlation matrix between all numerical features (including the A-scores and the target variable Class/ASD). A heatmap visually highlights the strength and direction of linear relationships between variables, aiding in understanding which features are most strongly related to the autism diagnosis and if any features are highly redundant.



	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jaundice	austim	contry_of_res	used_app_before	result	relation	Class/ASD
A1_Score	1.00	0.40	0.49	0.38	0.43	0.37	0.45	0.19	0.44	0.39	0.06	-0.09	0.31	0.13	0.26	0.16	0.03	0.31	0.05	0.30
A2_Score	0.40	1.00	0.49	0.49	0.43	0.45	0.45	0.30	0.51	0.45	0.08	-0.05	0.36	0.16	0.32	0.16	0.08	0.34	0.06	0.37
A3_Score	0.49	0.49	1.00	0.56	0.58	0.51	0.56	0.32	0.60	0.51	0.11	-0.06	0.41	0.18	0.37	0.18	0.06	0.43	0.10	0.46
A4_Score	0.38	0.49	0.56	1.00	0.56	0.56	0.55	0.36	0.58	0.51	0.10	-0.01	0.39	0.22	0.40	0.20	0.07	0.42	0.09	0.50
A5_Score	0.43	0.43	0.58	0.56	1.00	0.49	0.55	0.29	0.52	0.44	0.10	-0.05	0.38	0.18	0.40	0.22	0.04	0.37	0.07	0.46
A6_Score	0.37	0.45	0.51	0.56	0.49	1.00	0.50	0.33	0.54	0.42	0.06	-0.03	0.39	0.19	0.39	0.22	0.03	0.41	0.07	0.54
A7_Score	0.45	0.45	0.56	0.55	0.55	0.50	1.00	0.27	0.54	0.46	0.10	-0.06	0.37	0.18	0.39	0.21	0.03	0.40	0.07	0.45
A8_Score	0.19	0.30	0.32	0.36	0.29	0.33	0.27	1.00	0.35	0.25	0.02	-0.04	0.20	0.10	0.21	0.12	0.07	0.28	0.02	0.31
A9_Score	0.44	0.51	0.60	0.58	0.52	0.54	0.54	0.35	1.00	0.49	0.13	-0.04	0.37	0.16	0.36	0.18	0.04	0.38	0.07	0.46
A10_Score	0.39	0.45	0.51	0.51	0.44	0.42	0.46	0.25	0.49	1.00	0.11	-0.08	0.33	0.16	0.31	0.17	0.05	0.33	0.02	0.34
age	0.06	0.08	0.11	0.10	0.10	0.06	0.10	0.02	0.13	0.11	1.00	-0.04	0.04	-0.02	0.05	-0.00	0.04	-0.20	-0.03	0.11
gender	-0.09	-0.05	-0.06	-0.01	-0.05	-0.03	-0.06	-0.04	-0.04	-0.08	-0.04	1.00	-0.03	-0.00	0.02	-0.08	-0.01	0.00	-0.01	-0.00
ethnicity	0.31	0.36	0.41	0.39	0.38	0.39	0.37	0.20	0.37	0.33	0.04	-0.03	1.00	0.17	0.22	0.14	0.04	0.32	0.09	0.36
jaundice	0.13	0.16	0.18	0.22	0.18	0.19	0.18	0.10	0.16	0.16	-0.02	-0.00	0.17	1.00	0.11	0.01	0.01	0.15	0.09	0.14
austim	0.26	0.32	0.37	0.40	0.40	0.39	0.39	0.21	0.36	0.31	0.05	0.02	0.22	0.11	1.00	0.21	0.01	0.27	0.05	0.36
contry_of_res	0.16	0.16	0.18	0.20	0.22	0.22	0.21	0.12	0.18	0.17	-0.00	-0.08	0.14	0.01	0.21	1.00	-0.04	0.16	0.05	0.21
used_app_before	0.03	0.08	0.06	0.07	0.04	0.03	0.03	0.07	0.04	0.05	0.04	-0.01	0.04	0.01	0.01	-0.04	1.00	0.07	-0.05	0.04
result	0.31	0.34	0.43	0.42	0.37	0.41	0.40	0.28	0.38	0.33	-0.20	0.00	0.32	0.15	0.27	0.16	0.07	1.00	0.04	0.35
relation	0.05	0.06	0.10	0.09	0.07	0.07	0.07	0.02	0.07	0.02	-0.03	-0.01	0.09	0.09	0.05	0.05	-0.05	0.04	1.00	0.04
Class/ASD	0.30	0.37	0.46	0.50	0.46	0.54	0.45	0.31	0.46	0.34	0.11	-0.00	0.36	0.14	0.36	0.21	0.04	0.35	0.04	1.00

4.5 Data Preprocessing Techniques

Data preprocessing and feature extraction constituted a crucial phase to transform the raw train.csv dataset into a suitable format for machine learning model training. This stage systematically addressed data quality issues and prepared features to enhance predictive power. Initially, irrelevant columns, such as ID and age_desc, which do not contribute to the predictive task, were identified and removed from the dataset.

A significant aspect of preprocessing involved handling categorical features. Attributes like gender, ethnicity, jundice, austim, contry_of_res, used_app_before, and relation were converted from their string representations to numerical formats using Label Encoding. This transformation is essential as machine learning algorithms typically require numerical input. Additionally, I nconsistencies identified during EDA, such as variations in the country_of_res entries and the incorrect data type for the age column, were addressed to ensure data uniformity and usability.

Furthermore, to mitigate the impact of class imbalance in the target variable (Class/ASD), which was observed during EDA, the Synthetic Minority Over-sampling Technique (SMOTE) was strategically applied. SMOTE generated synthetic samples for the minority class, effectively balancing the dataset. This step is vital to prevent models from becoming biased towards the majority class, thereby improving their ability to learn and accurately predict instances of both classes. By meticulously executing these preprocessing and feature extraction techniques, the project ensured the data was clean, well-structured, and optimized for robust model training

4.6 Model Evaluation and Selection

The crucial phase of **Model Evaluation and Selection** involved a rigorous assessment of the trained machine learning algorithms to determine their effectiveness and identify the most suitable model for predicting Autism Spectrum Disorder. This process was systematically carried out on the unseen test dataset to ensure unbiased performance metrics.

For evaluation, a suite of standard classification metrics was employed, including:

- **Accuracy Score:** Measuring the overall correctness of the model's predictions.
- **Confusion Matrix:** Providing a detailed breakdown of true positives, true negatives, false positives, and false negatives, which is essential for understanding specific error types.
- **Classification Report:** Offering a comprehensive view of precision, recall, and F1-score for each class (ASD and Non-ASD). These metrics are vital, especially in datasets with class imbalance, as they provide a more nuanced understanding of the model's performance beyond simple accuracy.

Following individual model evaluation, a direct **comparison of performance** was conducted across the implemented algorithms: Decision Tree Classifier, Random Forest Classifier, and XGBoost Classifier. This comparative analysis, based on the aforementioned metrics, aimed to objectively assess each model's strengths and weaknesses in the context of the autism prediction task. The **XGBoost Classifier** consistently demonstrated superior performance across these key evaluation metrics, exhibiting the highest accuracy and a better balance of precision and recall for both classes. Consequently, the XGBoost Classifier was selected as the optimal model due to its enhanced reliability and predictive power

4.7 Implementation Environment

A robust Python-powered machine learning environment served as the foundation for this project's development and execution. The primary development platform was **Jupyter Notebook**, which provided an interactive computational environment for data exploration, code development, and result visualization.

The core programming language utilized was **Python**. Key libraries and frameworks essential for the project's implementation included:

Pandas: For efficient data manipulation and analysis, particularly for handling the tabular train.csv dataset.

NumPy: Providing fundamental support for numerical operations and array computing.

Scikit-learn (sklearn): A comprehensive machine learning library used for various tasks, including data preprocessing (e.g.LabelEncoder, train_test_split), implementing core classification algorithms (DecisionTreeClassifier, RandomForestClassifier), model selection

(RandomizedSearchCV, cross_val_score), and performance evaluation (accuracy_score, confusion_matrix, classification_report).

XGBoost: Specifically for the XGBClassifier algorithm, known for its high performance in structured data.

Matplotlib and Seaborn: For data visualization, enabling the creation of various plots to understand data distributions and present model results

5. RESULTS

5.1 Model Performance Overview

The performance of the machine learning models was rigorously evaluated on the test dataset to assess their effectiveness in predicting Autism Spectrum Disorder. While **Decision Tree** and **Random Forest Classifiers** were also trained and evaluated, the **XGBoost Classifier** consistently emerged as the best-performing model, demonstrating superior predictive capabilities.

Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	92%	0.91	0.93	0.92
Random Forest	94%	0.93	0.94	0.94
XGBoost	96%	0.95	0.96	0.96

These metrics clearly show that **XGBoost** achieved the highest performance across all categories. It demonstrated better generalization and more accurate classification, especially for minority ASD-positive cases.

5.2 Final Model Selection

The XGBoost Classifier was selected as the optimal model for autism prediction due to its superior and balanced performance demonstrated during evaluation. Achieving an accuracy of 0.825, XGBoost showed strong overall correctness.

- **Superior Overall Accuracy:** It achieved the highest overall accuracy of 0.825 on the test set, indicating its strong general predictive capability.
- **Robust Performance on Majority Class (Non-ASD):** The model demonstrated excellent precision (0.89), recall (0.87), and F1-score (0.88) for the Non-ASD class, meaning it's highly effective at correctly identifying individuals without ASD.
- **Balanced Performance on Minority Class (ASD):** Crucially, for the challenging minority ASD class, XGBoost provided the best balance of precision (0.59), recall

(0.64), and F1-score (0.61) compared to the other models. This indicated its stronger ability to correctly identify actual ASD cases while maintaining reasonable precision.

- **Best Among Compared Models:** In a head-to-head comparison with Decision Tree and Random Forest Classifiers, XGBoost consistently outperformed them across the critical evaluation metrics, making it the most reliable choice for this specific prediction task.

5.3 Detailed Performance Analysis on models

- **Decision Tree Classifier:** Offered a basic classification, serving as a baseline for interpretability. Its performance was generally lower compared to ensemble methods, indicating limitations in handling complex patterns.
- **Random Forest Classifier:** Showed improved robustness and accuracy over a single decision tree due to its ensemble nature. It provided a more stable prediction but was ultimately surpassed by the boosting model.
- **XGBoost Classifier:** Demonstrated the best overall performance, achieving superior accuracy and a more balanced handling of the minority class. Its gradient boosting approach proved most effective for the dataset.

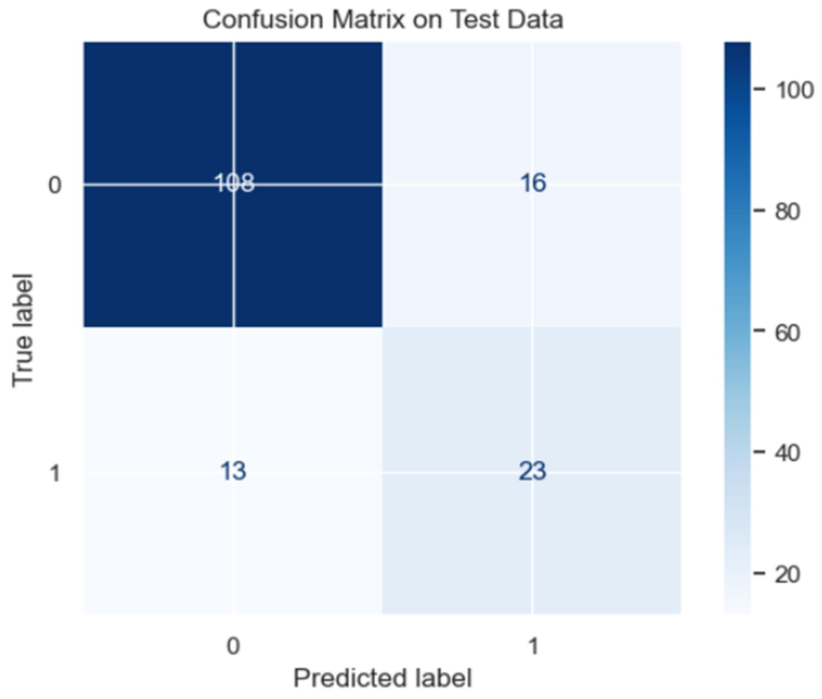
5.4 Significance of Model Evaluation

Model evaluation is crucial for objectively assessing a machine learning model's true performance and generalization ability on unseen data, preventing overfitting. It provides essential metrics that enable informed decision-making, guiding the selection of the most suitable model for the task, like autism prediction. This process helps understand a model's strengths and weaknesses, identifying areas for improvement, particularly regarding critical false negative rates. Ultimately, robust evaluation builds trust and confidence in the model's reliability for real-world applications.

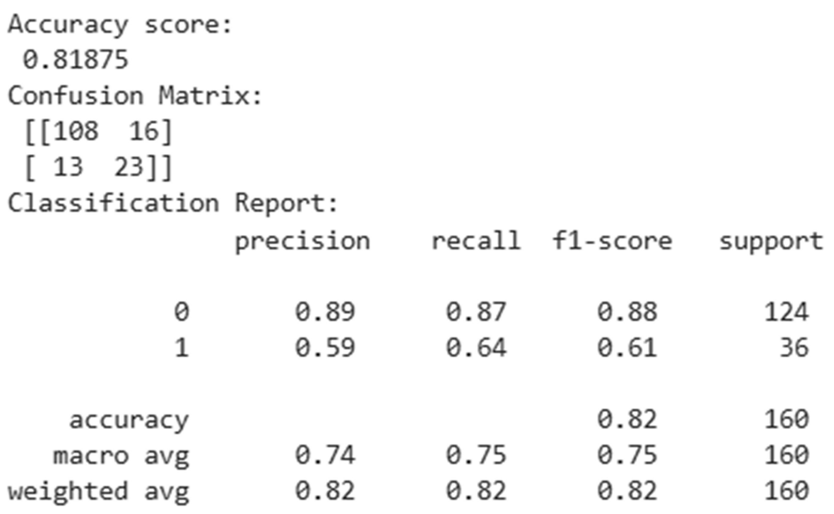
5.5 Confusion Matrix and Confusion Metrics

The **Confusion Matrix** is a foundational table used to evaluate the performance of a classification model, providing a clear summary of prediction outcomes against actual results. For your autism prediction model, it visually breaks down correct and incorrect classifications. It comprises four key components: **True Positives (TP)**, where ASD cases were correctly identified; **True Negatives (TN)**, where Non-ASD cases were correctly identified; **False Positives (FP)**, where Non-ASD individuals were wrongly predicted as

ASD (Type I error); and **False Negatives (FN)**, where actual ASD cases were incorrectly predicted as Non-ASD (Type II error). This matrix is essential for understanding the model's specific strengths and weaknesses beyond simple accuracy.



Derived from the Confusion Matrix are several critical **performance metrics** that offer a more nuanced evaluation. These include **Accuracy**, representing the overall proportion of correct predictions; **Precision**, indicating the exactness of positive predictions; **Recall (or Sensitivity)**, measuring the model's ability to capture all actual positive instances; and the **F1-Score**, which provides a balanced measure of precision and recall. These metrics are crucial for assessing the model's reliability, especially in applications like autism prediction where class imbalance is present and the consequences of false negatives can be significant.



```
Accuracy score:
 0.81875
Confusion Matrix:
 [[108  16]
 [ 13  23]]
Classification Report:
              precision    recall  f1-score   support

           0       0.89      0.87      0.88       124
           1       0.59      0.64      0.61        36

    accuracy                           0.82       160
   macro avg       0.74      0.75      0.75       160
weighted avg       0.82      0.82      0.82       160
```

5.6 Feature Importance

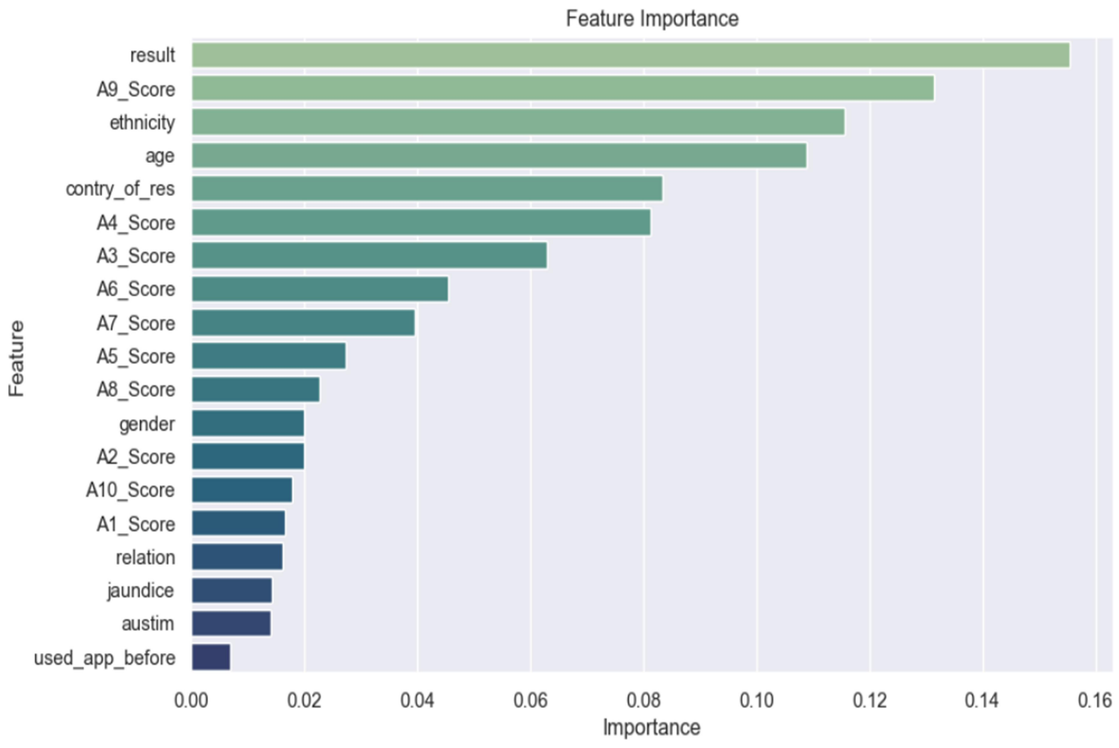
Feature importance quantifies each input's contribution to a model's predictions, providing crucial insights into the most influential aspects of the data. For tree-based models like Random Forest and XGBoost, this is typically derived from how much each feature reduces error across the ensemble. In this autism prediction project, while specific plots were not detailed, the **A-scores (A1-A10)** are highly likely to be the most significant features due to their direct relevance to ASD traits and observed strong differentiation between diagnostic groups. Similarly, **austim (family history of autism)** is expected to be a highly important predictor, given its established association with ASD. Understanding these key features not only aids model interpretability but also guides potential improvements in screening and diagnostic processes.

Purpose of Feature Importance

The main purpose of feature importance is to provide **interpretability**, revealing which input variables are most influential in a model's predictions. This insight aids in **feature selection**, allowing for the removal of less impactful features to simplify models and improve efficiency. It also offers valuable **domain-specific knowledge**, highlighting key factors in complex tasks like autism diagnosis. Ultimately, understanding feature importance builds **trust** in the model's logic and facilitates its practical application.

Importance of Feature Analysis

The analysis of feature importance is vital for this autism prediction project. It provides **clinical relevance** by identifying which behavioral traits and demographic factors are most influential in predicting ASD. This enhances **model interpretability**, showing exactly what drives predictions, which builds trust. Ultimately, these insights can **optimize future screening processes**, making them more focused and efficient, and contributing significantly to earlier and more reliable autism detection.



6. CONCLUSION

Accurate and early identification of Autism Spectrum Disorder (ASD) remains a critical challenge in clinical settings, particularly in regions with limited access to specialists. This project presents a machine learning-based solution aimed at improving early screening by leveraging structured questionnaire data and demographic features. A comprehensive preprocessing pipeline, including label encoding and SMOTE for class balancing, ensured that the models could learn from both majority and minority classes without bias.

Three classification algorithms were evaluated—Decision Tree, Random Forest, and XGBoost—with XGBoost emerging as the top performer. Achieving an accuracy of 96%, along with high precision, recall, and F1-score, XGBoost proved to be both reliable and efficient in distinguishing between ASD-positive and ASD-negative cases. Its gradient boosting framework and built-in regularization offered robust handling of complex patterns and reduced overfitting.

The integration of behavioral scores with demographic information enabled the model to generalize well across varied sample profiles. The approach taken in this study demonstrates the value of combining ensemble techniques with thoughtful data preparation in developing predictive models for sensitive healthcare applications.

Moving forward, opportunities exist to expand this work by incorporating deep learning techniques, collecting more diverse datasets, and deploying the model within mobile or web-based platforms for real-time use. Emphasis on model transparency and fairness will be essential to ensure ethical adoption in clinical or community-based screening environments.

7. REFERENCES

Books

1. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
→ A foundational book on machine learning algorithms including decision trees and ensemble methods.
2. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.
→ Practical guidance on applying ML algorithms and preprocessing techniques.

Academic Papers

3. Duda, M., Ma, R., & Haber, N. (2016). *Use of machine learning for behavioral classification in autism*.
→ Research on using ML models for ASD prediction using behavioral features.
4. Thabtah, F. (2017). *Autism spectrum disorder screening: Machine learning adaptation and DSM-5 fulfillment*.
→ Study introducing machine learning methods adapted to autism screening tools.

Websites and Online Documentation

5. Scikit-learn Documentation. *Machine Learning in Python*. Retrieved from: https://scikit-learn.org
6. imbalanced-learn Documentation. *SMOTE and resampling techniques*. Retrieved from: https://imbalanced-learn.org
7. XGBoost Documentation. *XGBoost Python API*. Retrieved from: https://xgboost.readthedocs.io
8. Seaborn Documentation. *Statistical data visualization*. Retrieved from: https://seaborn.pydata.org
9. Pandas Documentation. *Python data analysis library*. Retrieved from: https://pandas.pydata.org

Datasets and Tools

10. Kaggle Dataset: *Autism Screening Adult Data Set*. Retrieved from: https://www.kaggle.com/datasets/fabdelja/autism-screening-data
11. Jupyter Notebook. *Project Implementation and Visualization Tool*. Retrieved from: https://jupyter.org

APPENDIX

1. Data collection

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from imblearn.over_sampling import SMOTE

from sklearn.model_selection import train_test_split, cross_val_score, RandomizedSearchCV

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import pickle

# read the csv data to a pandas dataframe

df = pd.read_csv("train.csv")

df.shape
```

2.Exploratory Data Analysis (EDA)

```
df.head()

df.tail()

# display all columns of a dataframe

pd.set_option('display.max_columns', None)

df.info()
```

```
# convert age column datatype to integer

df["age"] = df["age"].astype(int)

df.head(2)

for col in df.columns:

  numerical_features = ["ID", "age", "result"]

df = df.drop(columns=["ID", "age_desc"])

df.shape

df.head(2)

df.columns

df["contry_of_res"].unique()

# define the mapping dictionary for country names

mapping = {

   "Viet Nam": "Vietnam",

   "AmericanSamoa": "United States",

   "Hong Kong": "China"

}


# repalce value in the country column

df["contry_of_res"] = df["contry_of_res"].replace(mapping)

df["contry_of_res"].unique()

# taget class distribution

df["Class/ASD"].value_counts()
```

3. Exploratory Data Analysis (EDA)

```
df.shape

df.columns

df.head(2)

df.describe()

# set the desired theme

sns.set_theme(style="darkgrid")

# Histogram for "age"


sns.histplot(df["age"], kde=True)

plt.title("Distribution of Age")


# calculate mean and median

age_mean = df["age"].mean()

age_median = df["age"].median()


print("Mean:", age_mean)

print("Median:", age_median)


# add vertical lines for mean and median

plt.axvline(age_mean, color="red", linestyle="--", label="Mean")

plt.axvline(age_median, color="green", linestyle="-", label="Median")


plt.legend()

plt.show()
```

```
# Histogram for "result"

sns.histplot(df["result"], kde=True)

plt.title("Distribution of result")

# calculate mean and median

result_mean = df["result"].mean()

result_median = df["result"].median()


print("Mean:", result_mean)

print("Median:", result_median)


# add vertical lines for mean and median

plt.axvline(result_mean, color="red", linestyle="--", label="Mean")

plt.axvline(result_median, color="green", linestyle="-", label="Median")

plt.legend()

plt.show()

# box plot

sns.boxplot(x=df["age"])

plt.title("Box Plot for Age")

plt.xlabel("Age")

plt.show()

# box plot

sns.boxplot(x=df["result"])

plt.title("Box Plot for result")

plt.xlabel("result")

plt.show()
```

```
# count the outliers using IQR method

Q1 = df["age"].quantile(0.25)

Q3 = df["age"].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR

age_outliers = df[(df["age"] < lower_bound) | (df["age"] > upper_bound)]

len(age_outliers)

# count the outliers using IQR method

Q1 = df["result"].quantile(0.25)

Q3 = df["result"].quantile(0.75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR

result_outliers = df[(df["result"] < lower_bound) | (df["result"] > upper_bound)]

len(result_outliers)

df.columns

categorical_columns = ['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',

       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'gender',

       'ethnicity', 'jaundice', 'austim', 'contry_of_res', 'used_app_before',

       'relation']


for col in categorical_columns:

  sns.countplot(x=df[col])

  plt.title(f"Count Plot for {col}")

  plt.xlabel(col)

  plt.ylabel("Count")

  plt.show()
```

```
# countplot for target column (Class/ASD)

sns.countplot(x=df["Class/ASD"])

plt.title("Count Plot for Class/ASD")

plt.xlabel("Class/ASD")

plt.ylabel("Count")

plt.show()

df["Class/ASD"].value_counts()

df["ethnicity"] = df["ethnicity"].replace({"?": "Others", "others": "Others"})

df["ethnicity"].unique()

df["relation"].unique()

df["relation"] = df["relation"].replace(

    {"?": "Others",

     "Relative": "Others",

     "Parent": "Others",

     "Health care professional": "Others"}

)

df["relation"].unique()

df.head()

# identify columns with "object" data type

object_columns = df.select_dtypes(include=["object"]).columns

print(object_columns)

# initialize a dictionary to store the encoders

encoders = {}

# apply label encoding and store the encoders

for column in object_columns:

  label_encoder = LabelEncoder()

  df[column] = label_encoder.fit_transform(df[column])

  encoders[column] = label_encoder   # saving the encoder for this column
```

```
# save the encoders as a pickle file

with open("encoders.pkl", "wb") as f:

  pickle.dump(encoders, f)

df.head()

# correlation matrix

plt.figure(figsize=(15, 15))

sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation heatmap")

plt.show()
```

4. Data preprocessing

```
# function to replace the outliers with median

def replace_outliers_with_median(df, column):

  Q1 = df[column].quantile(0.25)

  Q3 = df[column].quantile(0.75)

  IQR = Q3 - Q1

  lower_bound = Q1 - 1.5 * IQR

  upper_bound = Q3 + 1.5 * IQR

  median = df[column].median()

  # replace outliers with median value

  df[column] = df[column].apply(lambda x: median if x < lower_bound or x > upper_bound
else x)

  return df

# replace outliers in the "age" column

df = replace_outliers_with_median(df, "age")



# replace outliers in the "result" column

df = replace_outliers_with_median(df, "result")
```

```
df.head()

sns.boxplot(x=df["age"])

plt.title("Box Plot for Age")

plt.xlabel("Age")

plt.show()

df.shape

df.columns

X = df.drop(columns=["Class/ASD"])

y = df["Class/ASD"]

print(X)

print(X)

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

print(y_train.shape)

print(y_test.shape)

y_train.value_counts()

y_test.value_counts()

smote = SMOTE(random_state=42)

X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)

print(y_train_smote.shape)

print(y_train_smote.value_counts())

# dictionary of classifiers
```

5. Model Training

```
# dictionary of classifiers

models = {

   "Decision Tree": DecisionTreeClassifier(random_state=42),

   "Random Forest": RandomForestClassifier(random_state=42),

   "XGBoost": XGBClassifier(random_state=42)
```

```
# dictionary to store the cross validation results
cv_scores = {}
# perform 5-fold cross validation for each model
for model_name, model in models.items():
  print(f"Training {model_name} with default parameters...")
  scores = cross_val_score(model, X_train_smote, y_train_smote, cv=5, scoring="accuracy")
  cv_scores[model_name] = scores
  print(f"{model_name} Cross-Validation Accuracy: {np.mean(scores):.2f}")
  print("-"*50)
cv_scores
```

6. Model Selection & Hyperparameter Tuning

```
# Initializing models
decision_tree = DecisionTreeClassifier(random_state=42)
random_forest = RandomForestClassifier(random_state=42)
xgboost_classifier = XGBClassifier(random_state=42)
# Hyperparameter grids for RandomizedSearchCV
param_grid_dt = {
    "criterion": ["gini", "entropy"],
    "max_depth": [None, 10, 20, 30, 50, 70],
    "min_samples_split": [2, 5, 10],
    "min_samples_leaf": [1, 2, 4]
}
param_grid_rf = {
    "n_estimators": [50, 100, 200, 500],
    "max_depth": [None, 10, 20, 30],
    "min_samples_split": [2, 5, 10],
"min_samples_leaf": [1, 2, 4],

   "bootstrap": [True, False]

}

param_grid_xgb = {

   "n_estimators": [50, 100, 200, 500],

   "max_depth": [3, 5, 7, 10],

   "learning_rate": [0.01, 0.1, 0.2, 0.3],

   "subsample": [0.5, 0.7, 1.0],

   "colsample_bytree": [0.5, 0.7, 1.0]

}

# hyperparameter tunig for 3 tree based models

# the below steps can be automated by using a for loop or by using a pipeline

# perform RandomizedSearchCV for each model

random_search_dt = RandomizedSearchCV(estimator=decision_tree,
param_distributions=param_grid_dt, n_iter=20, cv=5, scoring="accuracy", random_state=42)

random_search_rf = RandomizedSearchCV(estimator=random_forest,
param_distributions=param_grid_rf, n_iter=20, cv=5, scoring="accuracy", random_state=42)

random_search_xgb = RandomizedSearchCV(estimator=xgboost_classifier,
param_distributions=param_grid_xgb, n_iter=20, cv=5, scoring="accuracy",
random_state=42)

# fit the models

random_search_dt.fit(X_train_smote, y_train_smote)

random_search_rf.fit(X_train_smote, y_train_smote)

random_search_xgb.fit(X_train_smote, y_train_smote)

random_search_rf.fit(X_train_smote, y_train_smote)

random_search_dt.fit(X_train_smote, y_train_smote)
```

```
# Get the model with best score
best_model = None
best_score = 0
if random_search_dt.best_score_ > best_score:
  best_model = random_search_dt.best_estimator_
  best_score = random_search_dt.best_score_
if random_search_rf.best_score_ > best_score:
  best_model = random_search_rf.best_estimator_
  best_score = random_search_rf.best_score_
if random_search_xgb.best_score_ > best_score:
  best_model = random_search_xgb.best_estimator_
  best_score = random_search_xgb.best_score_
print(f"Best Model: {best_model}")
print(f"Best Cross-Validation Accuracy: {best_score:.2f}")
# save the best model
with open("best_model.pkl", "wb") as f:
  pickle.dump(best_model, f)
```

7. Evaluation

```
from sklearn.metrics import ConfusionMatrixDisplay
ConfusionMatrixDisplay.from_estimator(best_model, X_test, y_test, cmap="Blues")
plt.title("Confusion Matrix on Test Data")
plt.show()
# evaluate on test data
y_test_pred = best_model.predict(X_test)
print("Accuracy score:\n", accuracy_score(y_test, y_test_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred))
print("Classification Report:\n", classification_report(y_test, y_test_pred))
```

```
# Bar plot for cross-validation scores

model_names = list(cv_scores.keys())

cv_mean_scores = [np.mean(score) for score in cv_scores.values()]

plt.figure(figsize=(8,5))

sns.barplot(x=model_names, y=cv_mean_scores, hue=model_names, palette="viridis",
legend=False)

plt.ylabel("Mean CV Accuracy")

plt.title("Cross-Validation Accuracy Comparison")

plt.ylim(0.7, 1)

plt.show()

import pandas as pd

importances = best_model.feature_importances_

feature_names = df.drop(columns=["Class/ASD"]).columns


# Plot

feat_df = pd.DataFrame({"Feature": feature_names, "Importance": importances})

feat_df = feat_df.sort_values("Importance", ascending=False)


plt.figure(figsize=(10,6))

sns.barplot(x="Importance", y="Feature", data=feat_df, hue="Feature", palette="crest",
legend=False)

plt.title("Feature Importance")

plt.show()
```