University of Hertfordshire UH

School of Physics, Engineering and Computer Science

MSc Data Science Project

7PAM2002-0206-2023

Department of Physics, Astronomy and Mathematics

Data Science FINAL PROJECT REPORT

Project Title:

Evaluating Machine Learning Algorithms for Autism Spectrum Disorder

Student Name and SRN:

Anjana Joshy

22022447

Supervisor: Dr Gulay Gurkan

Date Submitted: 01/05/2024

Word Count: 9930

DECLARATION STATEMENT

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science at the University of Hertfordshire.

I have read the guidance to students on academic integrity, misconduct and plagiarism information at **Assessment Offences and Academic Misconduct** and understand the University process of dealing with suspected cases of academic misconduct and the possible penalties, which could include failing the project module or course.

I certify that the work submitted is my own and that any material derived or quoted from published or unpublished work of other persons has been duly acknowledged. (Ref. UPR AS/C/6.1, section 7 and UPR AS/C/5, section 3.6). I have not used chatGPT, or any other generative AI tool, to write the report or code (other than where I have declared or referenced it).

I did not use human participants or undertake a survey in my MSc Project.

I hereby give permission for the report to be made available on module websites provided the source is acknowledged.

Student Name printed: Anjana Joshy

Student Name signature: 

Student SRN number: 22022447

UNIVERSITY OF HERTFORDSHIRE

SCHOOL OF PHYSICS, ENGINEERING AND COMPUTER SCIENCE

ACKNOWLEDGMENT

I want to express my heartfelt gratitude to God for the strength and guidance that were instrumental in successfully completing this project. His presence has been a constant throughout this journey.

I sincerely thank my supervisor, **Dr. Gulay Gurkan**, for her invaluable guidance and mentoring. Her extensive knowledge has been crucial throughout my academic career and this project.

I also extend my appreciation to the University of Hertfordshire for its enriching academic environment and resources, which significantly contributed to my research and personal development.

Finally, I express my gratitude to my family and friends for their unwavering support and encouragement. Their faith in my abilities and steadfast encouragement have been my pillars of strength, helping me navigate this challenging yet rewarding journey.

ABSTRACT

This project investigates the use of machine learning (ML) models to improve the diagnosis of autism spectrum disorder (ASD), with the goal of determining the most effective algorithms for detecting ASD-related patterns in young children. The study uses a Kaggle dataset with 1,985 records and 28 features that cover a variety of attributes such as demographic information, Autism Spectrum Quotient scores, and behavioural assessments. The methodology begins with data preprocessing, feature scaling, and dimensionality reduction using Principal Component Analysis (PCA), then Support Vector Machine (SVM), Random Forest (RF), and Logistic Regression (LR) models. The findings show that RF model emerged as the top performer for ASD diagnosis, showcasing high accuracy and precision. RF's superior performance was confirmed through confusion matrices and precision-recall curves, solidifying its effectiveness. Following closely behind, the SVM demonstrated balanced precision and recall, although slightly lower than RF. Conversely, LR showed lower accuracy, indicating a need for further optimization. The analysis reveals the strength of RF and SVM in ASD diagnosis, supporting the project's objective of integrating ML into ASD diagnostic frameworks. The study concludes that these models offer a promising approach for early ASD detection, with future work recommended to test the models' generalizability on other ASD-related datasets.

CONTENT

1. INTRODUCTION

ASD is a complex neurodevelopmental condition characterised by persistent difficulties in verbal and non-verbal communication, social interaction, and repetitive behaviours. The severity and impact of ASD vary widely from one individual to another, adding layers of complexity to its study and management. Ongoing research suggests that ASD's causes are multifactorial, involving a combination of genetic and environmental factors that continue to puzzle researchers (Simeoli et al., 2021). Such complexities highlight the importance of early detection and comprehensive interventions for improving affected people's quality of life. Several factors, including developmental stages, IQ, and environmental conditions, such as access to specialised educational and therapeutic services, are important in the effective management of ASD (Hirota and King, 2023). Furthermore, the frequent occurrence of ASD alongside other medical and psychiatric conditions, including epilepsy, sleep disturbances, anxiety, and depression, highlights the importance of integrated care approaches.

The timing of ASD diagnosis varies greatly, with diagnoses made anywhere from 38 to 120 months old. The traditional diagnostic approach, which relies on observational analysis and structured interviews, faces challenges in terms of objectivity and timeliness (Simeoli et al., 2021). Early detection is critical because it allows for early intervention, which has a significant impact on the growth and development of children with ASD. As a result, addressing and eliminating barriers to early diagnosis is critical to ensuring that all affected children receive timely and appropriate treatment.

The application of ML into ASD research opens up promising possibilities for diagnosis and understanding. It represents a shift towards objective, scalable, and potentially more accurate diagnostic processes by analysing large datasets to uncover patterns that humans would struggle to detect. This technological advancement emphasises the increasing importance of computational methods in brain disorder research (Simeoli et al., 2021).

1.1 Aim:

This project aims to identify the most effective ML models for detecting and analyzing patterns associated with ASD in young children and to investigate the differences in their outputs.

1.2 Objectives:

- To evaluate the efficacy of various ML models in identifying distinctive patterns related to ASD.
- To compare the outputs of different models to understand their strengths and weaknesses in diagnosing ASD.
- To improve the accuracy and timeliness of ASD diagnosis through the integration of effective ML models into existing diagnostic frameworks.

2. BACKGROUND

The widespread use of ML in healthcare, particularly in the early detection of autism, represents a significant shift towards employing computational methods to improve diagnostic accuracy and efficiency. Our literature review focused on pivotal studies that used various ML techniques to classify ASD across age groups, highlighting their methodologies, findings, and limitations. The papers required in this report take a comprehensive approach to ASD diagnosis that employs a variety of ML models such as LR, SVM, RF, AdaBoost, and others. This selection criteria provided us with insights into the current landscape of ML applications in ASD diagnosis, demonstrating the potential for tailored ML methods to effectively address the diagnostic complexities of ASD.

Chowdhury and Iraj, 2020 introduced a concise ML based approach aimed at enhancing the detection of ASD across different ages, utilizing a combination of feature scaling (FS) methods and ML techniques on datasets for Toddlers, Adolescents, Children, and Adults. The effectiveness of various ML strategies, including Ada Boost (AB), RF, and others, was evaluated with FS strategies like Quantile Transformer (QT) and Normalizer. Through preprocessing, scaling, and classifying the data, our analysis pinpointed AB and Linear Discriminant Analysis (LDA) as the leading classifiers. AB achieved outstanding accuracy rates of 99.25% and 97.95% for Toddlers and Children, respectively, whereas LDA provided the most accurate results for Adolescents and Adults, with accuracies of 97.12% and 99.03%. The study underlines the significance of choosing appropriate feature selection techniques to identify key ASD risk factors, aiding in clinical assessments. Despite its advancements in early ASD detection, the study's scope was limited by the amount of available data, suggesting a need for a more comprehensive dataset to refine the predictive model's applicability across all age stages. Future efforts will focus on expanding data collection to enhance the model's generalizability for ASD and similar neuro-developmental conditions. This work offers a new direction in early ASD detection, showcasing the potential of tailored ML methods to tackle ASD's diagnostic complexities effectively.

Akter et al (2021) investigated various ML algorithms, including Artificial Neural Networks (ANN), Decision Trees (DT), Gradient Boosting (GB), K Nearest Neighbour (KNN), LR , and RF, to determine their effectiveness in the early diagnosis of ASD across different age demographics. LR emerged as the best model, achieving exceptional accuracy rates in ASD screenings. Although LR led the study, RF also demonstrated notable accuracy, with reported rates around 95.9%, indicating its potential for ASD diagnosis. The research methodology included preprocessing techniques to improve dataset integrity, as well as feature transformation methods such as standardisation and normalisation to improve predictive model performance. Despite the study's notable accuracy, it highlights the inherent limitations of relying on behavioural indicators for ASD detection and the constrained dataset sizes used. These aspects highlight the need for future studies to broaden the scope of data collection and analysis to refine and validate the effectiveness of the presented models.

The work by Raj and Masood (2020) assesses multiple ML algorithms for their ability to accurately detect ASD in various age groups, employing datasets that span children, adolescents, and adults. The models tested were Naïve Bayes, SVM, LR, KNN, Neural Network, and the Convolutional Neural Network (CNN). The CNN model emerged as the most successful, with remarkable accuracy levels of 99.53% in adults, 98.30% in children, and 96.88% in adolescents, demonstrating its efficacy for ASD screening and advanced predictive capacity. SVM also achieved notable performance, with an accuracy rate of 98%, underscoring its potential for effective ASD diagnosis. The research methodology highlighted the importance of data preprocessing and feature

transformation in improving model performance. The study ensured the reliability of the ML techniques used by dealing with missing values and applying methods such as standardization and normalization. This approach not only demonstrated CNNs' applicability in neuro-disorder diagnostics, but it also established the way for future medical research. By validating these findings across larger datasets, the study opens up new possibilities for incorporating advanced ML techniques into ASD diagnosis, thereby contributing to early intervention and management strategies for the disorder.

Mahedy Hasan et al. (2023) describes an effective ML framework for improving the early detection of ASD in various age groups, including toddlers, children, adolescents, and adults. The framework successfully classifies ASD using four distinct Feature Scaling techniques i.e., QT, PT, Normalizer and MAS and eight different ML models. Notably, AB demonstrated exceptional accuracy in diagnosing ASD in Toddlers and Children, with success rates exceeding 97%, whereas Linear Discriminant Analysis (LDA) proved most efficient in Adolescents and Adults, with accuracy reaching 99.03%. Furthermore, the study conducted a thorough evaluation of feature significance using four Feature Selection Techniques: Info Gain Attribute Evaluator (IGAE), Gain Ratio Attribute Evaluator (GRAE), Relief F Attribute Evaluator (RFAE), and Correlation Attribute Evaluator. The goal of this study was to identify the critical factors influencing ASD prediction across age groups, providing valuable insights for medical professionals in ASD screening. This study's holistic approach, which includes multiple FS methods, ML models, and detailed feature significance assessment, suggests a promising strategy for early ASD detection, potentially facilitating the development of specialized intervention and care plans.

In the comprehensive study "A Machine Learning Way to Classify ASD" by R et al.(2021), a variety of ML models were explored to classify ASD using datasets from the UCI ML repository. This study is significant because it navigates the complexities of ASD diagnosis across various age groups, using models such as SVM, KNN, RF, NB, SGD, AB, and CN2 Rule Induction to determine the most effective classification approach. The study's unique methodology included attribute extraction using Principal Component Analysis (PCA) and model comparison based on accuracy, precision, recall, and F1 score. Notably, Stochastic Gradient Descent (SGD) performed exceptionally well in diagnosing ASD in adults and children, with accuracies of 99.7% and 99.6%, respectively, while AB performed well in toddler datasets with 99.8% accuracy. These findings are significant for the project because they provide insights into selecting appropriate models for ASD diagnosis based on specific demographic groups, laying the groundwork for improved diagnostic precision. The study provides a useful comparative analysis of ML models for ASD classification, emphasizing the importance of tailored algorithm selection in improving diagnostic outcomes. This comparison method focuses on how accurately different algorithms perform, highlighting how ML can transform the diagnosis of ASD. It also points out the need for more research into making these models clearer and ensuring the data used is reliable, to make the most of ML in healthcare.

The study "Detection of ASD in Children Using Machine Learning Techniques" by Vakadkar et al.'s (2021) aimed to address the challenges in diagnosing ASD at an early stage using conventional methods by incorporating ML techniques to speed up and improve the diagnosis process. The study used ML models (SVM, RFC, NB, LR, and KNN) to analyse a dataset of toddler screening results using the Q-CHAT-10 method. The primary goal was to determine whether a child is susceptible to ASD during the early stages, thereby streamlining the diagnosis. Among the models tested, the LR model performed the best, with an accuracy of 97.15% on the used dataset. Based on the dataset generated by the Q-CHAT-10 screening method, this demonstrates the potential of

LR to accurately predict ASD in children at an early stage. The researchers thoroughly preprocessed the dataset, cleaning it and encoding categorical variables to ensure that the ML models were trained on relevant and structured information. The dataset was then divided into training and testing segments so that the effectiveness of each ML model could be properly assessed using various performance metrics such as accuracy and F1. This structured approach emphasised the importance of careful data handling and strategic use of ML algorithms in improving the diagnostic process for ASD in children, demonstrating the research's commitment to improving medical diagnostics through technological advancements.

Based on their findings, I'm particularly interested in SVM, RF, and LR because of their demonstrated efficacy in ASD classification, ability to manage high-dimensional data, and resistance to overfitting. These models provide a comprehensive approach to ASD diagnosis, combining linear and nonlinear techniques to effectively address the condition's complexities.

3. DATASET

The dataset, titled "ASD Children Traits," was sourced from Kaggle and provided by the Computer Science Department at the University of Arkansas. It is publicly shared under the CC0: Public Domain license, facilitating broad accessibility for research purposes. (www.kaggle.com, n.d.)

This dataset, containing 1,985 records and spanning 28 different fields, offers a detailed overview of assessments related to ASD, encompassing both categorical and numerical data types. It includes scores from the Autism Spectrum Quotient (ASQ) which range from A1 to A10, Social Responsiveness Scale, Qchat_10 score, and Childhood Autism Rating Scale. Demographic details such as age, gender, and ethnicity are also recorded, showcasing a wide ethnic variety with notable numbers of Asian, White European, and Middle Eastern individuals. Additionally, the dataset records other important aspects such as speech and learning disorders, depression, genetic conditions, and ASD family history. Most assessments were conducted by healthcare professionals or family members. Information on developmental and behavioral conditions, including global developmental delay, social or behavioral issues, anxiety disorders, and the presence of jaundice, is also included, making this dataset a rich resource for analyzing ASD characteristics.

This dataset was chosen for my research because it covers a wide range of attributes relevant to ASD screening, providing an ideal foundation for evaluating the predictive abilities of various ML models in diagnosing ASD. The dataset's diversity, particularly in demographic and ASD-related conditions, contributes significantly to the research. The diverse ethnic representation and inclusion of a wide range of ASD-related conditions highlight ASD's complexity as a neurodevelopmental disorder. This diversity is critical for creating ML models that are robust, inclusive, and capable of accurately detecting ASD across multiple populations and presentations.

My research examines the performance of ML models on this dataset to determine their potential suitability and efficacy in clinical settings for ASD screening. The dataset's rich variety not only allows for a better understanding of the spectrum of ASD characteristics, but it also helps to create models that can adapt to the slight differences in ASD manifestations. This is critical for conducting accurate screenings and ensuring that diagnoses result in prompt intervention and support measures, ultimately benefiting individuals with ASD and their families by catering to their unique needs and circumstances.

3.1 ETHICAL CONSIDERATION

The project maintains ethical standards by excluding personal details from the dataset and ensuring confidentiality throughout. Every step of the project adheres to ethical guidelines, promoting transparency. By prioritizing privacy and ethical conduct, the project upholds integrity in ASD diagnosis research, eliminating the need for specific institutional approval.

3.2 Exploratory Data Analysis (EDA):

EDA of the ASD Screening Data marks the beginning of our deep dive into understanding patterns and trends that may help improve how we screen for ASD. The journey begins with an overview from Table 1, which shows us how the dataset is structured. This table shows that our dataset contains 1,985 records spanning 28 different types of information, including numbers and categories. It also indicates where we are missing information, which will be useful as we proceed with our analysis.

#	Column	Non-Null Count	Dtype
0	CASE_NO_PATIENT'S	1985 non-null	int64
1	A1	1985 non-null	int64
2	A2	1985 non-null	int64
3	A3	1985 non-null	int64
4	A4	1985 non-null	int64
5	A5	1985 non-null	int64
6	A6	1985 non-null	int64
7	A7	1985 non-null	int64
8	A8	1985 non-null	int64
9	A9	1985 non-null	int64
10	A10_Autism_Spectrum_Quotient	1985 non-null	int64
11	Social_Responsiveness_Scale	1976 non-null	float64
12	Age_Years	1985 non-null	int64
13	Qchat_10_Score	1946 non-null	float64
14	Speech Delay/Language Disorder	1985 non-null	object
15	Learning disorder	1985 non-null	object
16	Genetic_Disorders	1985 non-null	object
17	Depression	1984 non-null	object
18	Global developmental delay/intellectual disability	1985 non-null	object
19	Social/Behavioural Issues	1971 non-null	object
20	Childhood Autism Rating Scale	1985 non-null	int64
21	Anxiety_disorder	1985 non-null	object
22	Sex	1985 non-null	object
23	Ethnicity	1985 non-null	object
24	Jaundice	1985 non-null	object
25	Family_mem_with_ASD	1985 non-null	object
26	Who_completed_the_test	1985 non-null	object
27	ASD_traits	1985 non-null	object

dtypes: float64(2), int64(13), object(13)
memory usage: 434.3+ KB

Table 1: 'Autism Overview' - Essential Information in Brief.

This early look at the data through Table 1 allows us to notice the dataset's variety, which covers a wide range of ASD-related conditions. It emphasises how ASD manifests itself in a variety of ways, as well as the importance of using sophisticated screening models. To improve our understanding, I used value counts to delve deeper into the specifics of each feature, providing a clearer picture of the dataset's composition. Furthermore, by sorting the age values, I discovered that the dataset covers a wide age range, from one to eighteen years. This analysis provides a comprehensive understanding of the demographic distribution, which is critical for understanding the manifestation of ASD across different age groups.

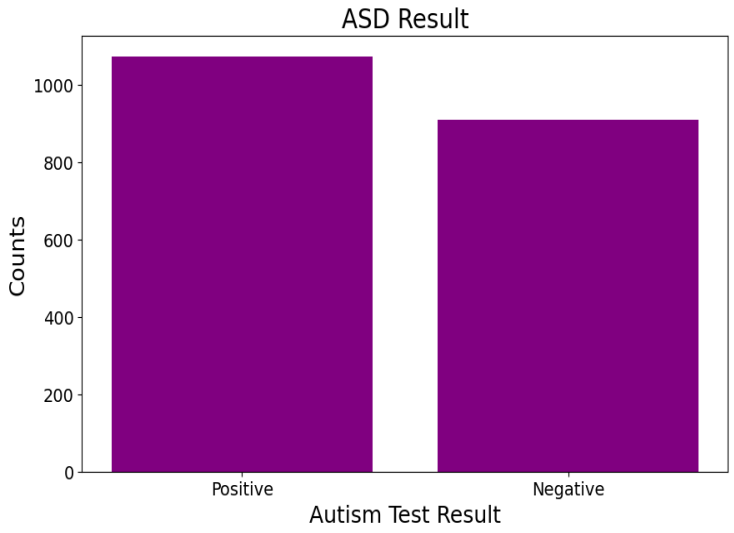

Figure 1: ASD Trait Distribution - Positive and Negative Cases Illustrated

Figure 1 shows a bar chart depicting the distribution of ASD traits, which provides important insight into the dataset's composition, with positive cases slightly exceeding 1000 and negative cases close to 900. This near-equilibrium state suggests that the dataset is relatively balanced, which is critical for developing machine learning models for ASD screening. A balanced dataset ensures that predictive models built on this data are less likely to exhibit bias toward either positive or negative cases, increasing their reliability and generalizability to new, previously unseen data. This balance enables researchers and clinicians to be confident that the tools they develop will perform consistently across diverse populations.

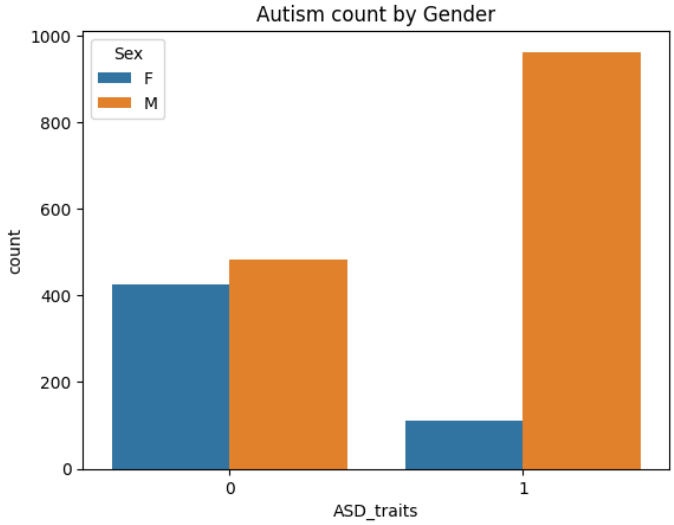

Figure 2: A Comparative Plot of Male and Female Cases

Transitioning to Figure 2, which depicts a clear difference in the occurrence of positive ASD traits between males and females, providing important insight into gender differences in ASD manifestation. With males showing a higher prevalence of positive ASD traits, this finding is consistent with previous research indicating that ASD is more commonly diagnosed in males than females. However, the close frequency of negative ASD traits across both genders, which ranges between 400 and 500, suggests that, while ASD is more common in men, the traits associated with ASD are distributed more evenly across genders than is commonly assumed. This underscores the need for gender-sensitive approaches in ASD screening and diagnosis, ensuring that models and methodologies are finely tuned to recognize ASD traits across genders effectively.

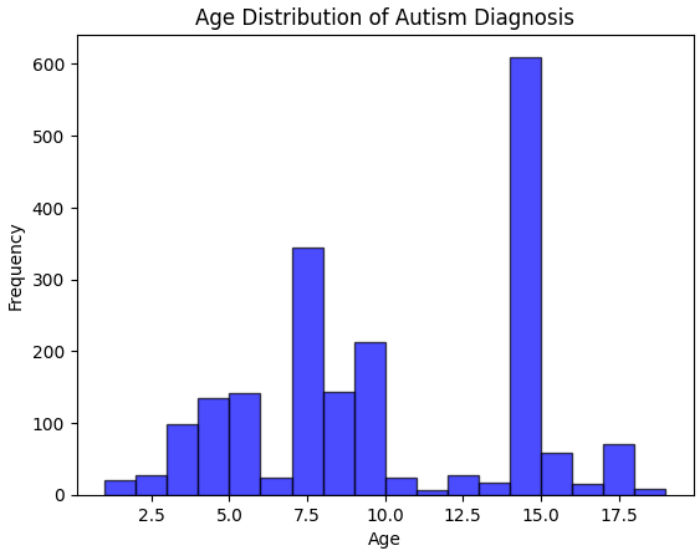

Figure 3: Distribution of ASD Cases Across Different Age Groups

The age distribution of autism diagnoses, explored through a histogram in Figure 3, delineates a pronounced peak at age 14, indicating that this age is critical for ASD detection. This trend, characterised by the highest diagnosis rates in the mid-teen years and closely followed by early childhood, emphasises the critical periods for early intervention efforts. Furthermore, the histogram demonstrates meaningful diagnosis rates in late childhood and preschool, emphasising the significance of these developmental stages in ASD detection. Furthermore, a discernible but lower peak in late adolescence indicates the possibility of later-life diagnoses, highlighting the importance of ongoing awareness and screening at all ages. This analysis not only highlights age 14 as a critical juncture for ASD diagnosis, but it also emphasises the importance of remaining vigilant and providing timely intervention across various developmental stages in order to effectively support individuals with ASD.

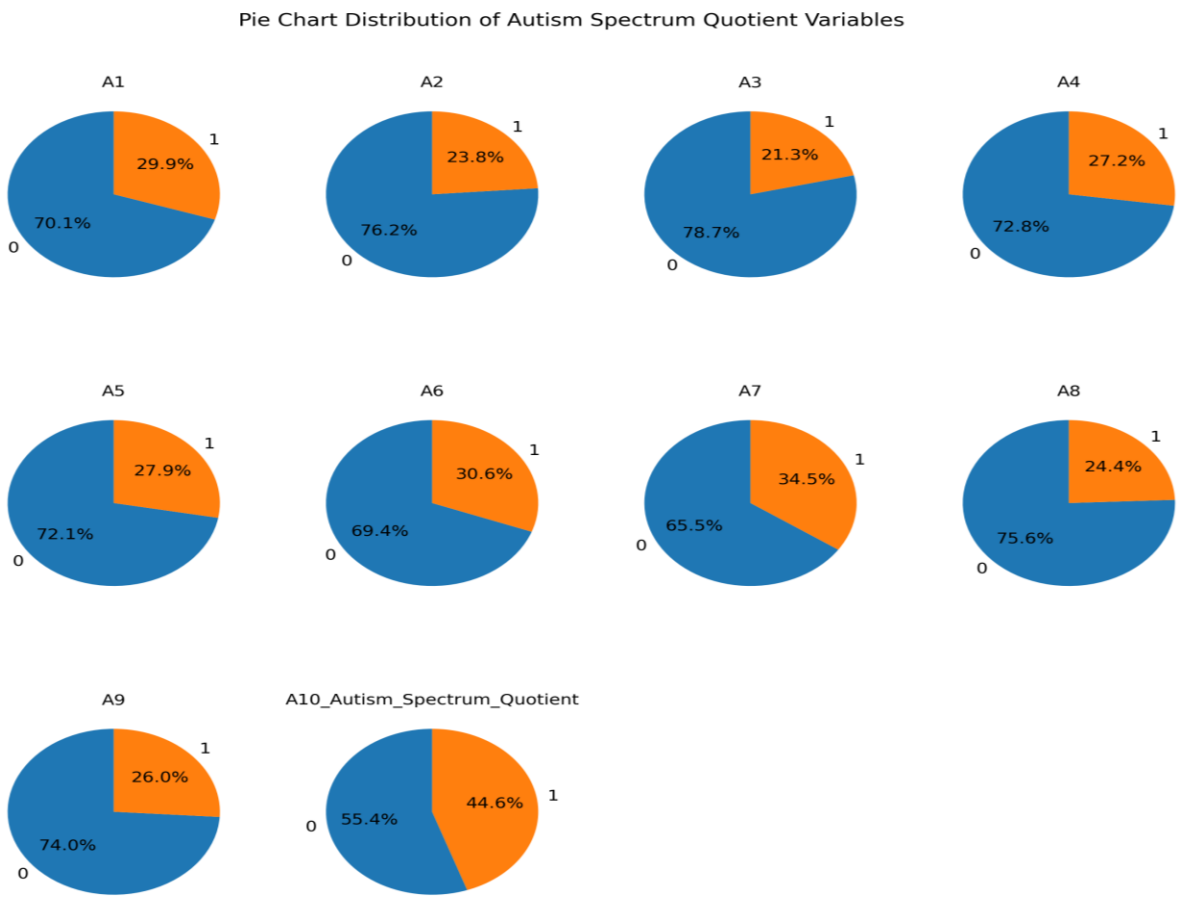

Figure 4: Proportional Distribution of Responses to Autism Spectrum Quotient Items

In figure 4, the pie chart collection offers a visual breakdown of responses to ASQ variables, A1 through A10. Each chart presents the proportion of binary responses—likely coded as 0s and 1s—across different questionnaire items. For instance, items A2, A3, and A8 display a higher percentage of 0 responses, suggesting a predominant tendency toward one type of answer within these variables. Conversely, A10 exhibits a near even split, indicating a more balanced distribution of responses. This visual data presentation aids in quickly discerning patterns across the AQ items, revealing which traits are more universally observed and which show greater variance, thereby providing a clearer understanding of the dataset and informing subsequent research directions in the context of autism.

After analyzing the 'ASD Children Traits' dataset, we found a wide variety of demographics and an even distribution of ASD traits. These key points are critical as they match our goal to find the best ML models for spotting ASD in children. The diverse demographics highlight our aim to test models on different groups, and the balanced ASD traits help us see how different models perform in diagnosing ASD accurately. These observations are essential for our main objective: to make ASD diagnosis more accurate and faster, which directly supports the goals of our project.

4. METHODOLOGY

4.1 Feature Correlation and Standardization

I refined the dataset by standardising categorical data to eliminate inconsistencies and simplifying feature names for clarity. The ethnic groups and responses to the test revealed differences in capitalization and naming conventions. To address this, I first changed all of the entries in the 'Ethnicity' and 'Who_completed_the_test' columns to lowercase to ensure consistency throughout the dataset. Following that, I used mappings to combine various representations of the same ethnic group or responder type into a consistent format. For example, variations such as 'Asian' and 'asian' were all mapped to 'asian', removing any ambiguity from the dataset. This standardisation process was critical for precise data analysis, particularly when aggregating data or comparing groups.

To improve the dataset's navigability, I simplified the feature names. Long or complex feature names, such as "A10_Autism_Spectrum_Quotient" and "Global developmental delay/intellectual disability," were shortened to "A10 Quotient" and "GDD/Intellect Disability", respectively. This renaming was intended not only to make the dataset more manageable, but also to keep the names meaningful and reflective of the data they represent. I chose concise yet descriptive names to refer and make analysis easier throughout the research.

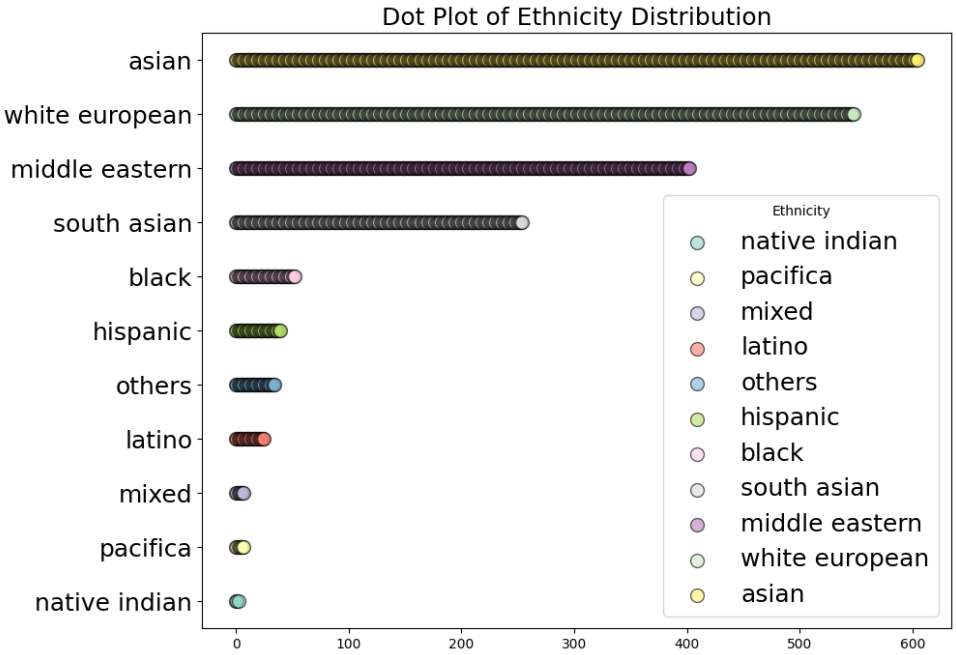

Figure 5: Autism Study Ethnicity Breakdown - Visualized in a Dot Plot

The dot plot in figure 5 visualises the ethnic makeup of the autism dataset, with a strong representation of Asians and White Europeans. This graphical approach gives an intuitive sense of the scale of each ethnic group's representation in the study. Asians are the largest group, with White Europeans and Middle Easterners also well represented, reflecting the dataset's diverse range. Notably, there are a significant number of South Asians, Blacks and Hispanics, demonstrating the dataset's diversity. Although the visibility of smaller groups, such as Pacific Islanders and Native Indians, is limited, it is an important aspect of the dataset because it emphasises the inclusion of a diverse range of ethnic backgrounds. This diversity is critical for research that aims to create universally applicable and culturally sensitive diagnostic and

therapeutic strategies. The dot plot's clear differentiation between the groups allows for immediate and comparative observations of the ethnic diversity found in the study, demonstrating the effectiveness of this visualisation method for conveying demographic data.

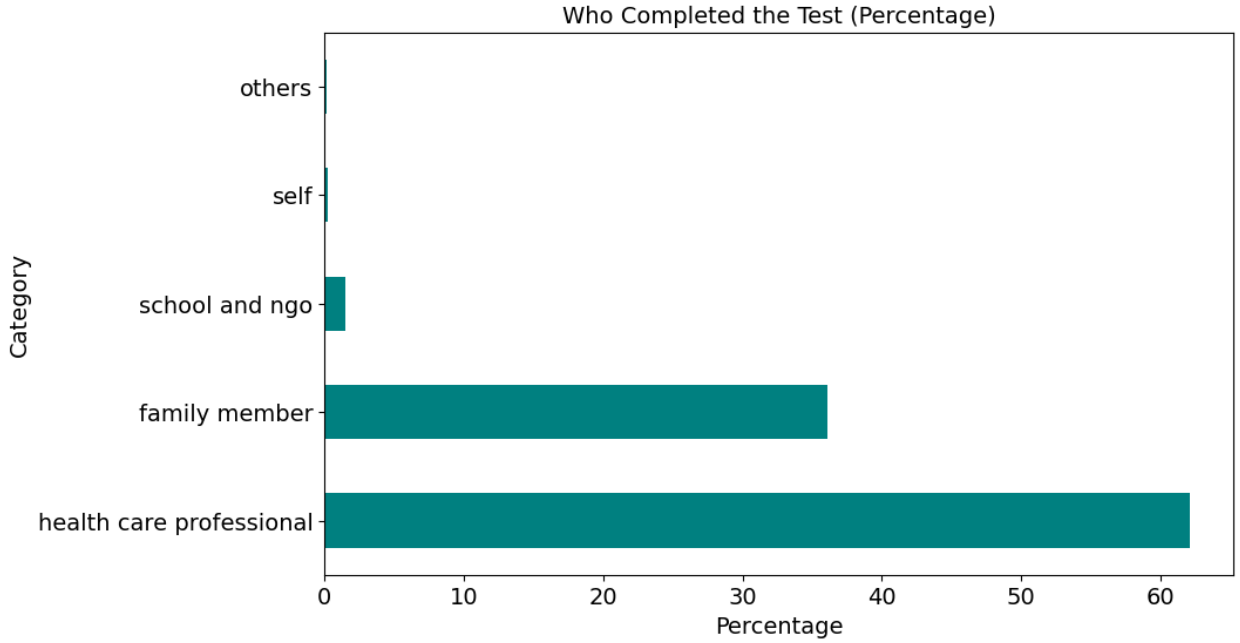

Figure 6: Distribution of ASD Screening Test Respondents Using Bar Chart

Figure 6 shows the distribution of respondents who completed the ASD screening test as a horizontal bar chart. According to the chart, health care professionals complete the most tests with above 60%. Family members follow closely behind with around 35%, indicating that the screening process is heavily influenced by family dynamics. Contributions from schools and Non-Governmental Organizations (NGO), self-responses, and other sources are significantly lower, emphasising the importance of professional and family support in ASD detection efforts. This graphic depicts the collaborative efforts of health care providers and families in navigating the challenges and steps to early ASD diagnosis and management.

4.2 Data Cleaning and Preprocessing

My methodology included several steps to ensure the dataset's integrity and usability for analysis.

Features	With Missing Values	Features	Without Missing Values
CASE_NO_PATIENT'S	0	CASE_NO_PATIENT'S	0
A1	0	A1	0
A2	0	A2	0
A3	0	A3	0
A4	0	A4	0
A5	0	A5	0
A6	0	A6	0
A7	0	A7	0
A8	0	A8	0
A9	0	A9	0
A10 Quotient	0	A10 Quotient	0
Responsiveness scale	9	Responsiveness scale	0
Age_Years	0	Age_Years	0
Qchat_10_Score	39	Qchat_10_Score	0
Language Disorder	0	Language Disorder	0
Learning disorder	0	Learning disorder	0
Genetic_Disorders	0	Genetic_Disorders	0
Depression	1	Depression	0
GDD/Intellect Disability	0	GDD/Intellect Disability	0
Social/Behavior Issues	14	Social/Behavior Issues	0
Autism rating	0	Autism rating	0
Anxiety_disorder	0	Anxiety_disorder	0
Sex	0	Sex	0
Ethnicity	0	Ethnicity	0
Jaundice	0	Jaundice	0
Family_ASD	0	Family_ASD	0
Test_Completed_By	0	Test_Completed_By	0
ASD_traits	0	ASD_traits	0

Table 2: Before and After: Resolving Missing Data in Autism Spectrum Analysis Dataset

Table 2 reflects a targeted approach to handling missing values in key features of the dataset. Using mean imputation for "Responsiveness scale" and "Qchat_10_Score" addresses the continuous nature of these variables, integrating the central tendency into the gaps left by missing data. For "Social/Behavior Issues," a categorical feature, mode imputation ensures the most frequent category fills in, preserving existing distributional characteristics. A singular missing instance in "Depression" is also treated with mode imputation, maintaining consistency in approach. Post-imputation, a re-evaluation of the dataset indicates that the previously observed missing values across the selected columns have been effectively addressed with higher presence of missing in Qchat_10_Score, ensuring a complete dataset conducive to subsequent analytical tasks and preserving the robustness of the dataset for future modeling and interpretation tasks in the context of autism spectrum analysis.

Considering the focus on identifying patterns and characteristics associated with ASD, I determined that some columns, such as 'CASE_NO_PATIENT'S' which simply enumerates patients, did not contribute to the analysis. Consequently, I removed this column from the dataset to streamline the data and focus on variables with direct relevance to ASD traits and diagnosis. This step is crucial in refining the dataset, ensuring that the analysis remains focused on informative features that can lead to meaningful insights.

After streamlining the dataset by removing non-contributory columns such as 'CASE_NO_PATIENT'S' to better focus on variables directly related to ASD traits and diagnosis, I addressed the issue of duplicate entries. Recognising that duplicates could distort the analysis by overrepresenting specific data points, I found and removed 656 duplicate rows. This critical step ensured that each dataset entry represented a unique case, preserving the data's integrity and diversity. The removal of these duplicates, as evidenced by a subsequent count of zero remaining duplicates, was critical in improving the dataset's quality, ensuring that subsequent analyses yield reliable and unbiased insights into ASD characteristics and patterns.

Moving to the next part, I identified and managed outliers that had the potential to skew the analysis results. I used the Interquartile Range (IQR) method to calculate the lower and upper bounds for each numerical column and identified values that fell outside of this range as outliers. These outliers were then clipped to the nearest boundary, preserving the data distribution while eliminating extreme values that could distort model training.

Column	Outliers Before Handling	Outliers After Handling
Language Disorder	277	0
Learning disorder	262	0
Genetic_Disorders	321	0
Depression	279	0
GDD/Intellect Disability	280	0
Social/Behavior Issues	280	0
Autism rating	247	0
Anxiety_disorder	283	0
Jaundice	104	0

Table 3: Comparive Analysis of Dataset Before and After Outliers Handling

The table 3 showcases the comprehensive management of outliers across a range of features in the dataset, effectively demonstrating the IQR method's capability to normalize data. Features such as "Language Disorder," "Learning Disorder," and "Social/Behavior Issues" first exhibited many outliers, which were successfully reduced to zero following the IQR adjustment. This method, which re-aligns data points beyond 1.5 times the IQR from the 25th and 75th percentiles to the nearest accepted value, has substantially decreased variation without sacrificing data integrity, allowing for more accurate statistical analysis. Critical features including "Genetic Disorders" and "Anxiety disorder" have also seen a significant reduction in outlier counts, suggesting that the data is now more representative of the central tendency, crucial for features like "Depression" and "Autism rating." The table format succinctly captures the before-and-after states of outlier

treatment, underscoring the transformation of the dataset through the effective application of the IQR method.

4.3 Feature Engineering

During the data preprocessing phase, I performed detailed encoding of the dataset to convert categorical variables into a format suitable for machine learning algorithms. Initially, binary columns such as 'Speech Delay/Language Disorder', 'Learning Disorder', 'Genetic Disorders', and 'Depression', among others, were mapped to numeric values, with 'Yes' translating to 1 and 'No' to 0. This binary encoding simplified the dataset's structure, allowing algorithms to more effectively interpret categorical data points.

After applying one-hot encoding to the 'Sex', 'Ethnicity', and 'Test Completed By' columns in our dataset, we successfully transformed these nominal variables into binary columns for each category. For instance, the 'Ethnicity' column is now split into separate columns like 'Ethnicity native Indian', 'Ethnicity White European', etc., with rows marked as True or False to indicate the respondent's ethnicity. This method effectively prevents any numerical interpretation of categorical data, ensuring each category is treated as a distinct entity. This approach enhances our model's accuracy by providing a clear, unambiguous representation of each respondent's characteristics.

The heatmap in Figure 7 illustrates the correlations among various features, using red shades for positive correlations and blue for negative correlations. Since ASD traits is my target variable, I can see strong positive correlations with ASQ (A1 to A9), suggesting that these often occur together. The Qchat 10 Score also shows a positive correlation, indicating it's closely aligned with ASD traits. On the other hand, I notice that the A10 Quotient has a significant negative correlation with ASD traits, suggesting it measures something inversely related to the ASD traits. Conditions like Language Disorder and Genetic Disorders exhibit positive correlations with ASD traits, implying they tend to coexist with ASD. I also see that Anxiety disorder and Jaundice have weaker correlations, suggesting they are less related to ASD traits. This heatmap helps me understand the complex relationships among these features and how they connect to ASD traits, providing valuable insights for my analysis.

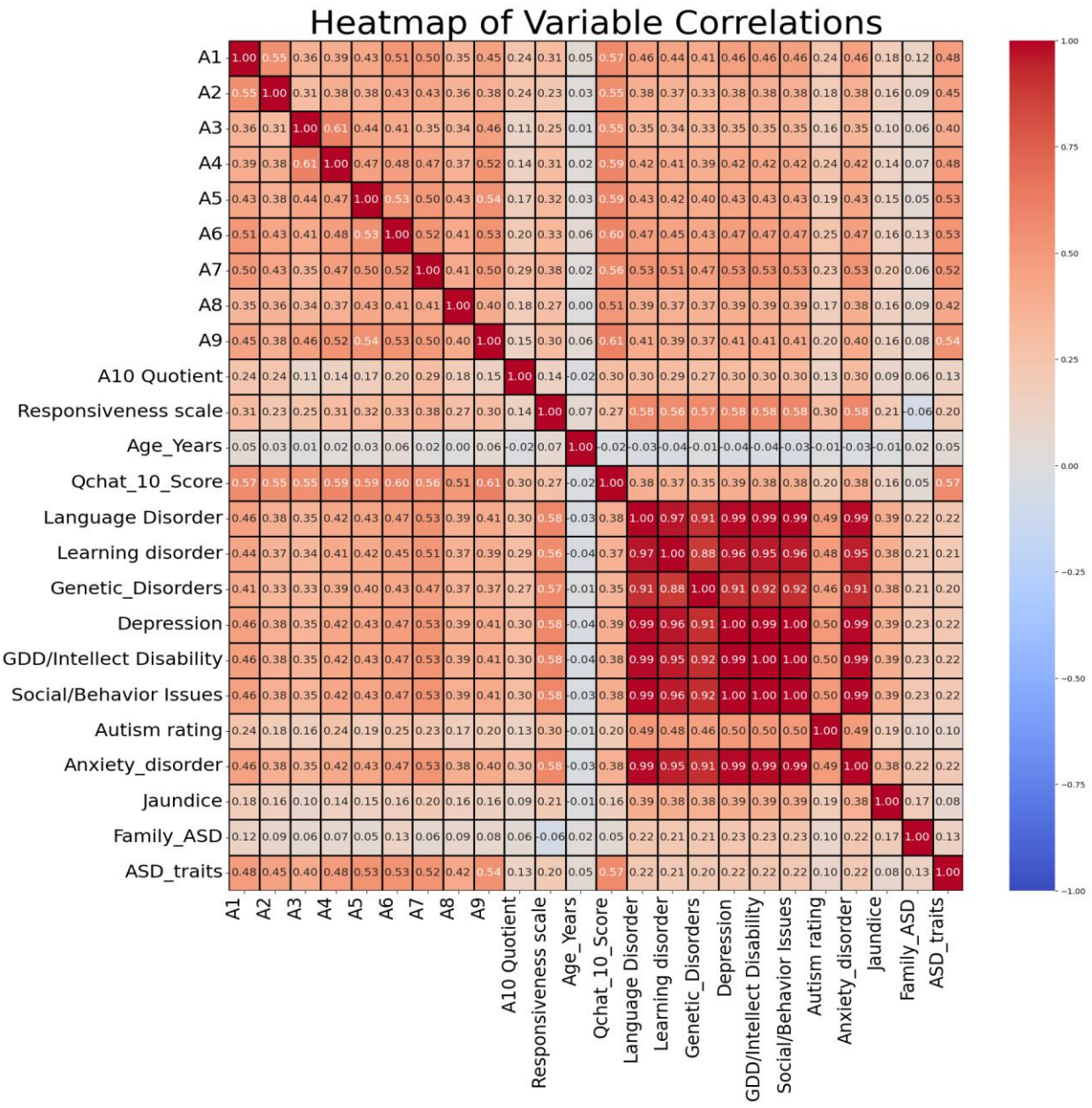

Figure 7: Heatmap of Features Correlations – Visualizing Relationships Between Variables

4.4 Data Preparation

The project commenced with the crucial step of splitting the dataset into training and testing sets, allocating 80% of the data for training and 20% for testing. This specific 80-20 split was chosen to ensure that the model had access to a substantial volume of data for learning while still reserving a significant portion for unbiased evaluation. The primary goal was to mimic real-world application scenarios as closely as possible, where the model's ability to generalize and perform on unseen data is paramount.

4.5 Feature Scaling

Following the split, feature scaling was implemented using the StandardScaler. This step is indispensable for models sensitive to the magnitude of input features. Scaling ensures that all features contribute equally to the model training process, eliminating any bias that might arise due to varying scales of data features. The scaler was fitted only on the training set to prevent data leakage and to ensure the model's performance reflects its capability to process unseen data accurately.

4.6 Dimensionality Reduction with PCA

After scaling, I used Principal Component Analysis (PCA) to reduce the dataset's dimensionality while maintaining 95% of the variance. This threshold was deliberately chosen to strike a balance between simplifying the dataset and preserving critical information, both of which are required for ML models to effectively diagnose ASD. By retaining 95% of the variance, the process aims to reduce information loss while focusing on the most informative aspects of the data. PCA was performed on the scaled training data and then applied to both the training and testing sets. This methodical scaling approach followed by PCA ensures that the ML models are trained on data optimised for both high performance and generalizability, which is critical for capturing the nuanced manifestations of ASD.

The preprocessing steps, which included scaling, PCA, and encoding, prepared the dataset for training with ML models like SVM, RF, and LR. These steps are intended to improve accuracy and efficiency while reducing overfitting risks. Encoding transforms categorical data, improving interpretability for these algorithms; however, it is important to note the effect of one-hot encoding on data sparsity, which is especially relevant for SVM. This preparation ensures that the dataset is suitable for these models, laying the groundwork for reliable ASD diagnosis tools.

4.7 SVM Model

After preparing the dataset with PCA to reduce its dimensionality and emphasize its most informative features, I proceeded to apply the SVM model. This choice was driven by SVM's robust performance with high-dimensional data, making it an excellent match for our PCA-optimized dataset. Initially, I chose a linear kernel for the SVM because it is simple and effective in linearly separable data. The regularisation parameter, C, was set at 0.1. This decision aimed to strike a delicate balance between minimising training error and ensuring the model's generalizability to new data. A lower C value was deliberately chosen to prioritise a wider margin between classes, allowing some misclassifications in favour of a model that is less prone to overfitting. To ensure the reproducibility of the results, the random state parameter was fixed at 42.

With the SVM model initialized, I refined it through strategic feature selection using Recursive Feature Elimination (RFE), which systematically evaluates each feature's contribution by iteratively removing the least significant ones based on their model weights, allowing us to focus solely on the most impactful features. The decision to retain the top 15 features was deliberate, with the goal of striking the best balance between model complexity and performance. To further understand which features had the most significant impact on the principal component, I created a graph depicting the feature contributions to PCA component 1. This bar chart provided a visual representation of the importance of various features, with taller bars indicating higher contributions. I set a threshold of 0.2 for selecting significant features, focusing on those with a loading score above this value.

Following this, I used Grid Search for hyperparameter tuning, experimenting with a wide range of parameter combinations. The parameter grid contained multiple values for the regularisation parameter 'C', different kernel types ('linear', 'polynomial', 'radial basis function', 'sigmoid'), and 'gamma' coefficients ('scale', 'auto'). This exhaustive search sought to identify the best settings for the SVM classifier, with each combination evaluated using 5-fold cross-validation. To evaluate the optimized model, I used metrices like accuracy, precision, recall, and a detailed classification report. A confusion matrix was created to visualize the model's predictive abilities, showing correct and incorrect classifications. This evaluation offered comprehensive insights into the model's strengths and areas for improvement.

For further validation, I ran cross-validation on the training data, obtaining an accuracy measure that accounted for variance across different dataset splits. This step was critical for confirming the model's stability and reliability. Additionally, precision-recall curves for both the training and test sets were plotted, illustrating the balance between precision and recall at different threshold levels. These analyses are particularly useful for understanding the model's performance, especially in scenarios where managing false positives and false negatives is crucial.

4.8 RANDOM FOREST

In developing the RF classifier for ASD diagnosis, I configured it with 100 decision trees and fixed the random state at 42. This setup ensured consistent results across multiple trials and allowed me to capitalize on the RF's ability to handle complex datasets, like those reduced by PCA. This initial step was crucial because it set the foundation for a model capable of identifying complex patterns within the data—an essential characteristic for accurate ASD detection.

When I started training the RF model, my initial focus was on identifying the most important features. I used the Gini index to evaluate each feature's contribution by measuring how much each reduced confusion or "impurity" during data splits. This approach helped me create a ranked list of features, allowing me to see which ones were most predictive for ASD. With this information, I could concentrate on the most crucial features, making the model more efficient and easier to understand. Using the feature importance scores, I selected the top 15 features to streamline the model. This reduction in complexity helped improve the model's diagnostic accuracy while removing unnecessary details. Additionally, I used these scores to create a plot showing the features' contributions to the top PCA component. By focusing on these top 15 features, the model became more transparent, which is crucial in clinical settings where it's important to understand how a model arrives at its predictions.

To tune the RF classifier for ASD diagnosis, I adjusted key hyperparameters to balance model complexity and generalization. I optimized the number of trees (estimators), ranging from 50 to 100, for robustness. To manage overfitting, I explored different maximum depth values (from 5 to 20), ensuring the trees weren't too complex but retained enough capacity to capture meaningful patterns. I varied the minimum samples split (between 5 and 20) to avoid granular splits, while minimum samples leaf (10 or 15) ensured leaf nodes had adequate samples, preventing overfitting. The maximum features parameter, indicating the maximum features considered at each split, was set to square root(sqrt) or to maintain tree diversity. These adjustments were optimized using a Grid Search CV with 10-fold cross-validation, leading to a robust RF model.

Having refined the RF model, I proceeded to evaluate its performance using various metrics like accuracy, precision, recall, and a detailed classification report. I also created confusion matrices to

visualize the model's ability to correctly and incorrectly classify cases, which helped me identify the model's strengths and weaknesses. To further validate the model's robustness, I used cross-validation to obtain an accuracy measure that accounted for variance across different dataset splits. This step was essential to confirm the model's stability and reliability. Additionally, precision-recall curves for both training and test sets were plotted to understand the trade-offs between precision and recall at different threshold settings.

4.9 LOGISTIC REGRESSION

To build the LR classifier for ASD diagnosis, I started by applying RFE within an LR framework. The goal was to isolate the top 15 most crucial features from the PCA-reduced dataset, enhancing the model's interpretability and effectiveness. By selecting only the most significant features, I aimed to avoid overfitting, ensuring the model's diagnostic accuracy wasn't compromised by less relevant data.

The model’s effectiveness was initially validated by calculating the accuracy on both the training and test datasets, providing immediate feedback on its predictive power and generalization capability. Further refinement was pursued through hyperparameter tuning, utilizing a Grid Search to optimize the regularization parameter (C), solver types, and maximum iterations. The C parameter was tuned within a wide range from 10^{-8} to 10^{8} to balance model flexibility with overfitting risk. Solver types such as 'liblinear' and 'lbfgs'(Limited-memory Broyden-Fletcher-Goldfarb-Shanno) were explored to find the most suitable one for this model, with maximum iterations set at 500, 1000, 5000, or 10,000 to ensure proper convergence

Feature importance was analyzed by assessing the coefficients of the LR model, providing insights into which features most influenced the classifier's predictions. I visualized these coefficients in a bar graph, highlighting the relative importance of each of the 15 features selected through RFE. Additionally, I investigated the features contributing to PCA component 1 by creating a graph that plotted the feature loadings. This analysis included a threshold of 0.2, allowing me to focus on the features with the highest impact on the principal component. By examining these feature loadings, I gained a deeper understanding of the data's structure, guiding further refinement of the LR model.

With the optimal parameters established, I evaluated the model's effectiveness by measuring accuracy on both the training and test datasets. Further assessments were done using precision, recall, and a detailed classification report, giving a comprehensive overview of the model's performance. Finally, I employed cross-validation with 10 folds to ensure the model's robustness and consistent performance across different subsets of data.

5. RESULT

Following the development and optimisation of the SVM, RF, and LR models for ASD diagnosis, the results section provides a thorough evaluation of their performance. The evaluation assesses each model's effectiveness using key metrics such as accuracy, precision, recall, and F1-score. These metrics offer a comprehensive view of the model's accuracy and its ability to correctly identify positive and negative cases effectively.

5.1 Metric Selection

The choice of metrics is crucial for evaluating the models' performance. Here is why these specific metrics were chosen and how they align with the project objectives:

- **Accuracy**: Accuracy measures the proportion of correct predictions out of the total number of predictions. It's calculated as follows:

  $$\text{Accuracy} = \frac{TP + TN}{TP+FP+TN+FN}$$

  - **Technical Expansion**:
    - TP: True Positives—cases correctly predicted as positive.
    - TN: True Negatives—cases correctly predicted as negative.
    - FP: False Positives—cases incorrectly predicted as positive.
    - FN: False Negatives—cases incorrectly predicted as negative.
  - **Importance:** Accuracy provides an overall measure of a model's performance, useful for understanding its effectiveness in general
- **Precision:** Precision indicates the proportion of true positive predictions out of all positive predictions. Here's the formula:

$$\text{Precision} = \frac{TP}{TP+FP}$$

  - **Importance**: Precision is crucial in medical contexts to minimize false positives, reducing the risk of misdiagnosis or unnecessary treatments. A high precision score indicates that most positive predictions are accurate.

- **Recall**: Recall measures the proportion of true positives out of all actual positive cases. It's calculated as follows:

$$\text{Recall} = \frac{TP}{TP+FN}$$

  - **Importance**: High recall is essential in medical diagnostics, ensuring the model does not miss positive cases, leading to early detection.

- **F1-score**: The F1-score is the harmonic mean of precision and recall, offering a balance between the two. It is calculated as follows:

  $$\text{F1-Score} = 2 \times \frac{\text{Precision} \times Recall}{\text{Precision} + Recall}$$

- **Importance**: The F1-score balances precision and recall, providing a single measure that accounts for both accuracy and sensitivity. This is particularly valuable in medical contexts, where both aspects are critical.

These metrics were selected because they collectively cover various aspects of model performance, providing a well-rounded evaluation that meets the project's objective of accurate ASD diagnosis.

5.2 Presentation of Results

The results are presented using various visualization techniques, each chosen for its ability to provide clear insights into the models' performance:

- **Confusion Matrices** are used to visually represent the true positive, true negative, false positive, and false negative cases. This visualization helps to identify where the models are prone to errors, guiding further tuning and improvement.
- **Feature Importance Plots** highlight which components and features contribute most to model predictions, aiding in understanding and refining the models' focus.
- **Precision-Recall Curves** illustrate the trade-off between precision and recall, offering a way to assess model sensitivity and specificity, critical for medical diagnostics.

Model	Training Accuracy	Test Accuracy	Class	Precision	Recall	F1-score
SVM	0.9022	0.8947	0	0.86	0.88	0.87
			1	0.92	0.91	0.91
RF	1.0000	0.9586	0	0.96	0.93	0.95
			1	0.96	0.98	0.97
LR	0.8946	0.8835	0	0.84	0.87	0.85
			1	0.91	0.90	0.90

Table 4 : Initial Performance Metrics of ML Models on Training and Test Data

From Table 4, the analysis of SVM, RF, and LR models in diagnosing ASD reveals significant insights about each model’s diagnostic capabilities. The RF model demonstrates a test accuracy of 0.9586, signaling high reliability in diagnosing ASD, complemented by a precision of 0.96 and a recall of 0.98 for Class 1. This highlights its ability to effectively identify true positive ASD cases with minimal false positives. However, the RF model also shows a training accuracy of 100%, which raises concerns about potential overfitting; this perfect training score might indicate that the model is too closely fitted to the training data, potentially limiting its generalizability to new, unseen data. The SVM model follows with a test accuracy of 0.8947, maintaining a good balance of precision (0.92) and recall (0.91). The LR model reports a slightly lower test accuracy of 0.8835, with precision and recall at 0.91 and 0.90 respectively, slightly lagging behind the other models but still showing solid performance. To further understand where each model excels or falls short,

examining their confusion matrices can provide deeper insights into their specific classification strengths and weaknesses.

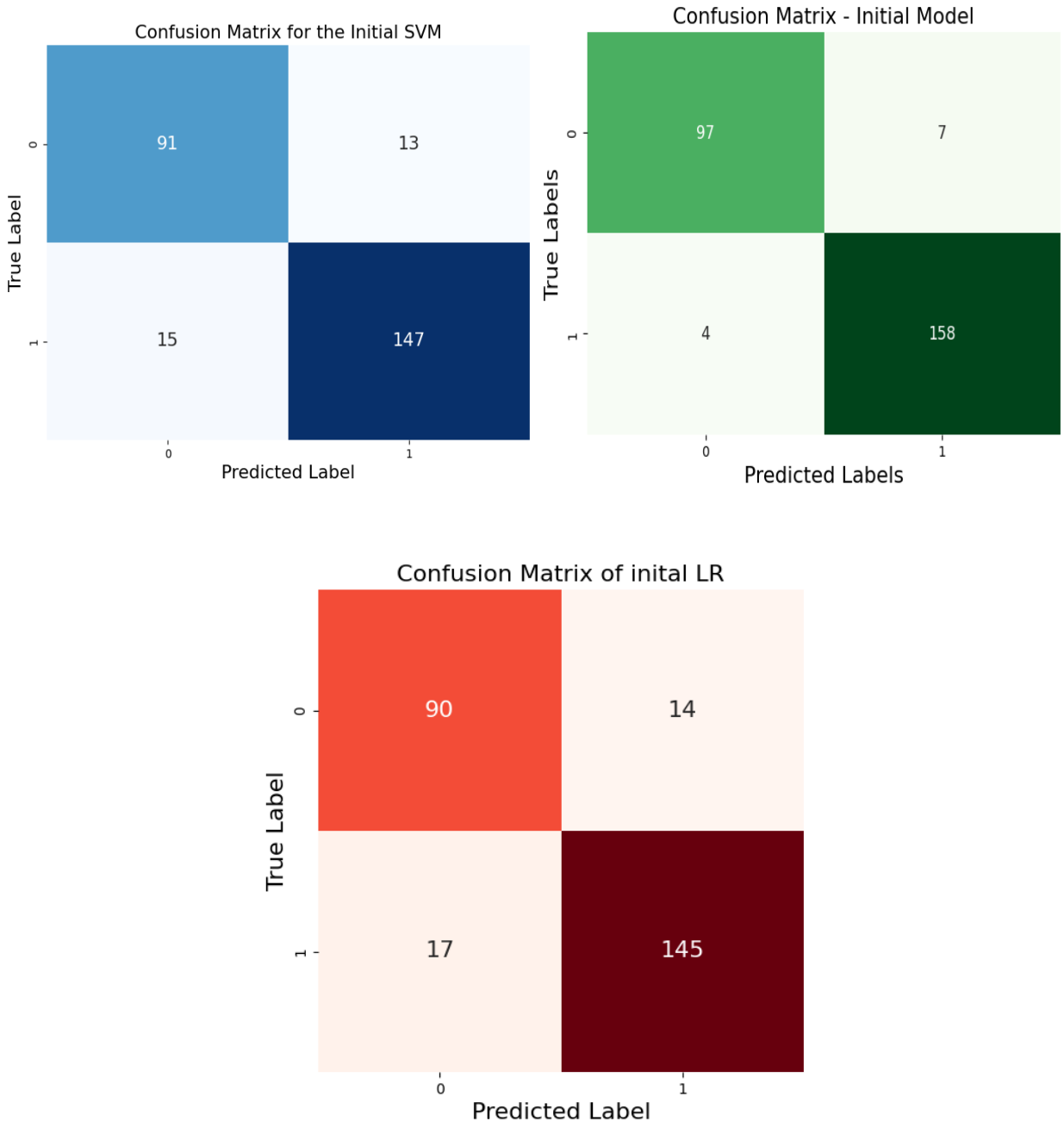

Figure 9: Confusion Matrices for Initial Model Results

The confusion matrices for the initial SVM, RF, and LR models provide useful information about their initial accuracy in diagnosing. Figure 9 shows that the SVM confusion matrix contains 15 false negatives and 13 false positives, indicating the need for additional tuning to reduce these

errors. While the model generally performs well, these misclassifications highlight areas for further optimisation to improve accuracy and reduce the risk of misdiagnosis or missed ASD cases.

In contrast, the RF confusion matrix performs better, with only four false negatives and seven false positives. This lower rate of misclassification suggests a more reliable predictive capability, though errors can still be reduced further. In Figure 9, the LR confusion matrix has a higher number of misclassifications, with 17 false negatives and 14 false positives. This implies that the LR model may require significant optimisation to achieve a more balanced precision and recall. These findings highlight the need for additional refinement to improve the models' accuracy and reliability for ASD diagnosis.

The feature importance plots for the SVM, RF, and LR models show which components and features have the greatest impact on the initial model's predictions for ASD. Each graph offers unique insights that aid in further refinement and hyperparameter tuning.

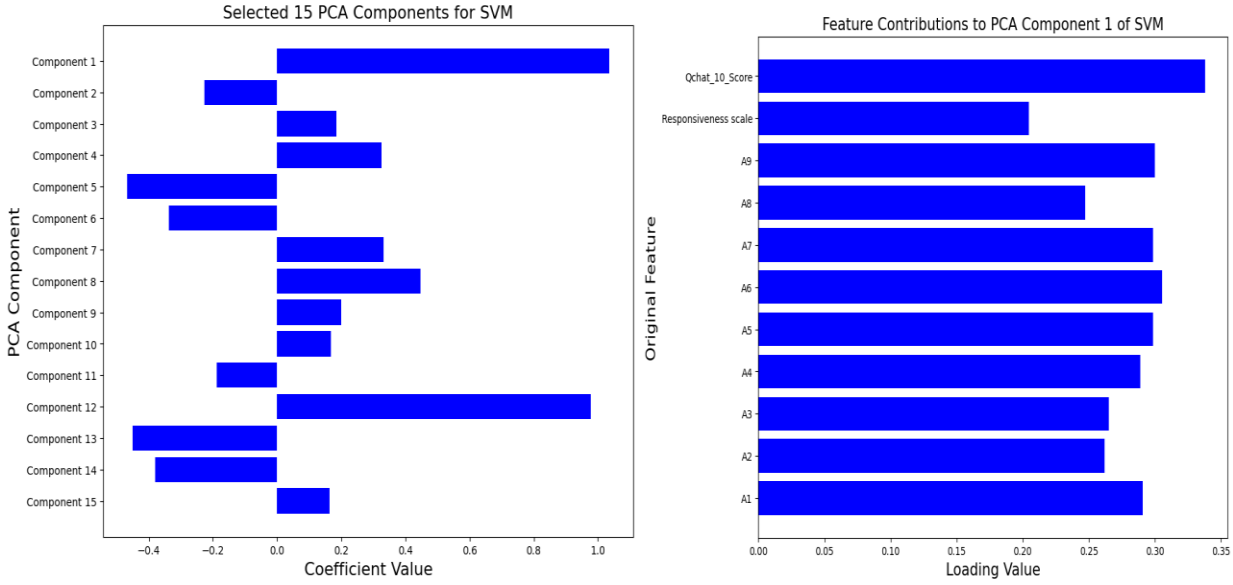

Figure 10: Feature Importance and Contribution to Component 1 of SVM

In Figure 10, the feature importance for the SVM model demonstrates the importance of different PCA components in diagnosing ASD. The first figure shows the selected PCA components, with the x-axis representing the coefficient values, which indicate the relative impact of each component on the SVM model's predictions, and the y-axis listing the components. Higher coefficient values indicate a stronger influence, with Component 1 having the greatest impact on the SVM's performance. The second figure depicts the feature contributions to PCA Component 1, with the x-axis representing the loading scores and the y-axis indicating the significant features. A 0.2 threshold helps to isolate the most influential features, revealing that Qchat 10 Score and Responsiveness Scale are significant contributors to the first principal component. This insight guides future optimization efforts by highlighting the features and components most crucial to the SVM model's accuracy in predicting ASD.

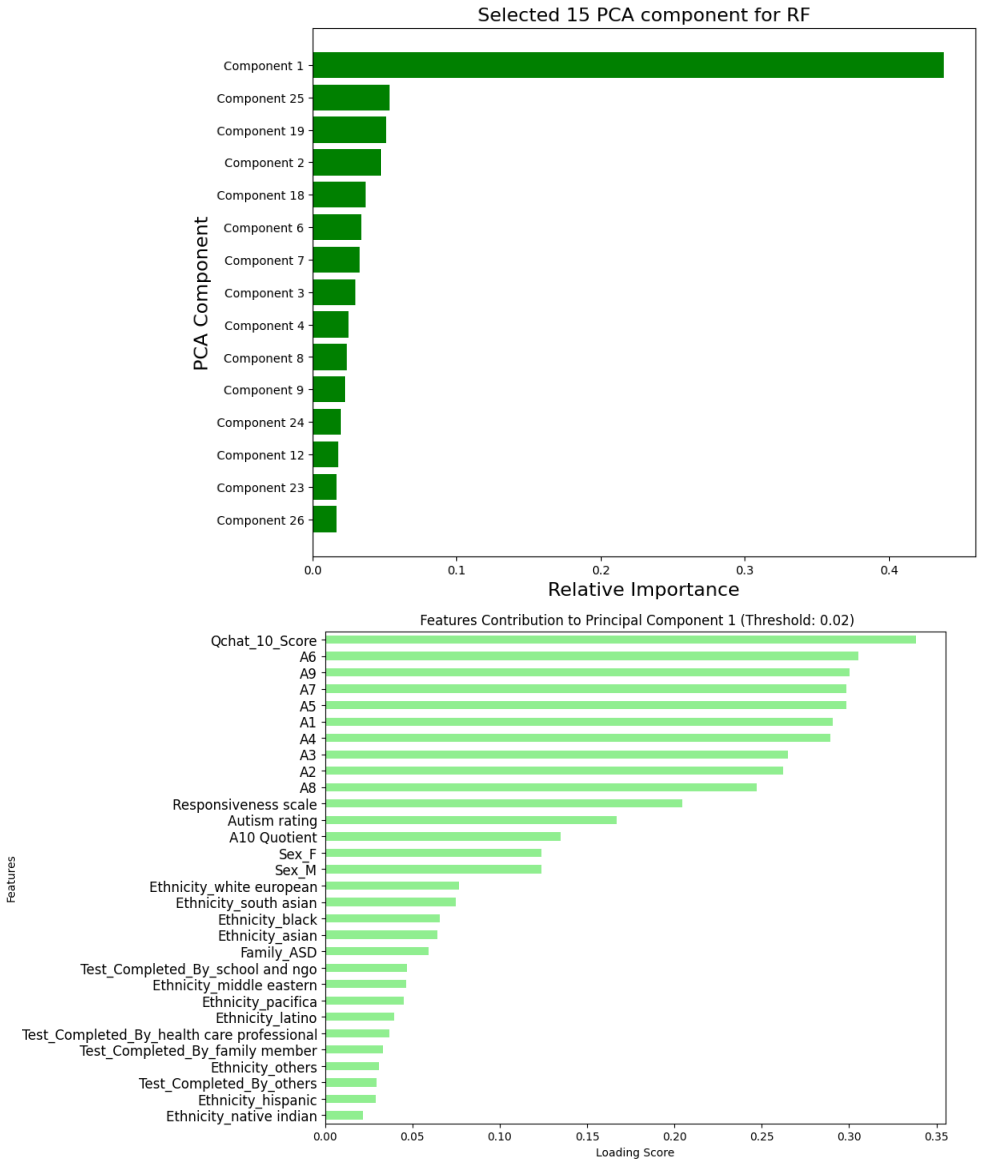

Figure 11: Feature Importance and Contribution to Component 1 of RF

The RF model's feature importance in Figure 11 reveal the key PCA components and features that contribute to its diagnostic predictions for ASD. The first figure depicts the selected PCA components, with the x-axis representing coefficient values that indicate the relative impact of each component on the RF model's predictions and the y-axis displaying the components. Higher coefficient values indicate a stronger influence, with Component 1 having the greatest impact on the model's performance. The second figure depicts the feature contributions to PCA Component 1, with the x-axis representing the loading scores and the y-axis displaying the significant features. A threshold of 0.02 helps focus on the most impactful features, demonstrating that the Responsiveness Scale and Autism Rating (A1 – A9) are major contributors.

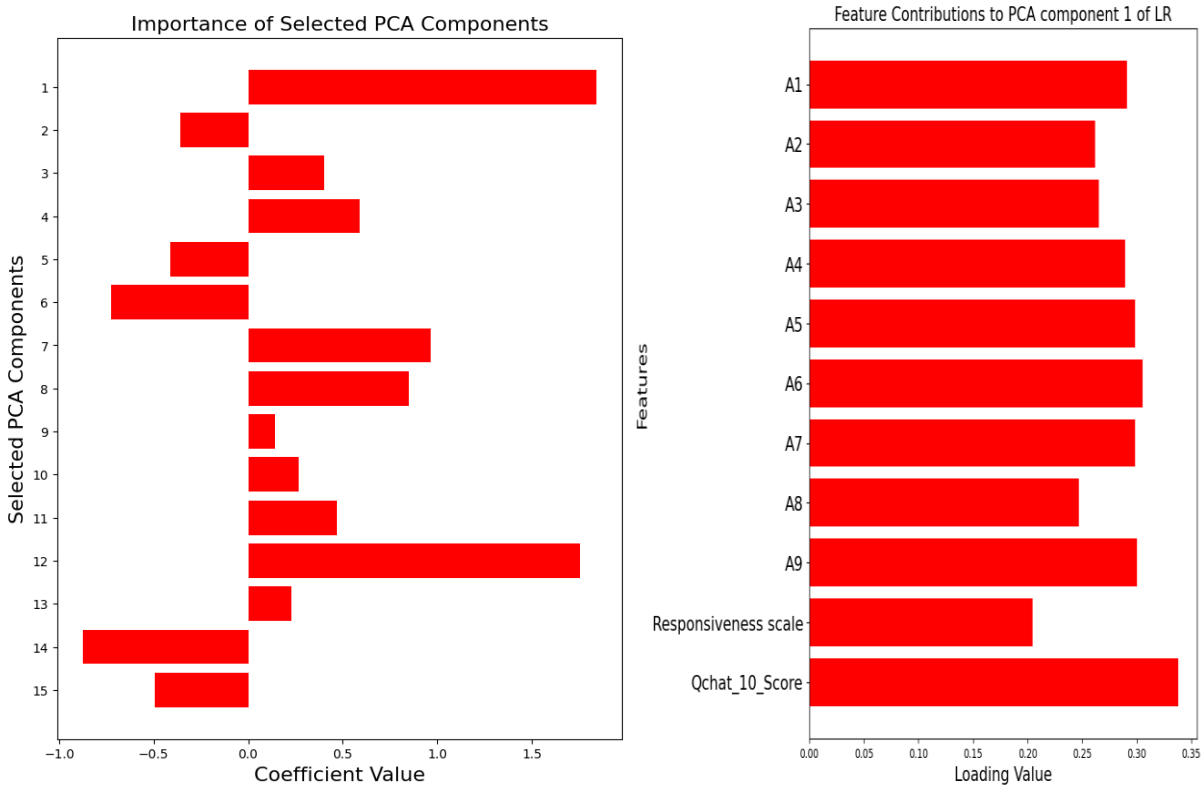

Figure 12: Feature Importance and Contribution to Component 1 of LR

The LR model's feature importance Figure 12 reveals the key PCA components and features that contribute to ASD diagnosis predictions. The first figure shows the selected PCA components, with the x-axis representing the coefficient values, which indicate the relative impact of each component on the LR model's predictions, and the y-axis listing the components. The high coefficient value for Component 1 indicates that it has a significant impact on the model's performance. The second figure shows the feature contributions to PCA Component 1, with the x-axis representing loading scores and the y-axis listing the significant features. A 0.2 threshold identifies the most critical elements, highlighting A1, A2, A3, and Qchat 10 Score as significant contributors to the first principal component. These feature importance plots for the SVM, RF, and LR models will help guide future hyperparameter tuning efforts by indicating which components and features are most important for improving model accuracy and reliability in diagnosing ASD.

5.3 HYPERPARAMETER TUNING RESULTS

Following the initial feature importance analysis, which identified the critical components and features influencing ASD predictions, the next step involved hyperparameter tuning to improve model performance. Grid Search with cross-validation was used to find the best configurations for each model, maximising accuracy while ensuring robustness. Here is an overview of the hyperparameter tuning results for three models.

- **SVM Model Tuning**:
  - **Regularization parameter (C):** Set to 1, indicating a balanced approach between maximizing the margin and minimizing the classification error.
  - **Gamma:** 'Auto' setting allows automatic calculation based on the input data, potentially improving generalization.

    - **Kernel:** 'RBF' kernel chosen for capturing complex relationships in the data.
- **RF Model Tuning**:
    - **Max Depth:** Limited to 10 to prevent overfitting.
    - **Max Features:** 'sqrt' setting selects a diverse set of features at each split.
    - **Min Samples Leaf:** Set to 10 for robustness against noise.
    - **Min Samples Split:** Requires a minimum of 5 samples to split a node, controlling complexity.
    - **Estimators:** 100 trees used in the forest ensemble for a balance between complexity and performance.
- **LR Model Tuning**:
    - **Regularization parameter (C):** Tuned to 3.678 for controlling regularization strength.
    - **Max Iterations:** Set to 500 to ensure convergence.
    - **Solver:** 'Liblinear' solver chosen for efficient handling of regularization.

Model	Training Accuracy	Test Accuracy	Class	Precision	Recall	F1-score
SVM	0.9868	0.9624	1	0.95	0.99	0.97
			0	0.98	0.92	0.95
RF	0.9755	0.9511	1	0.95	0.98	0.96
			0	0.96	0.91	0.94
LR	0.8937	0.8835	1	0.92	0.89	0.90
			0	0.83	0.88	0.85

Table 6: Initial Performance Metrics of ML Models on Training and Test Data

Hyperparameter tuning of the SVM, RF, and LR models significantly enhanced their diagnostic accuracy for ASD, as detailed in Table 6. The SVM model showed a substantial improvement, with test accuracy increasing from 0.8947 to 0.9624, accompanied by gains in precision and recall for both ASD-positive and ASD-negative cases. This indicates a robust enhancement in the model's ability to accurately identify and classify ASD instances, reducing both false positives and false negatives. The RF model, previously suspected of overfitting due to its perfect training accuracy, also showed an improved performance. After tuning, its training accuracy reduced to 0.9755, reducing the gap with its test accuracy now at 0.9511, which signifies a reduction in overfitting. This adjustment suggests that the model is now better generalized to new, unseen data, enhancing its reliability.

The fine-tuning process not only boosted the overall accuracy of the models but also optimized the balance between precision and recall, crucial for reliable ASD diagnosis. The improvements post-tuning highlights the importance of thorough parameter optimization in ML model development. By adjusting parameters like the regularization strength in LR, the kernel type in SVM, and tree depths in RF, each model was tailored to more effectively generalize to unseen data and to enhance their diagnostic accuracy. The increased precision and recall across the models

demonstrate a deeper understanding of their predictive capabilities, underlining the significance of hyperparameter tuning in practical medical applications.

Regarding the LR model, although it showed a slight decline in test accuracy from 0.8937 to 0.8835 and a decrease in precision for class 0 from 0.88 to 0.83, the adjustments in hyperparameters helped fine-tune its performance metrics, notably improving recall. These refinements, while modest compared to the other models, highlight areas for further enhancement and the potential benefits of continuous optimization. The detailed analysis of each model's performance before and after tuning underscores the critical role hyperparameter tuning plays in enhancing the diagnostic performance of models, leading to more accurate and reliable ASD diagnoses in clinical settings.

Model	CV Accuracy(Mean)	CV Std.Deviation
SVM	0.9680	0.0178
RF	0.9643	0.0169
LR	0.8861	0.0475

Table 7: Comparison of Cross-Validation Accuracy and Standard Deviation Across Models

Table 7 gives the CV results for the SV, RF, and LR models which reveal information about their consistency and stability. The RF model had a mean CV accuracy of 0.9643 and a low standard deviation of 0.0169, indicating consistent performance across data splits. Similarly, the SVM model's mean CV accuracy of 0.9680 and standard deviation of 0.0178 indicate consistent results. On the other hand, the LR model's mean CV accuracy is 0.8861 with a higher standard deviation of 0.0475, indicating more variable and less consistent performance compared to the RF and SVM models. This greater variability suggests that the LR model is less reliable across various test conditions, highlighting the superior stability and consistency of the RF and SVM models in cross-validation settings.

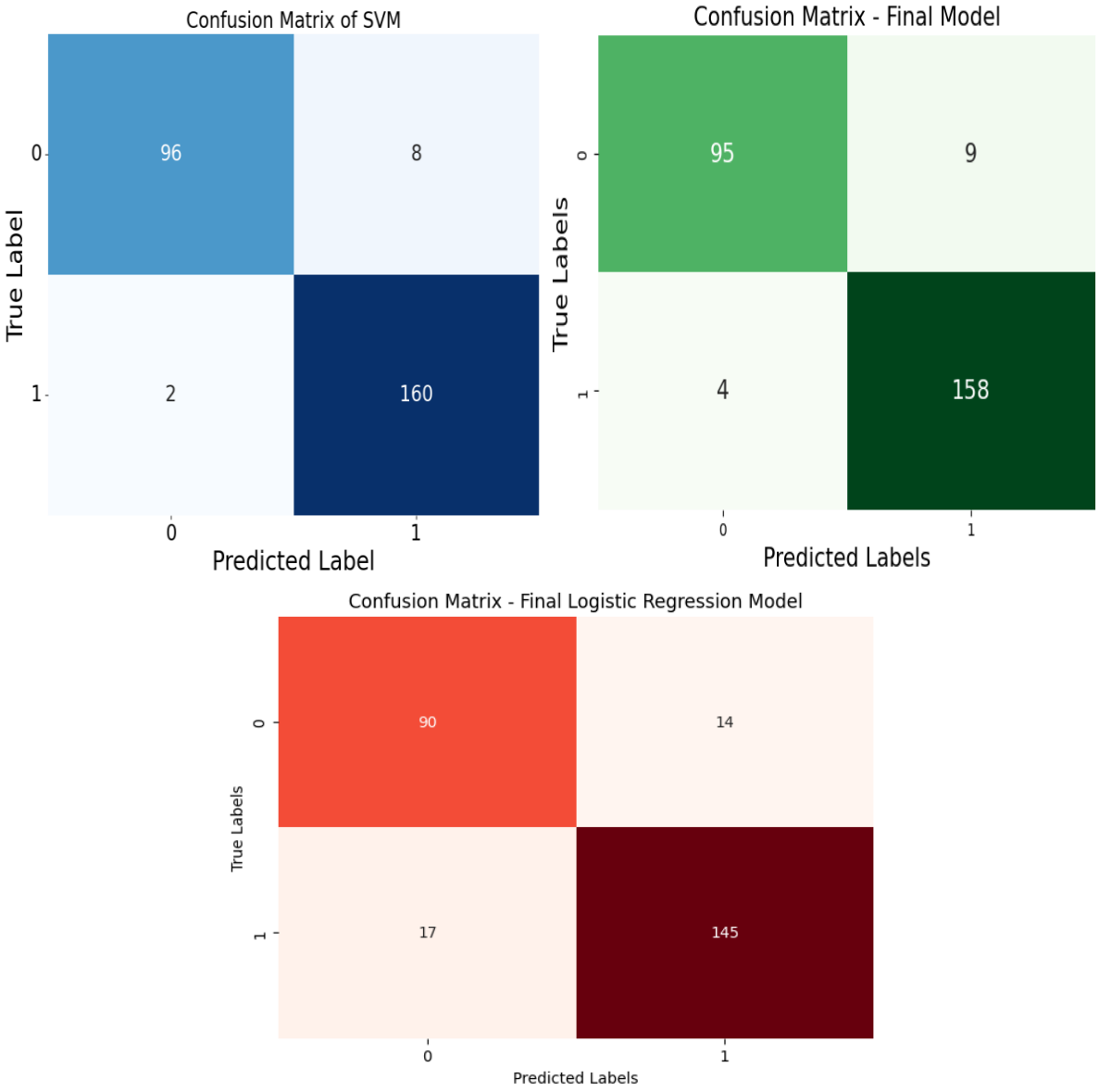

Figure 13: Confusion Matrix of Final models

The final confusion matrices shown in figure 13 of the SVM, RF, and LR models demonstrate how hyperparameter tuning affects model performance. The initial SVM confusion matrix had a mix of true and false classifications, with a moderate number of misclassifications. After tuning, the final confusion matrix showed improved accuracy. There were 91 true negatives, indicating the model's ability to correctly identify non-ASD cases. However, the number of false negatives decreased, suggesting that some ASD cases were still being missed. Despite this, the reduction in overall errors pointed toward improved model performance.

The RF model was already performing well, with 97 true negatives and a low number of false positives and false negatives in the initial confusion matrix. Following hyperparameter tuning, the RF model showed even greater accuracy in the final confusion matrix, with fewer false positives (9) and a stable number of true positives (158). This slight increase in false positives suggested a small drop in specificity, but the consistent true positives indicated robust reliability. The LR

model had a tougher start, with more misclassifications compared to the other two models. Despite hyperparameter tuning, the final confusion matrix still displayed a relatively high number of misclassifications, with 14 false positives and 17 false negatives. This limited improvement indicated that further refinement might be necessary for the LR model to match the accuracy levels of SVM and RF.

6. EVALUATION AND DISCUSSION

The models analyzed exhibited varying levels of performance, with RF emerging as the most robust overall. While the SVM attained the highest test accuracy of 0.9624, RF demonstrated a more balanced performance with an accuracy of 0.9511, coupled with consistent precision and recall. This is evident from RF's precision-recall curve depicted in Figure 14, maintaining high precision across various recall levels, indicative of its robust predictive capability for ASD diagnosis.

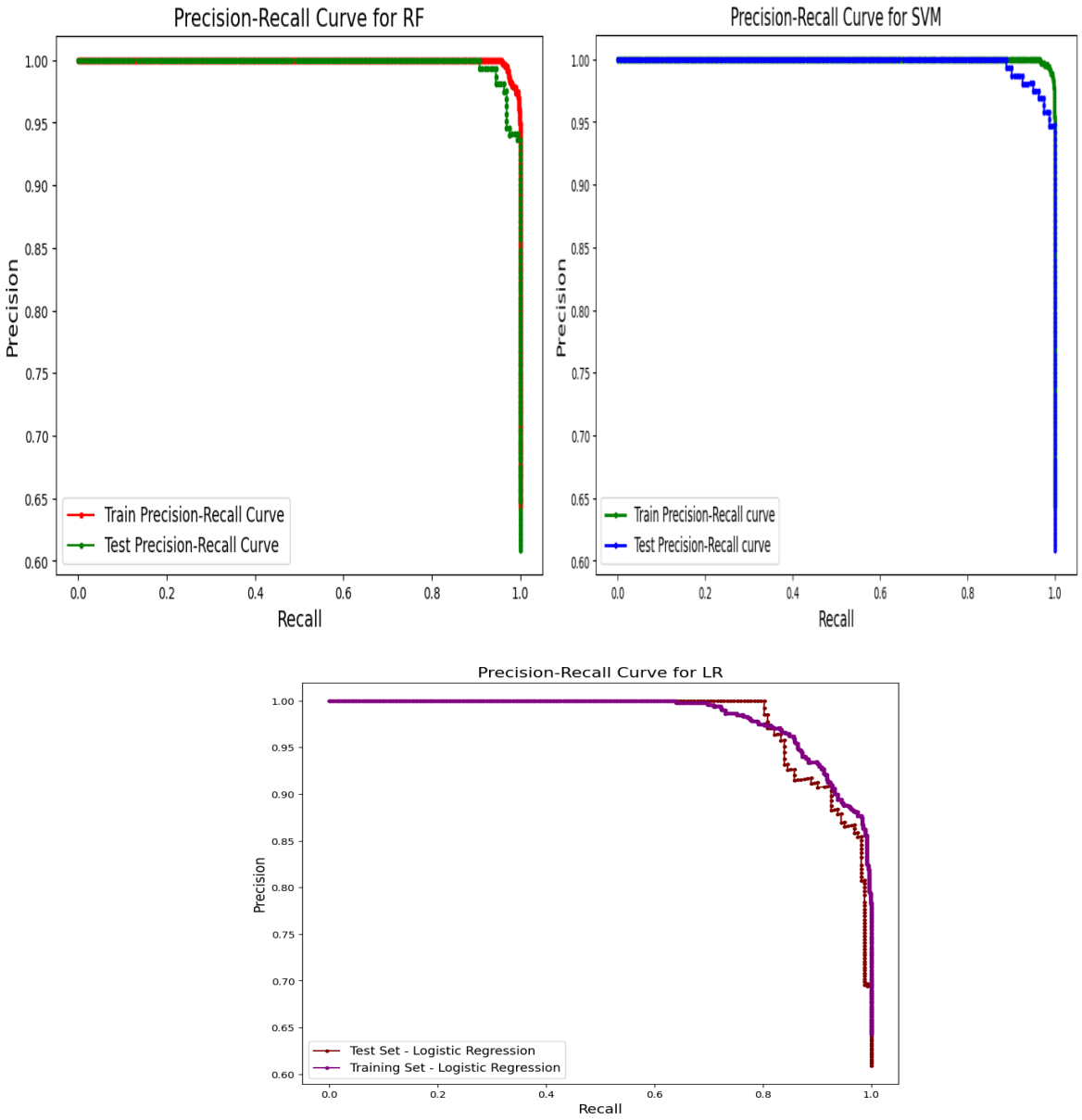

Figure 14: Comparative Analysis of Precision-Recall Metrics for Three Models

As from Figure 14, RF's strength lies in its balanced approach, achieving a consistent reduction in false positives and false negatives post-tuning. This stability suggests that RF is more reliable in

minimizing misclassifications, a critical factor in medical diagnostics. Despite the slight edge in test accuracy for SVM, RF's precision-recall curve demonstrated that it could maintain accuracy while reducing the risk of false positives, making it a more dependable choice. On the other hand, SVM's precision-recall curve showed a slight decline in precision with higher recall, indicating an increased risk of false positives in the test set. This inconsistency suggests that while SVM has a higher test accuracy, its reliability in consistently minimizing misclassifications is not as strong as RF's. The LR model, though with a lower test accuracy of 0.8835 compared to SVM and RF, demonstrated potential for improvement post-hyperparameter tuning. However, its performance remained less reliable due to a higher rate of misclassifications, highlighting the need for further optimization.

In summary, while SVM achieved the highest test accuracy, RF's balanced approach and consistent precision-recall performance made it the best model for ASD diagnosis. The RF model's consistent reduction in false positives and false negatives, along with its robustness in maintaining accuracy, contributed to its superior position among the models tested. This balance between accuracy and reliability is key to RF's success, making it the most dependable choice for ASD prediction.

6.1 Comparing Results with Background Studies

Comparison of project results with existing studies reveals significant similarities. In my project, the RF model performed best, aligning with Akter et al.'s (2021) finding that RF was highly accurate in ASD diagnosis. The SVM in my project also showed balanced performance, consistent with Raj and Masood's (2020) work, in which SVM produced reliable results across multiple age groups. My LR model, while less accurate than RF and SVM, resembles Vakadkar et al.'s (2021) observations, where LR was among the top performers, though requiring additional optimization. This suggests that with further refinement, LR's performance could improve.

My approach to evaluating these models involved methods like confusion matrix analysis and precision-recall curves, offering detailed insights into each model's misclassification rates and performance across varying recall levels. These methods are consistent with Chowdhury and Iraj's (2020) emphasis on feature scaling and attribute evaluation to refine ML models. These comparisons indicate that while RF and SVM are robust for ASD diagnosis, LR needs further tuning to achieve similar accuracy. The studies and my results highlight the need for a tailored approach to ML for accurate ASD diagnosis, providing a pathway for future research to optimize model performance.

Additionally, our analysis highlighted the significance of the \ASQ scale, ranging from A1 to A10, as a key feature for predicting ASD. This scale, commonly observed in individuals with ASD, offers valuable insights into the disorder's complexity. ASQ scores help identify various traits and characteristics associated with ASD, providing valuable guidance for targeted interventions and personalized treatment approaches. However, it's important to acknowledge the limitations inherent in our analysis, including potential biases within the dataset. Future research should prioritize addressing these limitations to deepen our understanding of the relationship between ASQ scores and ASD diagnosis, ultimately advancing the development of more effective diagnostic and intervention strategies.

The results of this project are closely aligned with the project's primary objectives, which are to identify the most effective ML models for detecting ASD and to improve diagnostic accuracy. The success of the RF model, with its balanced precision-recall performance, supports the objective of improving ASD detection through ML. The SVM model, which demonstrated strong generalizability, reinforces the goal of integrating effective ML techniques into diagnostic frameworks. These outcomes suggest that the project achieved its aim of using ML to enhance ASD diagnosis accuracy and reliability.

Regarding the project's application, the results suggest that integrating ML into ASD diagnosis could lead to more accurate and timely detection. The robust performance of RF and SVM implies the effectiveness of ML algorithms in identifying ASD traits and reducing misclassifications, critical for early intervention. By demonstrating that these models can maintain high accuracy while minimizing false positives and false negatives, the results advocate for the use of ML in clinical practices and research focused on ASD. This outcome not only validates the project's approach but also underscores the potential for further refinement and application of ML techniques to ASD diagnosis and treatment.

6.2 Limitation

Despite successfully developing ML models for ASD diagnosis, our project faces several limitations. The reliance on behavioral indicators introduces the potential for misclassifications due to variations unrelated to ASD. Furthermore, variations in feature distribution, despite efforts to balance the dataset, may bias the models and impact overall predictive performance. Additionally, LR's performance, while showing promise, requires further tuning to match the accuracy levels of RF and SVM.

6.3 Practical Usability of Models

The RF and SVM models exhibit strong potential for practical application in clinical settings, owing to their high accuracy and consistency. However, further validation on larger and more diverse datasets is essential to ensure their practical usability and reliability. Ethical considerations, including patient privacy and safety, must be addressed before deploying these models in practice. While LR shows potential, additional tuning and testing are necessary to achieve a performance level suitable for real-world use, indicating its applicability in specific, linear cases.

Overall, our analysis offers a comprehensive view of the models' performance, limitations, and practical implications. RF and SVM emerge as robust options for ASD diagnosis, while LR shows potential with further optimization. These findings provide valuable insights for future research and real-world application in ML-based ASD diagnosis.

7. CONCLUSION

In summary, the project's key findings show that ML models, specifically RF and SVM, are highly effective at diagnosing ASD with notable accuracy, precision, and recall. RF emerged as the most accurate model, followed by SVM, while LR demonstrated promise but required further tuning for improved performance. These results are consistent with existing literature, demonstrating the models' robustness in handling high-dimensional data commonly associated with ASD diagnosis.

These findings suggest that incorporating ML into ASD diagnostic frameworks is a promising approach to improving early detection and intervention. The high performance of RF and SVM models demonstrates their potential utility in practical settings, laying the groundwork for future development of diagnostic tools in healthcare. The implications of this work extend to clinical and research settings where early detection of ASD is critical. These models can help healthcare professionals make more accurate and timely diagnoses, potentially improving outcomes for people with ASD. Furthermore, the study's findings can help to advance ongoing research into the complexities of ASD and the development of more effective treatment approaches.

For future work, I would recommend applying the RF model, which had a balanced and high accuracy, to other ASD-related datasets to assess its generalizability and robustness. This approach would aid in determining whether the model's effectiveness remains consistent across different populations and data sources, thereby validating its practical utility. Furthermore, experimenting with different ML techniques and tuning parameters may result in greater diagnostic accuracy and consistency. These steps would pave the way for further progress in machine learning-based approaches to ASD diagnosis.

9. REFERENCES

Akter, T., Khan, Md.I., Ali, M.H., Satu, Md.S., Uddin, Md.J. and Moni, M.A. (2021). Improved Machine Learning based Classification Model for Early Autism Detection. 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST). doi:https://doi.org/10.1109/icrest51555.2021.9331013.

Chowdhury, K. and Iraj, M.A. (2020). Predicting Autism Spectrum Disorder Using Machine Learning Classifiers. 2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT). doi:https://doi.org/10.1109/rteict49044.2020.9315717.

Hirota, T. and King, B.H. (2023). Autism Spectrum Disorder: A Review. JAMA, [online] 329(2), pp.157–168. doi:https://doi.org/10.1001/jama.2022.23661.

Mahedy Hasan, S.M., Uddin, M.P., Mamun, M.A., Sharif, M.I., Ulhaq, A. and Krishnamoorthy, G. (2023). A Machine Learning Framework for Early-Stage Detection of Autism Spectrum Disorders. IEEE Access, 11, pp.15038–15057. doi:https://doi.org/10.1109/access.2022.3232490.

Raj, S. and Masood, S. (2020). Analysis and Detection of Autism Spectrum Disorder Using Machine Learning Techniques. Procedia Computer Science, 167, pp.994–1004. doi:https://doi.org/10.1016/j.procs.2020.03.399.

R, S., SL, A., Chatterjee, J.M., Alaboudi, A. and Jhanjhi, N. (2021). A Machine Learning Way to Classify Autism Spectrum Disorder. International Journal of Emerging Technologies in Learning (iJET), 16(06), p.182. doi:https://doi.org/10.3991/ijet.v16i06.19559.

Simeoli, R., Milano, N., Rega, A. and Marocco, D. (2021). Using Technology to Identify Children With Autism Through Motor Abnormalities. Frontiers in Psychology, 12. doi:https://doi.org/10.3389/fpsyg.2021.635696.

Vakadkar, K., Purkayastha, D. and Krishnan, D. (2021). Detection of Autism Spectrum Disorder in Children Using Machine Learning Techniques. SN Computer Science, 2(5). doi:https://doi.org/10.1007/s42979-021-00776-5.

www.kaggle.com. (n.d.). ASD children traits. [online] Available at: https://www.kaggle.com/datasets/uppulurimadhuri/dataset [Accessed 1 Apr. 2024].

APPENDIX

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
from sklearn.metrics import precision_recall_curve, classification_report
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV,  cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression



# Ignoring the warnings
import warnings
warnings.filterwarnings("ignore")
data = pd.read_csv('/content/data_csv.csv')
data.head()
data.tail()
data.shape
data.info()
# Identifying numerical columns
numerical_cols = data.select_dtypes(include=['int64', 'float64', 'uint8']).columns
print("Numerical columns:", numerical_cols)

# Identifying categorical columns
categorical_cols = data.select_dtypes(include=['object', 'category', 'bool']).columns
print("Categorical columns:", categorical_cols)
#counting values of each features
for column in data.columns:
    print(f"Value counts for {column}:")
    print(data[column].value_counts())
    print("\n")
#sorting age
age =data.sort_values(by = 'Age_Years')
age['Age_Years']
data.describe()
#plotting ASD_traits

traits_count = data['ASD_traits'].value_counts().sort_index()
```

```
# Creating a new DataFrame for clear labeling
traits_df = pd.DataFrame({
    'Trait': ['Positive', 'Negative'],
    'Counts': [traits_count[1], traits_count[0]]
})

# Plotting the bar chart
plt.figure(figsize=(10, 6))
plt.bar(traits_df['Trait'], traits_df['Counts'], color='purple')

# Setting the title and labels with increased font sizes
plt.title('ASD Result', fontsize=20)
plt.xlabel('Autism Test Result', fontsize=18)
plt.ylabel('Counts', fontsize=18)

# Setting the font size of the ticks on both axes
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)

# Save and show the plot
plt.savefig('ASD_Traits_Result.png')
plt.show()
#bar chart for Gender
sns.countplot(x='ASD_traits', hue='Sex', data=data)
plt.title('Autism count by Gender')
plt.xlabel('Autism Test Result', fontsize=18)
plt.ylabel('Counts', fontsize=18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.savefig("Bar chart for Gender")
plt.show()
# Plot the histogram for the age column
plt.hist(data['Age_Years'], bins=range(data['Age_Years'].min(), data['Age_Years'].max() + 1),
color='blue', edgecolor='black')

# Add titles and labels
plt.title('Age Distribution of Autism Diagnosis', fontsize = 18)
plt.xlabel('Age', fontsize = 16)
plt.ylabel('Frequency', fontsize = 16)

# Show all age values on the x-axis
plt.xticks(range(data['Age_Years'].min(), data['Age_Years'].max() + 1), fontsize= 14)
plt.savefig("histogram for Age")
plt.show()
#selecting columns for pie chart
selected_columns = data[['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9',
'A10_Autism_Spectrum_Quotient']]
```

```
# Determine the layout of the subplots
num_rows = 3
num_cols = 4
plt.figure(figsize=(30,26))

# Set font size for labels and autopct
label_fontsize = 25
autopct_fontsize = 25

# Loop through the columns and create a pie chart in each subplot
for i, column in enumerate(selected_columns):
    ax = plt.subplot(num_rows, num_cols, i+1)
    value_counts = selected_columns[column].value_counts()
    ax.pie(value_counts, labels=value_counts.index, autopct='%1.1f%%', startangle=90,
           textprops={'fontsize': label_fontsize})
    plt.title(column, fontsize=label_fontsize)

plt.subplots_adjust(top=0.93)
plt.suptitle("Pie Chart Distribution of Autism Spectrum Quotient Variables", fontsize=30)
plt.savefig('A1 to A10 Quotient variables')
plt.show()

#Normalize 'Ethnicity' and 'Who_completed_the_test' columns to consistent case
data['Ethnicity'] = data['Ethnicity'].str.lower()
data['Who_completed_the_test'] = data['Who_completed_the_test'].str.lower()

# Map variations of ethnicity names to a standardized name
ethnicity_mapping = {
    'asian': 'asian',
    'south asian': 'south asian',
    'middle eastern': 'middle eastern',
    'black': 'black',
    'mixed': 'mixed'
}
data['Ethnicity'] = data['Ethnicity'].map(ethnicity_mapping).fillna(data['Ethnicity'])

who_mapping = {
    'family member': 'family member'
}
data['Who_completed_the_test']                                                    =
data['Who_completed_the_test'].map(who_mapping).fillna(data['Who_completed_the_test'])

#value counts to see the consolidated categories
print("Value counts for Ethnicity:")
print(data['Ethnicity'].value_counts())
print("\nValue counts for Who_completed_the_test:")
print(data['Who_completed_the_test'].value_counts())
```

```
# Mapping of long column names to shorter ones
short_column_names = {
    "A10_Autism_Spectrum_Quotient": "A10 Quotient",
    "Social_Responsiveness_Scale": "Responsiveness scale",
    "Global developmental delay/intellectual disability": "GDD/Intellect Disability",
    "Social/Behavioural Issues": "Social/Behavior Issues",
    "Speech Delay/Language Disorder": "Language Disorder",
    "Childhood Autism Rating Scale": "Autism rating",
    "Family_mem_with_ASD": "Family_ASD",
    "Who_completed_the_test": "Test_Completed_By"
}

# Renaming the columns
data.rename(columns=short_column_names, inplace=True)
ethnicity_counts = data['Ethnicity'].value_counts()

# Create a dot plot
fig, ax = plt.subplots(figsize=(10, 8))
ethnicities = ethnicity_counts.index[::-1]
counts = ethnicity_counts.values[::-1]

# Define a base color
colors = plt.cm.get_cmap('Set3', len(ethnicities))

# Plotting each row of dots in reversed order
for i, (ethnicity, count) in enumerate(zip(ethnicities, counts)):
    # Generate a color for each ethnicity
    color = colors(i)
    # Create an array with the size of the count for the current ethnicity
    y = np.full(count, i)
    # Plotting
    ax.scatter(range(count), y, color=color, edgecolor='black', s=100, label=ethnicity, alpha=0.6)

# Setting the y-ticks
ax.set_yticks(range(len(ethnicities)))
ax.set_yticklabels(ethnicities, fontsize = 18)
ax.legend(title='Ethnicity', fontsize = 18)

ax.set_title('Dot Plot of Ethnicity Distribution', fontsize = 18)
plt.savefig('dot_plot_ethnicity.png')
plt.show()

# Calculate percentages from the counts
test_counts_percentage = data['Test_Completed_By'].value_counts(normalize=True) * 100

# Create a horizontal bar chart with percentages
plt.figure(figsize=(10, 6))
test_counts_percentage.plot(kind='barh', color='teal',fontsize = 14)
```

```
plt.title('Who Completed the Test (Percentage)', fontsize = 14)
plt.xlabel('Percentage',fontsize = 14)
plt.ylabel('Category', fontsize = 14)
plt.savefig("Who completed the test")
plt.show()
# Creating the DataFrame with missing values count
missing_values_df = pd.DataFrame(data.isnull().sum(), columns=["With Missing Values"])

# Saving the DataFrame to a CSV file
missing_values_df.to_csv("with missing values.csv", index=True)

#handling missing values
selected_columns = ['Responsiveness scale','Qchat_10_Score']
for column in selected_columns:
    if column in data.columns:
        data[column].fillna(data[column].mean(), inplace=True)
data['Social/Behavior Issues'].fillna(data['Social/Behavior Issues'].mode()[0], inplace=True)
data['Depression'].fillna(data['Depression'].mode()[0], inplace=True)
# Creating the DataFrame with missing values count
missing_values_df = pd.DataFrame(data.isnull().sum(), columns=["Without Missing Values"])

# Saving the DataFrame to a CSV file
missing_values_df.to_csv("without missing values.csv", index=True)

#dropping CASE_NO_PATIENT'S
data = data.drop(columns =["CASE_NO_PATIENT'S"])
data.info()
# Count the number of duplicate rows
number_of_duplicates = data.duplicated().sum()

# Print out the number of duplicate rows found
print(f"Number of duplicate rows before dropping duplicates: {number_of_duplicates}")

# Drop the duplicate rows
data = data.drop_duplicates()

# Count the number of duplicate rows to verify that duplicates have been removed
number_of_duplicates_after = data.duplicated().sum()

# Print out the number of duplicate rows after dropping duplicates
print(f"Number of duplicate rows after dropping duplicates: {number_of_duplicates_after}")
#encoding to change categorical variable to numerical variable
binary_columns = ['Language Disorder', 'Learning disorder', 'Genetic_Disorders', 'Depression',
'GDD/Intellect Disability', 'Social/Behavior Issues', 'Anxiety_disorder','Family_ASD',
'Jaundice','ASD_traits']
for column in binary_columns:
    data[column] = data[column].map({'Yes': 1, 'No': 0})
data
```

```
#plotting heatmap to find the feature correlation
plt.figure(figsize=(20, 20))
heatmap = sns.heatmap(data.corr(numeric_only=True), annot=True, cmap='coolwarm',fmt='.2f',
                      vmin = -1, vmax = 1, annot_kws={'size': 14}, linewidths=1, linecolor='black')
plt.title('Heatmap of Variable Correlations', fontsize = '42')

heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=90, horizontalalignment='right',
fontsize='23')
heatmap.set_yticklabels(heatmap.get_yticklabels(), rotation=0, fontsize='23')
plt.savefig("Heatmap")
plt.show()

# Step 2: One-hot encode nominal columns
data = pd.get_dummies(data, columns=['Sex', 'Ethnicity', 'Test_Completed_By'],drop_first=False)
data
df_original = data.copy()
#Function to identify outliers using IQR method
def identify_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data[(data[column] < lower_bound) | (data[column] > upper_bound)]
    return outliers

# Function to handle outliers using IQR method
def handle_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    data[column] = data[column].clip(lower=lower_bound, upper=upper_bound)
    return data

# Identify outliers in numerical columns
numeric_cols = data.select_dtypes(include='number').columns
columns_with_outliers_before = {}
for col in numeric_cols:
    outliers = identify_outliers(data, col)
    if not outliers.empty:
        columns_with_outliers_before[col] = len(outliers)

# Handle outliers
for col in numeric_cols:
    data = handle_outliers(data, col)
```

```
# Check for outliers after handling
columns_with_outliers_after = {}
for col in numeric_cols:
    outliers_after = identify_outliers(data, col)
    if not outliers_after.empty:
        columns_with_outliers_after[col] = len(outliers_after)

print("Columns with outliers before handling and their counts:")
for col, count in columns_with_outliers_before.items():
    print(f"{col}: {count}")

print("\nColumns with outliers after handling and their counts:")
for col, count in columns_with_outliers_after.items():
    print(f"{col}: {count}")

# Create a DataFrame from the 'columns_with_outliers_before' dictionary
outlier_comparison = pd.DataFrame(list(columns_with_outliers_before.items()),
columns=['Column', 'Outliers Before Handling'])

# Add a new column to this DataFrame for the 'columns_with_outliers_after' data
outlier_comparison['Outliers After Handling'] =
outlier_comparison['Column'].map(columns_with_outliers_after)

outlier_comparison['Outliers After Handling'].fillna(0, inplace=True)

outlier_comparison['Outliers After Handling'] = outlier_comparison['Outliers After
Handling'].astype(int)

# Display the pairwise comparison table with a bar representation
outlier_comparison.style.bar(subset=['Outliers Before Handling', 'Outliers After Handling'])
outlier_comparison.to_csv("outlier_comparison.csv", index=False)
outlier_comparison
#setting target and features
Y = data['ASD_traits']
X = data.drop(columns = ['ASD_traits'])
#Separate out a test set
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
# Initialize the scaler
scaler = StandardScaler()

# Fit the scaler on the training set only
scaler.fit(X_train)

# Transform the training, and test sets
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Apply PCA
pca = PCA(n_components=0.95)
pca.fit(X_train_scaled)

# Transform the scaled datasets
X_train_pca = pca.transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
# Initialize SVC with linear kernel
svc = SVC(kernel='linear', C=0.1, random_state=42)

# Apply RFE for feature selection
selector = RFE(estimator=svc, n_features_to_select=15, step=1)
X_train_selected = selector.fit_transform(X_train_pca, y_train)
X_test_selected = selector.transform(X_test_pca)

# Train SVC on selected features
svc.fit(X_train_selected, y_train)

# Predictions and evaluation
y_train_pred = svc.predict(X_train_selected)
y_test_pred = svc.predict(X_test_selected)

#print accuracy
print("Training Accuracy:", accuracy_score(y_train, y_train_pred))
print("Test Accuracy:", accuracy_score(y_test, y_test_pred))
print("\nClassification Report for Test Set:\n", classification_report(y_test, y_test_pred))
cm = confusion_matrix(y_test, y_test_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', cbar=False, annot_kws={"size": 16})
plt.title('Confusion Matrix for the Test Set', fontsize = 16)
plt.xlabel('Predicted Label', fontsize = 16)
plt.ylabel('True Label', fontsize = 16)
plt.savefig("initial confusion matrix of svm")
plt.show()
selected_coefficients = svc.coef_[0]

# generate component names
selected_features = ['Component ' + str(i+1) for i in range(len(selected_coefficients))]

# Plotting the feature importance
plt.figure(figsize=(10, 8))
plt.barh(selected_features, selected_coefficients, color='blue')
plt.xlabel('Coefficient Value')
plt.ylabel('PCA Component')
plt.title('Selected PCA Components for SVM')
plt.gca().invert_yaxis()  # Invert y-axis to have the highest coefficient at the top
plt.savefig("SVM feature selection")
plt.show()
```

```
# Get the PCA components
loadings = pca.components_
feature_names = np.array(X.columns)

# Choose the component
component_index = 0

# Set a threshold for filtering significant loadings only
threshold = 0.2

# Filter features based on the threshold
significant_loadings = loadings[component_index, :]
significant_features = feature_names[np.abs(significant_loadings) >= threshold] #selecting whose absolute values are equal to or exceed the threshold

# Plot the loadings for the significant features of the selected principal component
plt.figure(figsize=(10, 8))
plt.barh(significant_features, significant_loadings[np.abs(significant_loadings) >= threshold], color='blue')
plt.xlabel('Loading Value')
plt.ylabel('Original Feature')
plt.title(f'Feature Contributions to PCA Component 1 SVM')
plt.savefig("Features of component 1 svm")
plt.show()
# Define the parameter grid
param_grid = {
    'C': [0.01, 0.1,1],  # strong to weak regulariztion
    'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],  # Type of SVM kernel
    'gamma': ['scale', 'auto'],  # setting decision boundary for kernel
}
# Proceed with GridSearchCV using the selected features
grid_search = GridSearchCV(estimator=SVC(random_state=42), param_grid=param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=2)
grid_search.fit(X_train_selected, y_train)

# Display the best parameters
print("Best parameters found: ", grid_search.best_params_)
# Use the best estimator to make predictions
best_model = grid_search.best_estimator_

y_train_pred_best = best_model.predict(X_train_selected)
y_test_pred_best = best_model.predict(X_test_selected)

# Calculate accuracies with the best model
train_accuracy_best = accuracy_score(y_train, y_train_pred_best)
test_accuracy_best = accuracy_score(y_test, y_test_pred_best)

# Generate and print the classification report for the test set
```

```
print("Classification Report (Test Set):")
print(classification_report(y_test, y_test_pred_best))

# Print accuracies after Grid Search
print(f"Training Accuracy: {train_accuracy_best}")
print(f"Test Accuracy: {test_accuracy_best}")

cm_test = confusion_matrix(y_test, y_test_pred_best)

# Visualize the confusion matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(cm_test, annot=True, fmt="d", cmap='Blues', cbar=False,
        annot_kws={"size": 20})  # Adjusting size for better readability

plt.title('Confusion Matrix of SVM', size=20)
plt.xlabel('Predicted Label', size=24)
plt.ylabel('True Label', size=24)
plt.xticks(size=20)
plt.yticks(size=20, rotation=0)
plt.savefig("Confusion matrix of final svm")
plt.show()

cv_scores = cross_val_score(best_model, X_train_selected, y_train, cv=5, scoring='accuracy')

# Calculate the mean and standard deviation of the cross-validation scores
cv_mean = np.mean(cv_scores)
cv_std = np.std(cv_scores)

# Print the results
print(f"Cross-Validation Accuracy: {cv_mean:.4f} (+/- {cv_std:.4f})")
train_scores = best_model.decision_function(X_train_selected)
test_scores = best_model.decision_function(X_test_selected)

# Calculate precision and recall for both sets
precision_train, recall_train, thresholds_train = precision_recall_curve(y_train, train_scores)
precision_test, recall_test, thresholds_test = precision_recall_curve(y_test, test_scores)

# plot the precision-recall curves for both
plt.figure(figsize=(10, 6))

# Plot training set curve
plt.plot(recall_train, precision_train, color='green', lw=2,  marker='.', label='Train Precision-Recall curve')

# Plot test set curve
plt.plot(recall_test, precision_test, color='blue', lw=2, marker='.', label='Test Precision-Recall curve')
```

```
plt.xlabel('Recall', fontsize=14)
plt.ylabel('Precision', fontsize=14)
plt.title('Precision-Recall Curve for SVM', fontsize=16)
plt.legend(loc="lower left", fontsize=12)
plt.savefig("Precision_recall_of_svm.png", dpi=300, bbox_inches='tight')
plt.show()
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(X_train_pca, y_train)

# Calculate and print accuracy using predictions on the training set
y_train_pred = random_forest.predict(X_train_pca)
train_accuracy = accuracy_score(y_train, y_train_pred)
print(f"Training Accuracy: {train_accuracy:.4f}")

# Calculate and print accuracy using predictions on the test set
y_test_pred = random_forest.predict(X_test_pca)
test_accuracy = accuracy_score(y_test, y_test_pred)
print(f"Test Accuracy: {test_accuracy:.4f}")

# Print the classification report for the test set predictions
print("\nClassification Report for Test Set:")
print(classification_report(y_test, y_test_pred))
# Confusion Matrix for initial model
cm_initial = confusion_matrix(y_test, y_test_pred)
plt.figure(figsize=(7, 5))
sns.heatmap(cm_initial, annot=True, fmt='d', cmap='Greens',cbar = False, xticklabels=True,
yticklabels=True, annot_kws={"size": 12})
plt.title('Confusion Matrix - Initial Model', fontsize = 16)
plt.xlabel('Predicted Labels', fontsize = 16)
plt.ylabel('True Labels', fontsize = 16)
plt.savefig("Confusion matrix of initial RF")
plt.show()
importances = random_forest.feature_importances_
sorted_indices = importances.argsort()[::-1][:15]  # Get indices of top 15 features

# Generate component names
component_names = ['Component ' + str(i+1) for i in range(X_train_pca.shape[1])]

plt.figure(figsize=(10, 8))
plt.title('Top 15 Feature Importances Initial Random Forest Model')
plt.barh(range(15), importances[sorted_indices], align='center', color = 'green')
plt.yticks(range(15), [component_names[i] for i in sorted_indices])
plt.xlabel('Relative Importance')
plt.gca().invert_yaxis()
plt.savefig("Top component of RF")
plt.show()
# Get the loadings
```

```
loadings = pca.components_

component_number = 0  # for the first principal component

# Set a threshold for selecting significant features based on their loading scores
threshold = 0.02

# Get the loading scores for this component
loading_scores = pd.Series(loadings[component_number], index=X.columns)

# Select features that meet or exceed the threshold in absolute value
significant_features = loading_scores[loading_scores.abs() >= threshold]

# Sorting the significant features by their loading score magnitude
sorted_significant_features = significant_features.abs().sort_values(ascending=False)

# Plotting
plt.figure(figsize=(10, 6))
sorted_significant_features.plot(kind='barh', color='lightgreen')
plt.title(f' Features Contribution to Principal Component {component_number + 1} (Threshold:
{threshold})')
plt.xlabel('Loading Score')
plt.ylabel('Features')
plt.gca().invert_yaxis()  # Invert y-axis to have the highest value at the top
plt.savefig("Feature of component 1 of RF")
plt.show()
X_train_top15 = X_train_pca[:, sorted_indices]
X_test_top15 = X_test_pca[:, sorted_indices]
#Hyperparameter tuning with only the top 15 features
param_grid = {
   'n_estimators': [50, 75, 100],  # Reduced estimators to control complexity
   'max_depth': [5,10, 15, 20],  # Limited tree depth to avoid overfitting
   'min_samples_split': [5, 10, 15, 20],  # Increased minimum samples to split
   'min_samples_leaf': [10,15],  # Added minimum samples at leaf to regularize
   'max_features': ['sqrt', 'log2'],  # Control the number of features considered at each split
}

grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42),
                           param_grid=param_grid, cv=10, scoring='accuracy', n_jobs=-1, verbose=2)
grid_search.fit(X_train_top15, y_train)

print("Best parameters found:", grid_search.best_params_)
best_rf = RandomForestClassifier(**grid_search.best_params_, random_state=42)
best_rf.fit(X_train_top15, y_train)
# Predict with top 15 features on the training set
y_train_pred_final = best_rf.predict(X_train_top15)
# Predict with top 15 features
y_pred_final = best_rf.predict(X_test_top15)
```

```
# Calculate accuracy and print the classification report
accuracy_final = accuracy_score(y_test, y_pred_final)
accuracy_train_final = accuracy_score(y_train, y_train_pred_final)
print(f"Final Training Set Accuracy: {accuracy_train_final:.4f}")
print(f"Final Test Set Accuracy: {accuracy_final:.4f}")
print(classification_report(y_test, y_pred_final))

# Confusion Matrix for the final model
cm_final = confusion_matrix(y_test, y_pred_final)
plt.figure(figsize=(7, 5))
sns.heatmap(cm_final, annot=True, fmt='d', cmap='Greens',cbar = False, xticklabels=True, yticklabels=True, annot_kws={"size": 16})
plt.title('Confusion Matrix - Final Model', fontsize = 16)
plt.xlabel('Predicted Labels', fontsize = 16)
plt.ylabel('True Labels', fontsize = 16)
plt.savefig("Confusion matrix for RF")
plt.show()
# Perform 5-fold cross-validation and obtain the cross-validation scores
cv_scores = cross_val_score(random_forest, X_train_pca, y_train, cv=5, scoring='accuracy')

# Calculate the mean and standard deviation of the cross-validation scores
cv_scores_mean = cv_scores.mean()
cv_scores_std = cv_scores.std()

print(f"CV Accuracy: {cv_scores_mean:.4f} (+/- {cv_scores_std * 2:.4f})")
cv_scores_std
y_scores_train_best = best_rf.predict_proba(X_train_top15)[:, 1]  # Probabilities for the positive class on the training set
y_scores_test_best = best_rf.predict_proba(X_test_top15)[:, 1]  # Probabilities for the positive class on the test set

# Step 2: Calculate precision and recall for both sets
precision_train, recall_train, _ = precision_recall_curve(y_train, y_scores_train_best)
precision_test, recall_test, _ = precision_recall_curve(y_test, y_scores_test_best)

# Step 3: Plotting the curves
plt.figure(figsize=(8, 6))
plt.plot(recall_train, precision_train, marker='.', color='red', label='Train Precision-Recall Curve')
plt.plot(recall_test, precision_test, marker='.', color='green', label='Test Precision-Recall Curve')
plt.xlabel('Recall', fontsize=14)
plt.ylabel('Precision', fontsize=14)
plt.title('Precision-Recall Curve for RF', fontsize=16)
plt.legend(loc="lower left", fontsize=12)
plt.savefig("Precision_recall_of_RF.png", dpi=300, bbox_inches='tight')
plt.show()
# Initialize the Logistic Regression model for RFE
log_reg_for_rfe = LogisticRegression(random_state=42, max_iter=10000)
```

```
# Apply RFE for feature selection, aiming to select top 15 features
rfe = RFE(estimator=log_reg_for_rfe, n_features_to_select=15, step=1)
rfe.fit(X_train_pca, y_train)

# Select the features from training and test set based on RFE selection
X_train_selected = X_train_pca[:, rfe.support_]
X_test_selected = X_test_pca[:, rfe.support_]
# Train a new Logistic Regression model on the RFE-selected features
log_reg_selected = LogisticRegression(random_state=42, max_iter=10000)
log_reg_selected.fit(X_train_selected, y_train)

# Predict on the training set with the new model
y_train_pred = log_reg_selected.predict(X_train_selected)

# Predict on the test set with the new model
y_test_pred = log_reg_selected.predict(X_test_selected)

# Calculate and print the training accuracy
train_accuracy = accuracy_score(y_train, y_train_pred)
print(f"Training Set Accuracy: {train_accuracy:.4f}")

# Calculate and print the test accuracy
test_accuracy = accuracy_score(y_test, y_test_pred)
print(f"Test Set Accuracy: {test_accuracy:.4f}")

# Print the classification report for the test set predictions
print("\nClassification Report for Test Set:")
print(classification_report(y_test, y_test_pred))
conf_matrix = confusion_matrix(y_test, y_test_pred)

# Plot the confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Reds", cbar=False, annot_kws={"size":
16})
plt.xlabel('Predicted Label', fontsize = 16)
plt.ylabel('True Label', fontsize = 16)
plt.title('Confusion Matrix for Logistic Regression - Initial Model')
plt.savefig("Inital Confusion metrices of lr")
plt.show()
coef = log_reg_selected.coef_[0]

# Create a range for the x-axis
feature_range = np.arange(len(coef)) + 1

# Plotting
plt.figure(figsize=(8, 10))
plt.barh(feature_range, coef, color='red')
```

```
plt.ylabel('Selected PCA Components')
plt.xlabel('Coefficient Value')
plt.title('Importance of Selected PCA Components')
plt.yticks(feature_range)
plt.gca().invert_yaxis()
plt.savefig("Top features of LR")
plt.show()
loadings = pca.components_[0]  # Loadings for the first component

# Threshold for filtering significant loadings only
threshold = 0.2

# Filter features and their loadings based on the threshold
significant_loadingslr = loadings[np.abs(loadings) > threshold]
significant_featureslr = np.array(feature_names)[np.abs(loadings) > threshold]

# Creating a range for the y-axis for the filtered features
feature_range = np.arange(len(significant_loadingslr)) + 1

# Plotting
plt.figure(figsize=(8, 10))
plt.barh(feature_range, significant_loadingslr, color='Red')
plt.ylabel('Features')
plt.xlabel('Loading Value')
plt.title('Feature Contributions to component 1 (threshold:0.02)')
plt.yticks(feature_range, significant_featureslr)
plt.gca().invert_yaxis()  # Invert y-axis
plt.savefig(" Feature os component 1 LR")
plt.show()
param_grid = {
    'C': np.logspace(-8, 8, 100),
    'solver': ['liblinear', 'lbfgs'],
        'max_iter': [500, 1000,5000, 10000]
}
grid_search = GridSearchCV(estimator=LogisticRegression(random_state=42),
param_grid=param_grid, cv=15, scoring='accuracy', n_jobs=-1, verbose=2)
grid_search.fit(X_train_selected, y_train)
print("Best parameters found:", grid_search.best_params_)
# Train new Logistic Regression model with best parameters on RFE-selected features
best_log_reg = LogisticRegression(**grid_search.best_params_, random_state=42)
best_log_reg.fit(X_train_selected, y_train)

# Predict on the training set with the best model
y_train_pred_best = best_log_reg.predict(X_train_selected)

# Predict on the test set with the best model
y_test_pred_best = best_log_reg.predict(X_test_selected)
# Calculate and print the final training accuracy
```

```
final_train_accuracy = accuracy_score(y_train, y_train_pred_best)
print(f"Final Training Set Accuracy: {final_train_accuracy:.4f}")

# Calculate and print the final test accuracy
final_test_accuracy = accuracy_score(y_test, y_test_pred_best)
print(f"Final Test Set Accuracy: {final_test_accuracy:.4f}")

# Print the classification report for the test set predictions
print("\nFinal Classification Report for Test Set:")
print(classification_report(y_test, y_test_pred_best))
# Confusion Matrix for the final model
cm_final = confusion_matrix(y_test, y_test_pred_best)
plt.figure(figsize=(7, 5))
sns.heatmap(cm_final, annot=True, fmt='d', cmap='Reds',cbar = False,  xticklabels=True,
        yticklabels=True)
plt.title('Confusion Matrix - Final Logistic Regression Model')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.savefig("Confusion matrix LR")
plt.show()
# Perform cross-validation with 10-fold cross-validation
cv_scores = cross_val_score(best_log_reg, X_train_selected, y_train, cv=10, scoring='accuracy',
n_jobs=-1)

# Calculate the mean and standard deviation of the cross-validation scores
cv_mean = cv_scores.mean()
cv_std = cv_scores.std()

print(f"Cross-Validation Accuracy: {cv_mean:.4f} (+/- {cv_std * 2:.4f})")
y_scores_lr = best_log_reg.decision_function(X_test_selected)
precision, recall, thresholds = precision_recall_curve(y_test, y_scores_lr)

y_scores_train = best_log_reg.decision_function(X_train_selected)
# Calculate precision and recall for the training set
precision_train, recall_train, thresholds_train = precision_recall_curve(y_train, y_scores_train)

# Plotting both Training and Test Precision-Recall Curves
plt.figure(figsize=(10, 8))

# Plot for the test set
plt.plot(recall, precision, marker='.', color='maroon', label='Test Set - Logistic Regression')

# Plot for the training set
plt.plot(recall_train, precision_train, marker='.', color='purple', label='Training Set - Logistic
Regression')

plt.xlabel('Recall', fontsize=14)
plt.ylabel('Precision', fontsize=14)
```

```
plt.title('Precision-Recall Curve for LR', fontsize=16)
plt.legend(loc="lower left", fontsize=12)
plt.savefig("Precision_recall_of_LR.png", dpi=300, bbox_inches='tight')
plt.show()
```