VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI-590018, KARNATAKA.

A PROJECT REPORT

On

“AUTISM SPECTRUM DISORDER PREDICTION USING MACHINE LEARNING”

Submitted in Partial Fulfillment for the Award of the Degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Submitted By:

1EE20CS001	AKSHAY KUMAR P
1EE20CS005	BASAVARAJ ANGADI
1EE20CS011	DARSHAN GOWDA C
1EE20CS014	HARISH S

Under the Guidance of

POOJA M V

Assistant Professor

Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

EAST WEST COLLEGE OF ENGINEERING

Bengaluru - 560064

2023 - 2024

EAST WEST COLLEGE OF ENGINEERING

No.13, Major Akshay Girish Kumar Road, Sector A Yelahanka New Town,

Bangalore-64 (Affiliated to Visvesvaraya Technological University Belagavi)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Certificate

Certified that the Project Work entitled **"AUTISM SPECTRUM DISORDER PREDICTION USING MACHINE LEARNING"** carried out by **AKSHAY KUMAR P (1EE20CS001), BASAVARAJ ANGADI (1EE20CS005), DARSHAN GOWDA C (1EE20CS011), HARISH S (1EE20CS014),** bonafide students of **East West College of Engineering**, in partial fulfillment for the award of **Bachelor of Engineering** in **Computer Science and Engineering** of **Visvesvaraya Technological University**, **Belagavi** during the academic year 2023-2024. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the Department Library. The project report has been approved as it satisfies the academic requirements in respect of **Project Work Phase II (18CSP83)** prescribed for the said degree.

Signature of the Guide	**Signature of the HOD**	**Signature of the Principal**
Pooja M V	**Dr. Lavanya N L**	**Dr. Santosh Kumar G**
Assistant Professor	**Professor & Head**	**Principal**

EXTERNAL EXAMINATION:

Name of the Examiners	**Signature with Date**
1.______________________________	______________________________
2.______________________________	______________________________

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the completion of any task would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement ground my efforts with success.

We consider it as a privilege to express our gratitude and respect to all those who guided us in completion of project work.

We are, grateful to thank our Principal **Dr. Santhosh Kumar G**, East West College of Engineering, who patronized throughout our career & for the facilities provided to carry out this work successfully.

It's a great privilege to place on record our deep sense of gratitude to our beloved HOD **Dr. Lavanya N L** of Computer Science & Engineering, who patronized throughout our career & for the facilities provided to carry out this work successfully.

We are also grateful to thank our project Guide **Prof. Pooja M V,** Assistant Professor of CSE department for her invaluable support and guidance.

We would also like to thank the teaching and non-teaching staff members who have helped us directly or indirectly during the project.

Lastly but most importantly we would like to thank our family and friends for their co- operation and motivation to complete this project successfully.

AKSHAY KUMAR P	**1EE20CS001**
BASAVARAJ ANGADI	**1EE20CS005**
DARSHAN GOWDA C	**1EE20CS011**
HARISH S	**1EE20CS014**

ABSTRACT

A neuro-disease known as autism spectrum disorder (ASD) affects a person's ability to engage and communicate with others on a lifelong basis. Autism is referred to as a "behavioural disorder" since symptoms typically develop in the first two years of life, but it can be diagnosed at any point in one's life. According to the ASD theory, problems begin in childhood and persist into adolescence and maturity. With the increased use of machine learning-based models for illness prediction, it is now possible to identify diseases early based on a variety of physiological and health parameters. This element encouraged us to become more interested in the identification and examination of ASD disorders in order to develop more effective treatment approaches. The project attempts to explore the potential use of Nave Bayes, Support Vector Machine, Logistic Regression, KNN, Neural Network for predicting and analyzing the ASD problems in a child, adolescent, and adult populations, driven by the rise in the use of machine learning techniques in the research dimensions of medical diagnosis. We make a web interface so that one can utilize the trained ML model effectively. We also created a image recognition deep learning model using VGG16 to classify the patient into Autistic or Non-Autistic.

EAST WEST COLLEGE OF ENGINEERING

No.13, Major Akshay Girish Kumar Road, Sector A Yelahanka New Town, Bangalore-64
(Affiliated to Visvesvaraya Technological University Belagavi)

Department of Computer Science and Engineering

DECLARATION

We **AKSHAY KUMAR P (1EE20CS001), BASAVARAJ ANGADI (1EE20CS005), DARSHAN GOWDA C (1EE20CS011), and HARISH S (1EE20CS0014)** bonafide students of East West College of Engineering, hereby declare that the project entitled **"AUTISM SPECTRUM DISORDER PREDICTION USING MACHINE LEARNING"** submitted in partial fulfilment for the award of Bachelor of Engineering in Computer Science & Engineering of the Visvesvaraya Technological University, Belgaum during the year 2023-2024, is our original work and the project has not formed the basis for the award of any other degree, fellowship or any other similar titles.

Name & Signature of the Student with date.

1) AKSHAY KUMAR P (1EE20CS001)
2) BASAVARAJ ANGADI (1EE20CS005)
3) DARSHAN GOWDA C (1EE20CS011)
4) HARISH S (1EE20CS014)

ABBREVIATIONS

ASD: Autism Spectrum Disorder

SVM: Support Vector Machine

VGG16: Visual Geometry Group 16

ASM: Attribute Selection Measure

SRS: Software Requirements Specification

IDLE: Integrated Development and Learning Environment

GPU: General Processing Unit

I/O: Input or Output

API: Application Programming Interface

DFD: Data Flow Diagram

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

CHAPTER 1

INTRODUCTION

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental condition characterized by challenges in social interaction, communication, and behavior. It encompasses a wide range of symptoms and severity levels, which is why it's referred to as a "spectrum" disorder. Individuals with ASD may struggle with understanding social cues, maintaining eye contact, and engaging in typical social interactions. ASD begins before the age of 3 years and can last throughout a person’s life, although symptoms may improve over time. Some children show ASD symptoms within the first 12 months of life. In others, symptoms may not show up until 24 months of age or later. Some children with ASD gain new skills and meet developmental milestones until around 18 to 24 months of age, and then they stop gaining new skills or lose the skills they once had. As children with ASD become adolescents and young adults, they may have difficulties developing and maintaining friendships, communicating with peers and adults, or understanding what behaviors are expected in school or on the job. They may come to the attention of healthcare providers because they also have conditions such as anxiety, depression, or attention-deficit/hyperactivity disorder, which occur more often in people with ASD than in people without ASD. Communication difficulties can manifest as delayed speech development, difficulty with nonverbal communication, and a tendency towards literal interpretation of language. Additionally, individuals with ASD may exhibit repetitive behaviors, restricted interests, and sensitivity to sensory stimuli. Diagnosis typically involves a comprehensive evaluation by healthcare professionals, often including psychologists, developmental pediatricians, or psychiatrists. Early intervention and tailored support services, such as behavioral therapy and speech therapy, play a crucial role in helping individuals with ASD develop essential skills and lead fulfilling lives. It's essential to recognize the diversity of abilities and strengths within the ASD community and to foster understanding and acceptance to support their unique needs.

A child or adult with autism spectrum disorder may have problems with social interaction and communication skills, including any of these signs:

- Fails to respond to his or her name or appears not to hear you at times.
- Resists cuddling and holding, and seems to prefer playing alone, retreating into his

or her own world.

- Has poor eye contact and lacks facial expression.
- Doesn't speak or has delayed speech, or loses previous ability to say words or sentences.
- Can't start a conversation or keep one going, or only starts one to make requests or label items.
- Speaks with an abnormal tone or rhythm and may use a singsong voice or robot-like speech.
- Repeats words or phrases verbatim, but doesn't understand how to use them.
- Doesn't appear to understand simple questions or directions.
- Doesn't express emotions or feelings and appears unaware of others' feelings.
- Doesn't point at or bring objects to share interest.
- Inappropriately approaches a social interaction by being passive, aggressive or disruptive.
- Has difficulty recognizing nonverbal cues, such as interpreting other people's facial expressions, body postures or tone of voice.

1.1 Different Types of Autism Spectrum Disorder

Here are some terms often used to describe different types or subtypes of ASD:

- **Classic Autism:** This term is often used to refer to individuals with more severe symptoms of ASD, including significant challenges in social interaction, communication, and behavior.
- **Asperger's Syndrome:** Historically considered a separate diagnosis, Asperger's Syndrome is characterized by milder symptoms of ASD, particularly in the areas of language development and cognitive abilities. Individuals with Asperger's may have average or above-average intelligence and typically do not have delays in language development.
- **Pervasive Developmental Disorder-Not Otherwise Specified (PDD-NOS):** This term was previously used to describe individuals who showed some symptoms of ASD but did not meet the criteria for classic autism or Asperger's Syndrome. With the DSM-5, PDD-NOS is no longer used as a separate diagnosis, as all these conditions are now included under the umbrella term of ASD.
- **High-Functioning Autism (HFA):** This term is often used to describe individuals

with ASD who have average or above-average intelligence and relatively mild symptoms, particularly in terms of language development. It's similar to Asperger's Syndrome but doesn't necessarily imply the absence of language delays in early childhood.

- **Regressive Autism:** Some children with ASD appear to develop typically for a period and then experience a loss of previously acquired skills, such as language or social interaction. This phenomenon is known as regressive autism.
- **Non-Verbal Autism:** This term is used to describe individuals with ASD who have limited or absent speech development, relying on alternative forms of communication such as gestures, pictures, or assistive communication devices.

1.2 Causes of ASD

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental disorder with no single known cause. Instead, it is believed to result from a combination of genetic, environmental, and developmental factors. Here are some of the factors that are thought to contribute to the development of ASD:

- **Genetic Factors:** Research suggests that genetics plays a significant role in the development of ASD. Studies of twins have shown that identical twins are more likely to both have ASD compared to fraternal twins, indicating a strong genetic component. However, no single gene has been identified as the sole cause of ASD. Instead, it is believed that multiple genes, along with possible gene-environment interactions, contribute to the risk of developing ASD.
- **Environmental Factors:** Various environmental factors have been studied as potential contributors to ASD, although their role is not fully understood. These factors may include prenatal exposure to certain toxins, such as air pollution or maternal smoking, as well as maternal infections during pregnancy. However, the specific environmental factors and their mechanisms of influence are still under investigation.
- **Neurological and Developmental Factors:** ASD is characterized by differences in brain structure and function, although the exact nature of these differences is still being studied. Abnormalities in brain development, connectivity, and synaptic function have been observed in individuals with ASD. These neurological differences may contribute to the social, communication, and behavioral challenges associated with ASD.

- **Prenatal and Perinatal Factors:** Certain prenatal and perinatal factors have been associated with an increased risk of ASD, although the causal relationship is not fully understood. These factors may include maternal health conditions during pregnancy, such as gestational diabetes or hypertension, as well as complications during childbirth, such as prematurity or low birth weight.
- **Immune System Dysfunction:** Some research suggests that abnormalities in the immune system may play a role in the development of ASD. Immune system dysregulation, including inflammation and altered immune responses, has been observed in individuals with ASD. However, the exact relationship between immune system dysfunction and ASD is still being investigated.

1.3 Machine Learning

In general context, machine learning can be defined as a field in artificial intelligence that provides the system the capability to learn from the experience automatically without the human intervention and aims to predict the future outcomes as accurate as possible utilizing various algorithmic models. Machine Learning is very different than the conventional computation approaches, where systems are explicitly programmed to calculate or solve a problem. Machine learning deals with the input data that are used to train a model where the model learns different patterns in the input data and uses that knowledge to predict unknown results. The application of machine learning is incredibly vast. It is used in various applications like the spam filter, weather prediction, stock market prediction, medical diagnosis, fraud detection, autopilot, house price prediction, face detection, and many more.

1.3.1 Machine Learning Algorithms

1.3.1.1 Supervised Machine Learning

A support vector machine is a supervised learning model that divides the data into regions separated by a linear boundary. Here, the linear boundary divides the black circles from the white. Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data and consists of a set of training examples. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix.

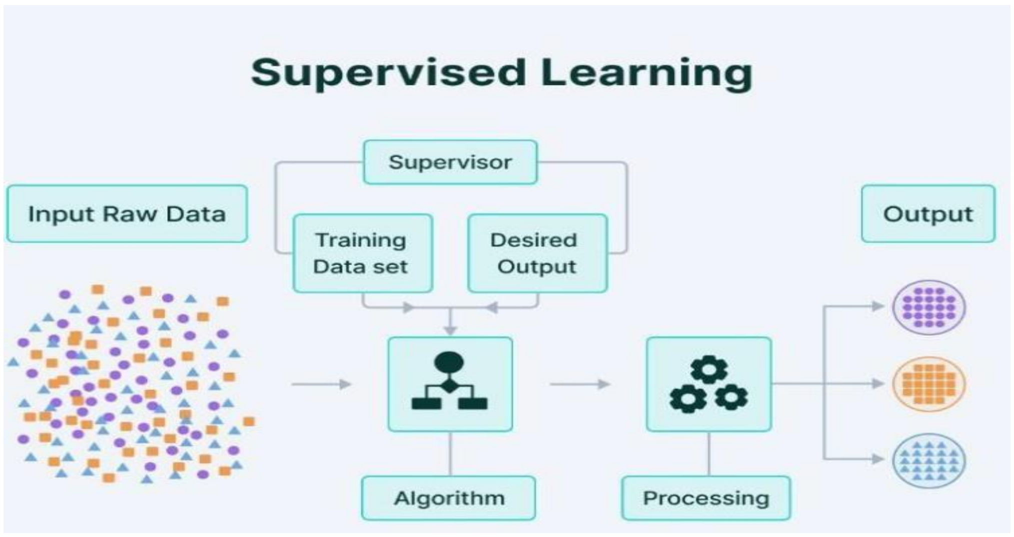


Figure 1.3: Supervised machine learning model

- **Random Forest:**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

How does Random Forest algorithm work?
Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.
The Working process can be explained in the below steps and diagram:

- **Step-1**: Select random K data points from the training set.
- **Step-2**: Build the decision trees associated with the selected data points (Subsets).
- **Step-3**: Choose the number N for decision trees that you want to build.
- **Step-4**: Repeat Step 1 & 2.
- **Step-5**: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

➢ **Logistic Regression:**

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems. In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc. Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.
- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

- **Decision Tree:**

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree- structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed based on features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, like a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

How does the Decision Tree algorithm work?

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:

- **Step-1**: Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2**: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- **Step-3**: Divide the S into subsets that contains possible values for the best attributes.
- **Step-4**: Generate the decision tree node, which contains the best attribute.
- **Step-5**: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

- **ADA Boost:**

ADA Boost is one of the first boosting algorithms to have been introduced. It is mainly used for classification, and the base learner (the machine learning algorithm that is boosted) is usually a decision tree with only one level, also called as stumps. It makes use of weighted errors to build a strong classifier from a series of weak classifiers. Adaptive Boosting which is a Machine Learning technique used as an Ensemble Method. The most widely used algorithm with Ada Boost is decision trees with one level.

How does the ADA Boost algorithm work?

- **Step 1**: Assign Equal Weights to all the observations
- **Step 2**: Classify random samples using stumps
- **Step 3**: Calculate Total Error
- **Step 4**: Calculate Performance of the Stump
- **Step 5**: Update Weights
- **Step 6**: Update weights in iteration
- **Step 7**: Final Predictions

- **Support Vector Machine:**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.

How does the Support Vector Machine algorithm work?

- **Step 1**: SVM algorithm predicts the classes. One of the classes is identified as 1 while the other is identified as -1.
- **Step 2**: As all machine learning algorithms convert the business problem into a mathematical equation involving unknowns. These unknowns are then found by converting the problem into an optimization problem.
- **Step 3**: For ease of understanding, this loss function can also be called a cost function whose cost is 0 when no class is incorrectly predicted.
- **Step 4**: As is the case with most optimization problems, weights are optimized by calculating the gradients using advanced mathematical concepts of calculus viz. partial derivatives.
- **Step 5**: The gradients are updated only by using the regularization parameter when there is no error in the classification while the loss function is also used when misclassification happens.
- **Step 6**: The gradients are updated only by using the regularization parameter when there is no error in the classification, while the loss function is also used when misclassification happens.

1.3.1.2 Unsupervised Learning

Unsupervised learning refers to the use of artificial intelligence (AI) algorithms to identify patterns in data sets containing data points that are neither classified nor labeled. The algorithms are thus allowed to classify, label, and/or group the data points contained within the data sets without having any external guidance in performing that task. In other words, unsupervised learning allows the system to identify patterns within data sets on its own. In unsupervised learning, an AI system will group unsorted information according to similarities and differences even though there are no categories provided. Unsupervised learning algorithms can perform more complex processing tasks than supervised learning systems. Additionally, subjecting a system to unsupervised learning is one way of testing AI. However, unsupervised learning can be more unpredictable than a supervised learning model. While an unsupervised learning AI system might, for example, figure out on its own how to sort cats from dogs, it might also add unforeseen and undesired categories to deal with unusual breeds, creating clutter instead of order. AI systems capable of unsupervised learning are often associated with generative learning models, although they may also use a retrieval-based approach. Chatbots, self-driving cars, facial recognition programs, expert systems, and robots are among the systems that may use either supervised or unsupervised learning approaches or both.

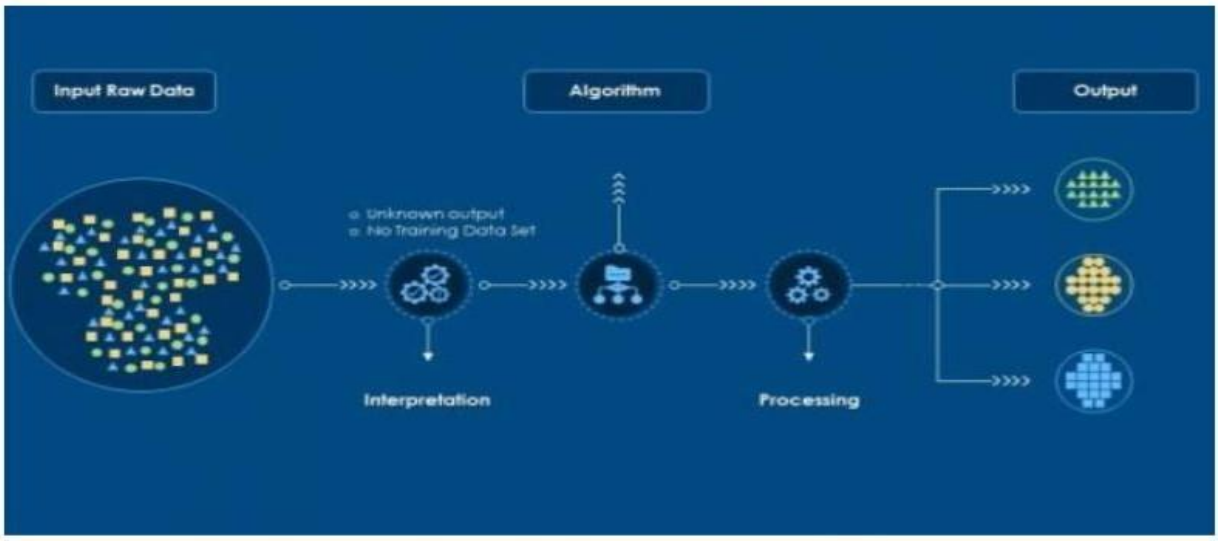


Figure 1.4: Unsupervised machine learning Model

Unsupervised machine learning algorithms play a crucial role in analyzing and interpreting large datasets where the output or target variable is not known. Unlike supervised learning, where the algorithm is trained on labeled data, unsupervised learning algorithms work with data that has no predefined labels. The primary goal is to infer the natural structure present within a set of data points.

- **Clustering Algorithms**

  Clustering is one of the most common tasks in unsupervised learning. Clustering algorithms aim to partition the dataset into distinct groups (clusters) based on the similarity of the data points. Some well-known clustering algorithms include:

  - K-means Clustering: This algorithm partitions the dataset into K clusters, where each data point belongs to the cluster with the nearest mean. It iteratively assigns data points to clusters and updates the cluster centroids until convergence.
  - Hierarchical Clustering: This method builds a tree-like structure (dendrogram) of nested clusters. It can be agglomerative (bottom-up approach, where each data point starts as its own cluster and merges with others) or divisive (top-down approach, where all data points start in one cluster and split recursively).
  - DBSCAN (Density-Based Spatial Clustering of Applications with Noise): DBSCAN identifies clusters based on the density of data points. It is capable of discovering clusters of arbitrary shape and can handle noise and outliers effectively.

- **Dimensionality Reduction Algorithms**

  Dimensionality reduction techniques are used to reduce the number of variables in a dataset while retaining as much information as possible. This is useful for visualizing high-dimensional data, reducing computational cost, and mitigating the curse of dimensionality. Common dimensionality reduction algorithms include:

  - Principal Component Analysis (PCA): PCA transforms the data into a new coordinate system, where the first few coordinates (principal components) capture most of the variance in the data. It identifies the directions (components) that maximize the variance and projects the data onto these components.
  - t-Distributed Stochastic Neighbor Embedding (t-SNE): t-SNE is a nonlinear dimensionality reduction technique particularly well-suited for visualizing high-dimensional data. It converts the similarities between data points into joint probabilities and minimizes the Kullback-Leibler divergence between these joint probabilities in the low-dimensional space.
  - Autoencoders: These are neural network-based models used for dimensionality reduction. An autoencoder consists of an encoder that compresses the input data

into a lower-dimensional representation and a decoder that reconstructs the input data from this representation. The model is trained to minimize the reconstruction error.

1.3.1.3 Reinforcement Learning

Reinforcement learning is an exciting area of machine learning that involves teaching agents to make decisions and take actions based on feedback in the form of rewards or penalties. Unlike other machine learning approaches, which require labeled datasets or explicit instructions, RL enables the agent to learn from experience, trial and error, and self-correction.

Reinforcement learning (RL) is a branch of machine learning concerned with training agents to make sequential decisions in dynamic environments with the goal of maximizing cumulative rewards. Unlike supervised learning, where models learn from labeled data, or unsupervised learning, which discovers patterns without explicit guidance, RL operates in an interactive setting where agents learn through trial and error. An RL setup typically involves an agent interacting with an environment, taking actions based on its current state, and receiving feedback in the form of rewards or penalties. The agent's objective is to learn a policy—a mapping from states to actions—that maximizes the cumulative reward over time. Key components include the agent, environment, state, action, and reward. RL algorithms can be categorized as model-free or model-based, depending on whether they learn directly from experience or use an internal model of the environment. Applications of RL span diverse domains such as robotics, gaming, finance, and healthcare, with recent advancements in deep reinforcement learning leading to breakthroughs in complex tasks. However, RL also poses challenges such as sample inefficiency and exploration-exploitation trade-offs, driving ongoing research efforts to address these issues and unlock the full potential of reinforcement learning. Recent advancements in deep reinforcement learning, which combines deep learning with RL, have led to breakthroughs in complex tasks such as game playing (e.g., AlphaGo), robotic control, and natural language processing. However, RL also poses challenges such as sample inefficiency, exploration-exploitation trade-offs, and instability during training, which researchers continue to address through ongoing research and innovation. Despite these challenges, reinforcement learning holds great promise for developing autonomous systems capable of learning and adapting to diverse and dynamic environments.

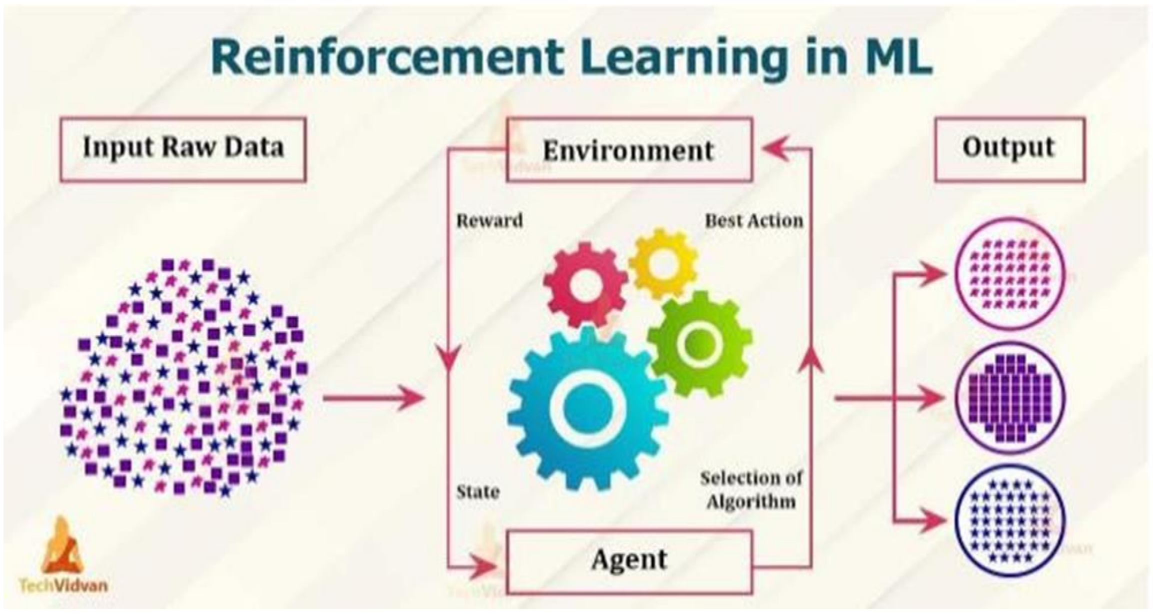


Figure 1.5: Reinforcement machine learning model

1.4 Deep Learning

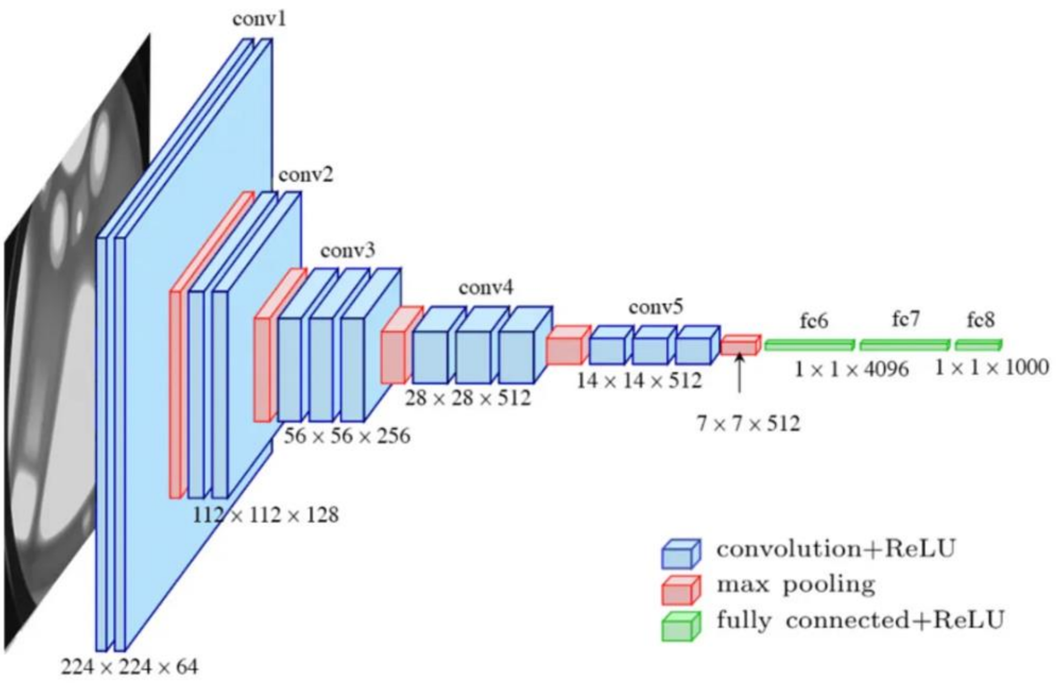
Deep learning is a subset of machine learning that focuses on neural networks with many layers, enabling the automatic learning of hierarchical representations from data. This approach has revolutionized numerous fields by achieving state-of-the-art performance in tasks such as image and speech recognition, natural language processing, and game playing. Deep learning models, often referred to as deep neural networks, mimic the structure and function of the human brain's neural networks, making them highly effective for complex pattern recognition.

- **Key Concepts in Deep Learning**
  - ➢ **Neural Networks:** The fundamental building block of deep learning is the neural network, composed of interconnected nodes (neurons) organized in layers. Each neuron receives input, processes it through an activation function, and passes the output to the next layer. A neural network typically consists of an input layer, one or more hidden layers, and an output layer.
  - ➢ **Layers:**
    - ❖ Input Layer: The initial layer that receives raw data.
    - ❖ Hidden Layers: Intermediate layers that transform the input into more abstract representations. Deep learning models have multiple hidden layers, hence the term "deep."
    - ❖ Output Layer: The final layer that produces the prediction or classification result.

- **Activation Functions:** These functions introduce non-linearity into the network, allowing it to learn complex patterns. Common activation functions include ReLU (Rectified Linear Unit), Sigmoid, and Tanh.
- **Training:** Deep learning models are trained using large datasets through a process called backpropagation, which involves adjusting the weights of the neurons to minimize the error (loss) between the predicted and actual outputs. The optimization is typically done using gradient descent algorithms.
- **Loss Functions:** These functions measure the difference between the predicted output and the actual output. Common loss functions include Mean Squared Error for regression tasks and Cross-Entropy Loss for classification tasks.

- **Architectures in Deep Learning**
  - **Convolutional Neural Networks (CNNs):** CNNs are designed for processing grid-like data such as images. They use convolutional layers to automatically learn spatial hierarchies of features. Key components of CNNs include convolutional layers, pooling layers, and fully connected layers. CNNs excel in image classification, object detection, and image generation tasks.
  - **Recurrent Neural Networks (RNNs):** RNNs are suited for sequential data, such as time series or natural language. They have connections that form directed cycles, allowing information to persist. Variants like Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) address issues like vanishing gradients and can capture long-term dependencies. RNNs are widely used in language modeling, speech recognition, and machine translation.
  - **Generative Adversarial Networks (GANs):** GANs consist of two neural networks, a generator and a discriminator, that compete against each other. The generator creates fake data, while the discriminator tries to distinguish between real and fake data. This adversarial process improves the generator's ability to produce realistic data. GANs are used in image synthesis, style transfer, and data augmentation.

- **Transformers:** Transformers have become the dominant architecture in natural language processing due to their ability to handle long-range dependencies and parallelize training. They use self-attention mechanisms to weigh the importance of different words in a sequence. Notable models include BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer).



- **Figure 1.6:** VGG16 Model

1.5 Aim

The aim of the project is to develop the system using various Machine Learning Algorithms to predict the **Autism Spectrum Disorder**.

The project aims to harness machine learning algorithms for predicting Autism Spectrum Disorder (ASD), a complex neurodevelopmental condition. By collecting diverse datasets encompassing behavioral traits, medical history, and demographic information, the endeavor seeks to develop accurate predictive models. Various algorithms including Decision Tree, Random Forest, Support Vector Machine and Neural Networks will be explored to discern patterns and relationships within the data. Rigorous validation techniques like model interpretation methods, will ensure the robustness and generalizability of the developed models. Ethical considerations, including data privacy and responsible AI usage, will underpin the project's approach towards advancing ASD diagnosis and intervention.

1.6 Problem Statement

The problem statement chosen for this project is to predict Autism Spectrum Disorder with the help of machine learning models. The current diagnostic methods for Autism Spectrum Disorder (ASD) suffer from subjectivity, variability. This hinders early and accurate detection. To address these challenges, our project aims to develop a machine learning-based system that can provide a more objective and scalable approach to ASD detection, improving the overall quality of care and support for individuals on the autism spectrum.

1.7 Objectives

- The data collection.
- Data Processing and Feature Extractions
- Exploratory Data Analysis
- Building the Model using Machine-learning classification algorithms.
- Evaluate the performance of the Algorithm (Logistic Regression, Decision Trees, Random Forest, Support Vector Machine, ADA Boost).
- Web Interface.

CHAPTER 2

CHAPTER 2

LITERATURE SURVEY

2.1 Literature Survey Papers

[1] Title: Machine Learning-Based Models for Early Stage Detection of Autism Spectrum Disorders

Year: 2019

Authors: Tania Akter, Md. Shahriare Satu, Md. Imran Khan, Pietro Lió, , Mohammad Hanif Ali

Description: In this paper they defined the ASD as Autism Spectrum Disorder (ASD) is a group of neurodevelopmental disabilities that are not curable but may be ameliorated by early interventions. They gathered early-detected ASD datasets relating to toddlers, children, adolescents and adults, and applied several feature transformation methods, including log, Z-score and sine functions to these datasets. Various classification techniques were then implemented with these transformed ASD datasets and assessed for their performance. They found SVM showed the best performance for the toddler dataset, while Adaboost gave the best results for the children dataset, Glmboost for the adolescent and Adaboost for the adult datasets. The feature transformations resulting in the best classifications was sine function for toddler and Z-score for children and adolescent datasets. After these analyses, several feature selection techniques were used with these Z-score-transformed datasets to identify the significant ASD risk factors for the toddler, child, adolescent and adult subjects. The results of these analytical approaches indicate that, when appropriately optimised, machine learning methods can provide good predictions of ASD status. This suggests that it may possible to apply these models for the detection of ASD in its early stages. The algorithm that provided the best accuracy varied depending on the age group in the study. For toddlers, the Naïve Bayes algorithm[98.33%] showed the best performance for children, the Adaboost algorithm[97.20%] yielded the highest accuracy, Glmboost [93.89%] was found to be the most effective algorithm for adolescents, Adaboost [98.36%] also performed well for adults in terms of accuracy.

These results indicate that different algorithms may be more suitable for different age groups when it comes to predicting Autism Spectrum Disorder (ASD) status.

[2] Title: Machine Learning Classifiers for Autism Spectrum Disorder

Year: 2019

Authors: Dadang Eman, Andi W.R Emanuel

Description: Autism Spectrum Disorder (ASD) is a brain development disorder that impacts communication and social interaction abilities. Numerous studies have utilized machine learning methods, including support vector machines, decision trees, naïve Bayes, random forests, logistic regression, and K-nearest Neighbors, to classify autism. This study focuses on reviewing ASD using supervised learning algorithms. Sixteen research articles meeting the study criteria were collected from online databases, with the most commonly used algorithm being the support vector machine (SVM) at 68.75%. The application of machine learning in ASD aims to accelerate and enhance the accuracy of diagnosis processes. The most widely used algorithm in the literature study was the Support Vector Machine (SVM), utilized in 68.75% of the articles. Other algorithms included Decision Trees (31.25%) and Random Forest (31.25%). The research articles focused on various prediction goals related to ASD, such as detecting ASD/ADHD using decision tree, random forest, SVM, logistic regression, and linear discriminant analysis. The studies reported high accuracy levels, with an Area Under Curve (AUC) of 0.965 in one study. Machine learning algorithms can analyze large datasets quickly and efficiently, enabling the early detection of ASD symptoms based on various input parameters. Machine learning models can identify complex patterns and relationships in data that may not be easily recognizable by human clinicians, leading to more accurate and timely diagnoses. By automating the screening process using machine learning algorithms, healthcare professionals can save time and resources, leading to faster assessments and interventions for individuals with ASD. Machine learning can help tailor treatment plans and interventions based on individual characteristics and responses, leading to more personalized and effective care for individuals with ASD. By providing data-driven insights and predictions, machine learning can support healthcare providers in making informed decisions regarding ASD diagnosis and treatment strategies, ultimately improving patient outcomes. The application of machine learning in ASD diagnosis was found to increase accuracy and speed up the decision-making process, potentially leading to earlier detection and treatment for individuals with ASD. Early detection of ASD through machine learning can lead to improved independence for patients in the future, as well as aiding researchers in further studies on ASD. These findings highlight the potential of machine learning algorithms in enhancing the classification and diagnosis of Autism Spectrum Disorder.

[3] Title: Learning Clusters in Autism Spectrum Disorder: Image-Based Clustering of Eye-Tracking Scanpaths with Deep Autoencoder

Year: 2019

Authors: Mahmoud Elbattah, Romuald Carette, Gilles Dequen, Jean-Luc Guérin, Federica Cilia

Description: In this paper they defined the ASD as Autism spectrum disorder is a lifelong condition characterized by social and communication impairments. This study attempts to apply unsupervised Machine Learning to discover clusters in ASD. The key idea is to learn clusters based on the visual representation of eye-tracking scanpaths. The clustering model was trained using compressed representations learned by a deep autoencoder. Their experimental results demonstrate a promising tendency of clustering structure. Further, the clusters are explored to provide interesting insights into the characteristics of the gaze behavior involved in autism. In this study, the clustering experiments utilize the K-Means algorithm to identify clusters in the Autism Spectrum Disorder (ASD) dataset based on visual representations of eye-tracking scanpaths. The K-Means algorithm is a popular unsupervised clustering technique that partitions the data into K clusters by iteratively assigning data points to the nearest cluster centroid and updating the centroids based on the mean of the data points in each cluster. By applying the K-Means algorithm with different numbers of clusters (K), the study aims to discover meaningful groupings in the eye-tracking data that may reflect distinct patterns related to ASD characteristics. They used the k value as k[2:4] and found that good quality of cluster was achieved with k = 2 rather than k = 3 and 4. The use of deep autoencoder technology contributes significantly to the clustering model in this study by enabling the learning of compressed representations of the visual data obtained from eye-tracking scanpaths. The deep autoencoder, a type of artificial neural network, is employed in an unsupervised manner to automatically learn the encoding and decoding functions of the input data. In this case, the autoencoder learns to compress the high-dimensional visual representations of eye-tracking scanpaths into lower-dimensional codings. By utilizing the deep autoencoder, this study extracted meaningful features from the eye-tracking data, which are essential for clustering analysis. The compressed representations learned by the autoencoder serve as the basis for training the clustering model, allowing for the identification of clusters in the ASD dataset. This approach not only facilitates dimensionality reduction but also enables the model to capture essential patterns and structures within the data that may not be apparent in the original high-dimensional space.

[4] Title: A Machine Learning Approach to Predict Autism Spectrum Disorder

Year: 2019

Authors: Kazi Shahrukh Omar, Prodipta Mondal, Nabila Shahnaz Khan, Md. Rezaul Karim Rizvi, Md Nazrul Islam

Description: This paper focuses on the increasing prevalence of Autism Spectrum Disorder (ASD) and the challenges associated with expensive and time-consuming screening tests. Leveraging artificial intelligence and machine learning (ML), the research aims to predict autism at an early stage. Previous studies lacked conclusive results on predicting autism traits across different age groups, prompting the development of an effective prediction model using ML techniques. The model, merging Random Forest-CART and Random Forest-ID3, was evaluated using the AQ-10 dataset and 250 real datasets. Results indicated that the proposed model outperformed in terms of accuracy, specificity, sensitivity, precision, and false positive rate for both datasets. Additionally, a mobile application based on the prediction model was developed to predict ASD for individuals of any age. The research aimed to develop an effective prediction model based on machine learning techniques to predict ASD at an early stage. The study merged Random Forest-CART and Random Forest-ID3 algorithms to create the prediction model. Evaluation of the model was conducted using the AQ-10 dataset and 250 real datasets collected from individuals with and without autistic traits. The results of the evaluation showed that the proposed prediction model achieved better performance in terms of accuracy, specificity, sensitivity, precision, and false positive rate for both types of datasets. A mobile application based on the prediction model was developed to predict ASD traits for individuals of any age group. This study on predicting Autism Spectrum Disorder (ASD) using machine learning utilized the following algorithms,Random Forest-CART (Classification and Regression Trees),
Random Forest-ID3 (Iterative Dichotomiser 3). The evaluation results of the proposed prediction model showed improved performance in terms of accuracy, specificity, sensitivity, precision, and false positive rate for both the AQ-10 dataset and the 250 real datasets collected from individuals with and without autistic traits. The research on predicting Autism Spectrum Disorder (ASD) using machine learning has significant implications for the future of diagnosing and treating individuals with ASD. The development of an effective prediction model using machine learning techniques allows for the early detection of autism traits in individuals of any age group. By providing a more efficient and accurate screening method, the use of machine learning in predicting ASD can help reduce the time and cost associated with traditional screening tests.

CHAPTER 3

CHAPTER 3

SYSTEM REQUIREMENTS SPECIFICATION

3.1 Introduction to Specific Requirement Specification

Requirements analysis is critical for project development. Requirements must be documented, actionable, measurable, testable, and defined to a level of detail sufficient for system design. Requirements can be architectural, structural, behavioural, functional, and functional. A software requirements specification (SRS) is a comprehensive description of the intended purpose and the environment for software under development.

3.1.1 Feasibility Study

The feasibility of the project is analyzed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis, the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

3.1.2 Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system is well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

3.1.3 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand for the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement as only minimal or null changes are required for implementing this system.

3.1.4 Social Feasibility

The aspect of the study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2 Functional Requirements

The tools to execute the Python programs can be many, among that we can go with Visual Studio, Anaconda Navigator (Jupyter Notebook) or any IDLE based on Python. The online tool from Google can be an effective solution towards the execution of Python coding.

3.2.1. Approach 1: Pycharm (Anaconda Navigator)

PyCharm, a powerful IDE from JetBrains, simplifies Python development with a wealth of features. Intelligent code completion suggests keywords and functions as you type, reducing errors and speeding up coding. Navigating large codebases is a breeze with PyCharm's tools, allowing you to jump between code elements and refactor your code safely for better maintainability. An integrated debugger lets you step through code line by line, inspect variables, and set breakpoints to efficiently fix errors. Built-in testing tools work seamlessly with popular frameworks like unittest and pytest, helping you write robust and reliable Python applications. Version control integration with Git, Subversion, or Mercurial allows you to track code changes, collaborate with others, and revert to previous versions if needed. For web development, PyCharm offers features tailored to Django and Flask frameworks, and for scientific computing, it integrates with libraries like NumPy and SciPy. Whether you're a beginner or a seasoned developer, PyCharm's free Community Edition or feature-rich Professional Edition can significantly enhance your Python development experience.

Figure 3.1: Pycharm

3.2.2 Approach 2: Python IDLE

Python IDLE (Python Integrated Development and Learning Environment) help is writing the code very effectively and efficiently and helpful tool to the Python learning who wants to start from the scratch and beginners can have an advantage to execute the code easily. This is a powerful interpreter and compiler to run the code. It's an Interactive Interpreter also known as shell, which executes the python written code, reads the input, evaluate the statements and print the output on the standard output screen provided.File Editor Help to edit the code, save the program in text files and store as .py file.

Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

Looking for Python 2.7? See below for specific releases

Figure 3.2: Python IDLE

Python:

Python's creation can be traced back to the late 1980s, with Guido van Rossum as the language's "Benevolent Dictator for Life" (BDFL) – a title reflecting his central role in its design and development. Van Rossum's vision was to create a language that was not only powerful but also emphasized code readability. He drew inspiration from languages like

ABC, which focused on clarity and ease of use.

The name "Python" itself has an interesting origin. Some accounts suggest it was inspired by the British comedy group Monty Python's Flying Circus, while others claim it simply reflects van Rossum's enjoyment of the namesake snake. Regardless of the exact inspiration, the name "Python" has become synonymous with a language that is both powerful and approachable.

Over the years, Python has undergone continuous evolution, with new features and functionalities being added to keep pace with the ever-changing technological landscape. However, the core principles of simplicity, readability, and versatility remain at the heart of the language, ensuring its continued popularity and influence in the programming world.

Python has become one of the most popular programming languages in the world, and for good reason. Here's a closer look at the key advantages that make Python such a compelling choice for programmers of all levels:

- **Simplicity and Readability:** Python boasts a clean and concise syntax that closely resembles natural language. This makes it easier to learn and understand, even for beginners with no prior coding experience. Imagine writing instructions for a recipe in a way that's clear and easy to follow – that's the essence of Python code.
- **Versatility and Wide Range of Applications:** Python is a general-purpose language, meaning it can be effectively applied to a vast array of tasks. From web development and data analysis to machine learning and scientific computing, Python has the tools and libraries to tackle diverse programming challenges. Think of Python as a Swiss Army knife for programmers, with functionalities for various needs.
- **Large and Active Community:** The Python community is thriving and highly supportive. This means you'll find a wealth of online resources, tutorials, and forums to help you learn and troubleshoot any issues you encounter. Whether you're stuck on a specific problem or simply seeking guidance, there's a vast network of Python programmers ready to assist you.
- **Extensive Standard Library and Third-Party Packages:** Python comes equipped with a rich standard library that offers a collection of built-in modules for common functionalities like file handling, networking, and working with data structures. This saves you time by providing pre-written code you can leverage in your projects, eliminating the need to reinvent the wheel. Beyond the standard library, there's a vast ecosystem of third-party packages available that further extend

Python's capabilities. These packages cover virtually any programming domain imaginable, providing specialized tools for specific tasks.

- **Focus on Code Readability:** One of the core philosophies behind Python's design is its emphasis on code readability. This means the language encourages writing clear and well-structured code that's easy to understand, not just for you but also for other programmers who may collaborate on the project or maintain the codebase in the future. Imagine a well-organized kitchen where everything has its place – Python code strives for the same level of clarity and maintainability.

Python's versatility extends far beyond the realm of code. Here's a glimpse into some of the captivating applications where Python excels:

- **Web Development with Frameworks:** Python has become a dominant force in web development, thanks in part to the emergence of powerful frameworks like Django and Flask. Django is a high-level, full-stack framework ideal for building complex web applications with a focus on security and scalability. Flask, on the other hand, is a lightweight microframework that offers more flexibility for developers who prefer a more hands-on approach. Using these frameworks, Python empowers programmers to create dynamic and interactive web experiences.

- **Scripting & Automation:** Python's ability to automate repetitive tasks makes it a valuable tool for streamlining workflows and boosting productivity. Imagine automating mundane data entry tasks or scheduling file backups – Python scripts can handle these and many other repetitive processes, freeing you to focus on more strategic work. Libraries like os and shutil provide functionalities for interacting with the operating system, while tools like Selenium can be used for automating web browser interactions.

- **Game Development:** Python's simplicity and ease of use make it an attractive option for aspiring game developers. Libraries like Pygame offer a comprehensive set of tools for creating engaging 2D games. Whether you're developing a simple puzzle game or a more complex side-scroller, Python provides a solid foundation for bringing your game concepts to life. The relatively shorter development time compared to some other languages makes Python ideal for prototyping and experimenting with game ideas.

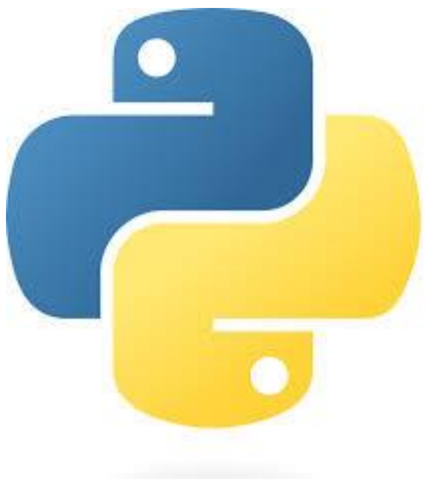


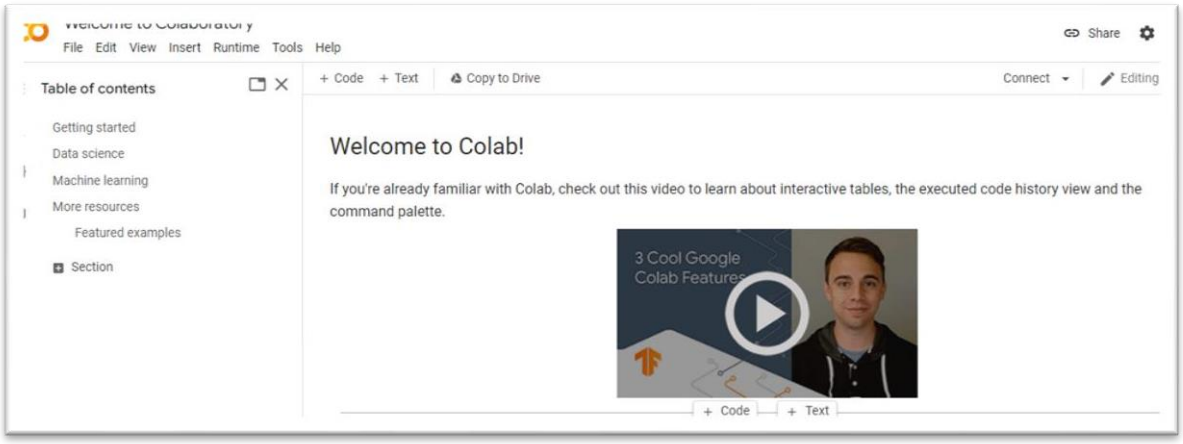
Figure 3.3: Python

3.2.3 Approach 3: Google Colab

Google Colab, also called as Colab in short is a powerful Machine Learning, Deep Learning and Data Analysis Tool that allows mixing the Python script along with text document. Rich support for Plotting the graphs, Diagram, Charts, Import Images, HTML Tags Support and LATEX format API conversions. Additional functional is it works on cloud model where document can be accessed and run on any platform independent of framework design and operating system. The runtime support for Virtual Hard Disk space and 12GB of RAM to execute the application is very excited feature of Colab. The uploading of files is very easy in this application so that it connects to the runtime.

Some of the important feature is:

- Remote Desktop Connection
- Runtime Environment
- Dataset Upload Features
- I/O operations and Operating System API Support
- General Processing Unit (GPU) availability



Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view and the command palette.

Figure 3.4: Google Colab

3.2.4 Approach 4: Jupyter Notebook

In a Jupyter notebook dedicated to machine learning endeavors, you embark on a journey of exploration, analysis, and experimentation within a single, interactive environment. This digital canvas seamlessly integrates code execution, visualization tools, and explanatory text, fostering a narrative that unfolds as you delve deeper into your data and models. The beauty of Jupyter lies in its versatility: you can weave together Python code snippets with markdown cells, allowing you to articulate the problem at hand, elucidate the intricacies of your dataset, and elucidate the rationale behind your chosen methodologies. This narrative structure not only guides readers through your thought process but also facilitates collaboration and knowledge sharing among peers and stakeholders. Moreover, Jupyter's real-time feedback loop empowers you to iterate rapidly, refining your models, and visualizations with each iteration. By encapsulating your analysis, findings, and insights within a single, executable document, Jupyter notebooks promote transparency, reproducibility, and effective communication in the realm of machine learning.

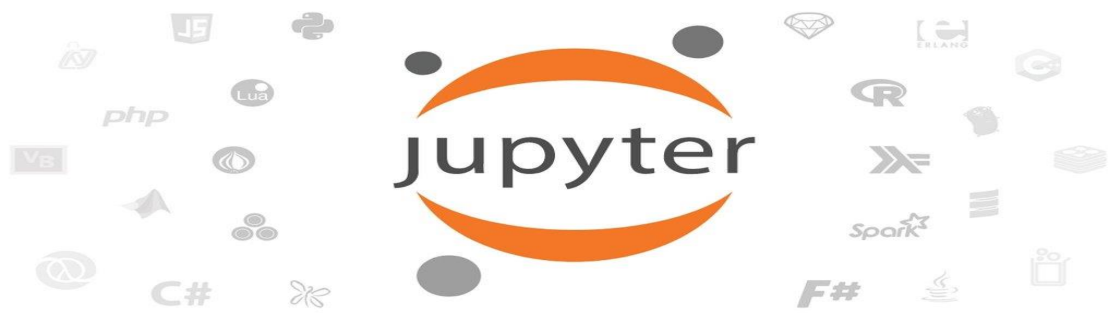


Figure 3.5: Jupyter Notebook

3.3 Non-Functional Requirements

These are requirements that are not functional in nature, that is, these are constraints within which the system must work. The program must be self-contained so that it can easily be moved from one Computer to another. It is assumed that network connection will be available on the computer on which the program resides.

Non-functional requirements (NFRs) define the constraints and environment within which the system must operate. These NFRs essentially set the stage for how the system behaves and interacts with its surroundings.

The NFRs focus on two key aspects:

- **Deployment and Portability:**
  - ➢ Self-Contained Program: This requirement ensures the program is a complete package, including all necessary files and libraries to function. Imagine a self-contained recipe that includes all ingredients and instructions, ready to be used in any kitchen. This allows the program to be easily moved from one computer to another without needing additional setup on the new machine. It promotes easy deployment and avoids compatibility issues.
- **Network Connectivity:**
  - ➢ Network Assumption: This NFR assumes a network connection will be available on the computer where the program runs. This suggests the program might rely on online resources or communicate with other systems over the network. It's

important to note that if internet access is crucial, alternative scenarios for offline functionality or limited connectivity might need to be considered as well.

- **Capacity, scalability, and availability**

The system shall achieve 100 percent availability at all times. The system shall be scalable to support additional clients and volunteers.

- ➢ 100% Availability: This is an ambitious target, as achieving perfect uptime is extremely difficult in practice. It likely means the system must be highly reliable with minimal downtime for maintenance or unexpected issues.
- ➢ Scalability: The system should be able to handle an increasing number of users (clients) and volunteers without significant performance degradation. This might involve using modular architecture that can be easily expanded with additional resources.

- **Maintainability**

The system should be optimized for supportability or ease of maintenance as far as possible. This may be achieved through the use of documentation of coding standards, naming conventions, class libraries, and abstraction.

- ➢ Coding Standards & Naming Conventions: Consistent coding styles and meaningful variable/function names improve code readability for future developers who need to modify or troubleshoot the system.
- ➢ Class Libraries: Reusable code components encapsulated in classes promote code organization and maintainability.
- ➢ Abstraction: Complex functionalities can be abstracted into higher-level concepts, making the code easier to understand and modify.

- **Randomness, verifiability, and load balancing**

The system should be optimized for supportability or ease of maintenance as far as possible.

- ➢ Randomness: It's unclear how randomness would be relevant for maintainability. It might be a separate requirement related to a specific system function, needing further context.
- ➢ Verifiability: This could be related to ensuring the system's outputs or behavior can be easily checked and validated.
- ➢ Load Balancing: This is typically a requirement for handling high volumes of traffic or user requests. It involves distributing workload across multiple resources to

prevent overloading any single component.

3.4 Requirements Specification

Python, an interpreted, high-level, and general-purpose programming language, was conceived by Guido van Rossum and made its debut in 1991. One of Python's distinctive features is its design philosophy, which prioritizes code readability, often achieved through the use of significant whitespace. This characteristic fosters clarity and simplicity in both small-scale scripting tasks and large-scale software development projects. Van Rossum served as the guiding force behind Python's evolution until stepping down from leadership responsibilities in July 2018. Python enjoys widespread adoption across various operating systems, thanks to the availability of interpreters tailored to different platforms. CPython stands out as the reference implementation of Python and is distinguished by its open-source nature, encouraging community contributions and collaborative development. The Python Software Foundation oversees the management of Python and CPython, fostering an inclusive and supportive ecosystem for developers worldwide.

3.4.1 Software Requirements

Software requirements serve as a roadmap for developing a software project. They encompass functional requirements, specifying what the software should do, and non-functional requirements, detailing how it should perform under various conditions like usability, security, and scalability. User requirements focus on the needs of end-users, while system requirements outline the capabilities and constraints of the overall system, including hardware and software components. Business requirements articulate high-level goals from a business perspective, guiding the project's direction. Software requirements refer to the specifications and dependencies necessary for a particular software application to function properly. These requirements encompass various aspects, including the operating system compatibility, hardware specifications, prerequisite software libraries, and any additional tools or components needed for installation and execution. Understanding and fulfilling software requirements is crucial for ensuring the smooth deployment and operation of the software, as it dictates the environment in which the application can run effectively. Additionally, software requirements documentation typically outlines the minimum and recommended configurations, installation instructions, and any potential constraints or limitations users may encounter. Overall, software requirements serve as a blueprint for developers and users alike, guiding the setup and usage of the software to meet the desired objectives efficiently. Use cases illustrate user-system interactions, helping to understand user needs and system functionality. Constraints, such as budgetary or regulatory limitations, also factor into requirements. Clear and traceable requirements are crucial for ensuring that development efforts align with user needs and project objectives, ultimately leading to a successful software product.

➢ Operating System	: Windows 64-bit
➢ Technology	: Python
➢ IDE	: Python IDLE
➢ Tools	: Anaconda
➢ Python Version	: Python 3

3.4.2 Hardware Requirements

Hardware requirements specify the physical components necessary to support the software system. These include the CPU type and speed, RAM amount, and storage space needed for installation and data storage. Graphics card specifications may be outlined for applications requiring graphical capabilities, and network requirements detail minimum bandwidth and protocol compatibility for network-based systems. Additionally, compatibility with peripherals like printers and scanners may be specified, along with supported operating systems and any other necessary hardware devices. These requirements ensure that the software can function effectively on various hardware configurations, optimizing performance and user experience while aiding developers in ensuring compatibility across different environments. Hardware requirements detail the necessary components and resources needed to support the software system, including CPU type and speed, RAM amount, and storage space. They ensure that the software can function effectively on different hardware configurations while optimizing performance and user experience. These requirements also aid developers in ensuring compatibility across various environments.

➢ Processor Type	: Intel Core TM i5
➢ Speed	: 2.4 GHZ
➢ RAM	: 8 GB RAM
➢ Hard disk	: 80 GB HDD
➢ Input Device	: Mouse, Keyboard

CHAPTER 4

CHAPTER 4

SYSTEM DESIGN

System design is the process of defining the architecture, interfaces, and data for a system that satisfies specific requirements. System design meets the needs of your business or organization through coherent and efficient systems. Once your business or organization determines its requirements, you can begin to build them into a physical system design that addresses the needs of your customers. The way you design your system will depend on whether you want to go for custom development, commercial solutions, or a combination of the two. System design requires a systematic approach to building and engineering systems. A good system design requires you to think about everything in an infrastructure, from the hardware and software, all the way down to the data and how it's stored.

4.1 System Architecture

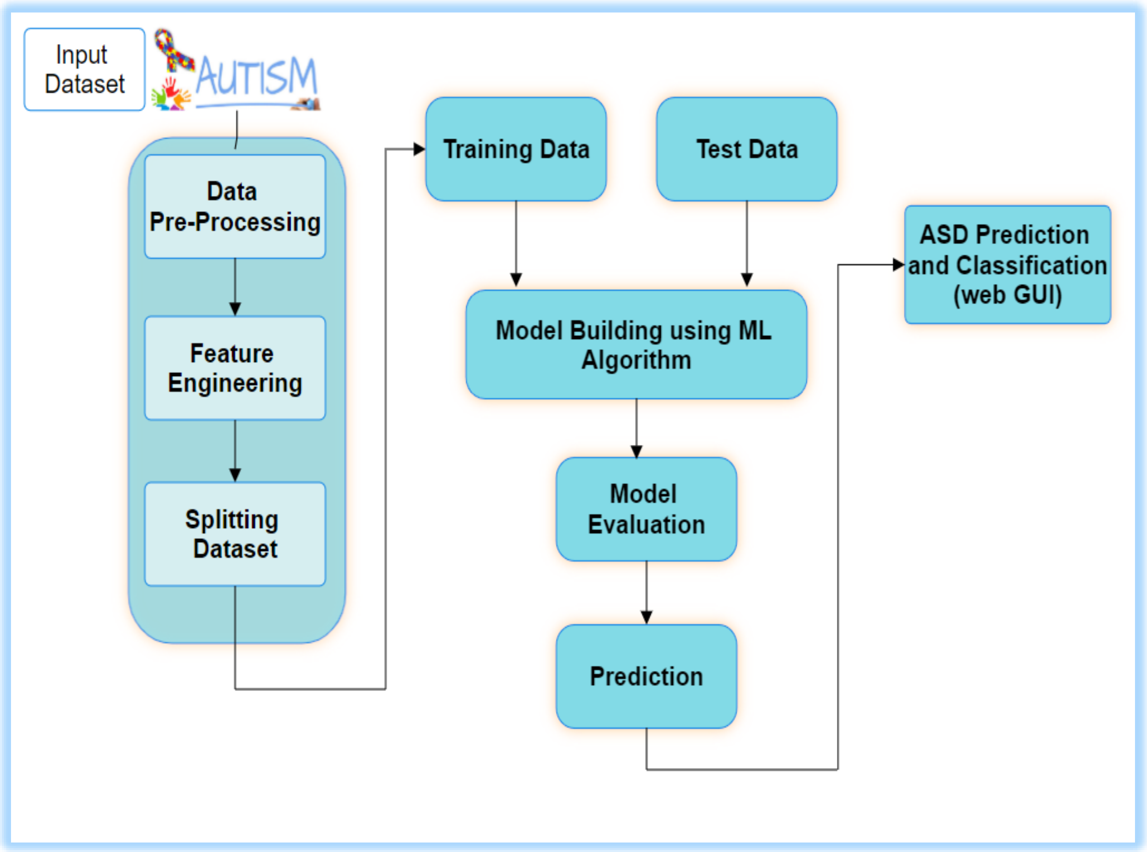


Figure 4.1: System Architecture

Autism spectrum disorder (ASD) is becoming an increasingly serious issue for people of all ages today.The maintenance of the subject's physical and mental health can be considerably helped by early discovery of this neurological condition. With the increased use of machine learning-based models for illness prediction, it is now possible to identify diseases early based on a variety of physiological and health parameters.

Autism Spectrum Disorder (ASD) diagnosis can be a lengthy process often involving various specialists and behavioral assessments. Machine learning offers potential to aid in early detection by analyzing data and predicting the likelihood of ASD. However, building a reliable model requires a well-defined process.

The first step is gathering data. This data could include information from questionnaires, observations, medical records, and genetic testing. It's crucial to ensure the data is comprehensive and includes relevant features like social behavior patterns, communication skills, repetitive actions, and sensory sensitivities. Once collected, the data is divided into two sets: training data and test data. The training data, like a student studying for a test, is used to train the machine learning model.Before training can begin, the data needs preparation. Imagine a messy room before organizing it. Data pre-processing involves cleaning the data by identifying and removing errors or inconsistencies. It might also involve formatting the data into a consistent structure and transforming it into a format the machine learning algorithm can understand.

This is where feature engineering comes in. Imagine a chef creating a new dish by combining ingredients. Feature engineering involves creating new features from the existing data that might be more informative for predicting ASD. For instance, the model might benefit from features that combine scores from various assessments to create a more comprehensive picture.Once the data is prepped and features are engineered, it's time to choose and train the machine learning model. Different algorithms work better for different tasks. Here, it's a classification problem - identifying individuals with or without ASD. Common choices include Support Vector Machines (SVMs) which can effectively find patterns in complex data, or Random Forests which combine multiple decision trees for robust predictions. The training process involves feeding the prepped data into the chosen algorithm. The algorithm then analyzes the data, identifying patterns and relationships between features and the presence or absence of ASD.After training, the model's performance is evaluated using the test data. This is like testing a student's knowledge after studying. The model makes predictions on the test data, and these predictions are compared to the actual diagnoses. Metrics like accuracy, precision, and recall help assess how well the model performs.Finally, if the model demonstrates good performance on the test data, it can be cautiously used to make predictions on new, unseen data. This could involve analyzing data from a child suspected of ASD to provide a preliminary indication of the likelihood of the disorder.

4.2 Data Flow Diagram

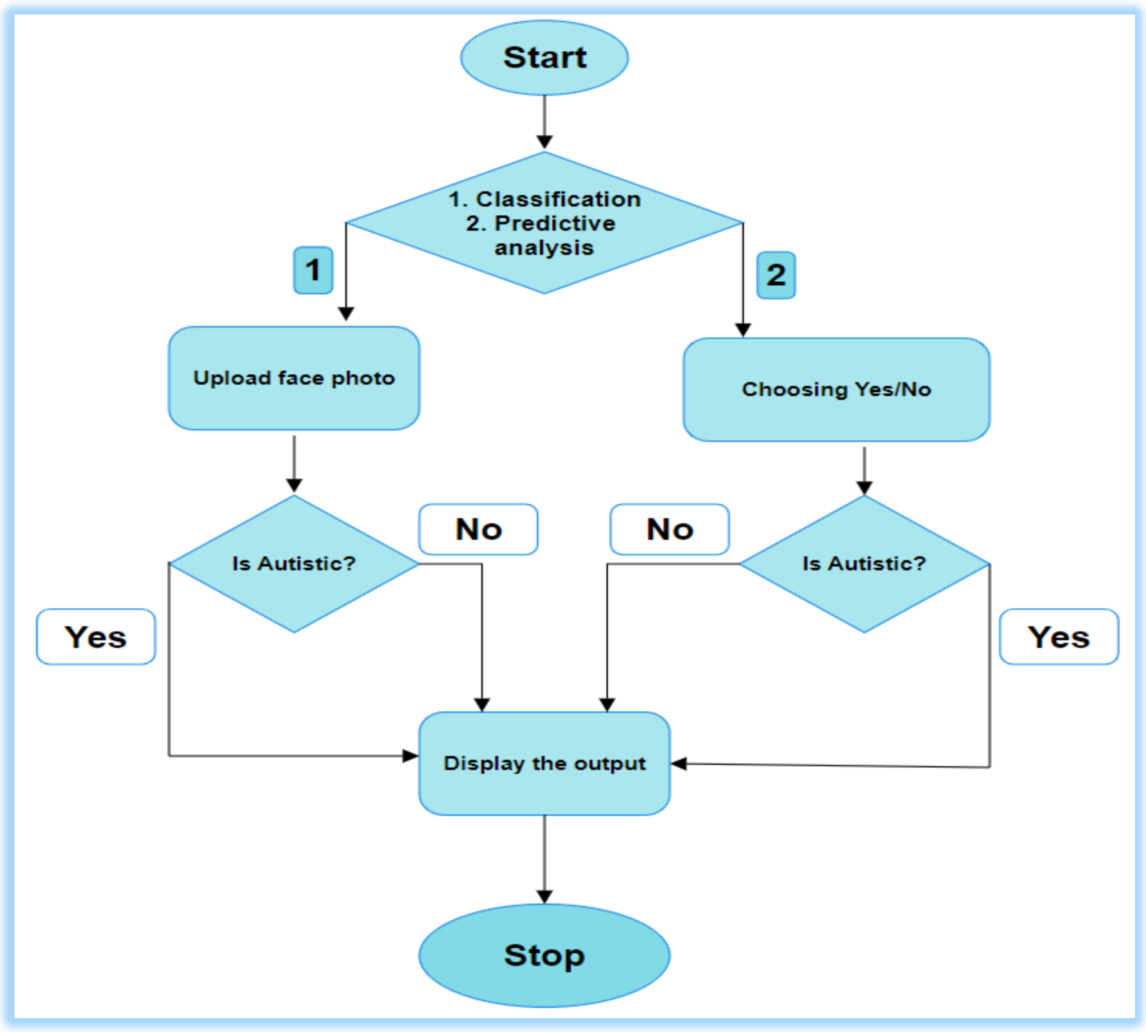


Figure 4.2: Data-Flow Diagram

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

The flowchart illustrates a decision-making process for determining if an individual is autistic using two distinct methods: classification and predictive analysis. The process begins with a decision point that asks whether to perform classification or predictive analysis. If classification is chosen, the user is prompted to upload a face photo. The system then analyzes the photo to determine if the person is autistic. If the result is positive, the process acknowledges that the person is autistic. If negative, the system proceeds to display the output, indicating the person is not autistic.

On the other hand, if predictive analysis is chosen, the user is asked to make a Yes/No choice based on specific criteria or questions provided by the system. Based on the user's response, the system determines whether the person is autistic. If the outcome is positive, it is acknowledged that the person is autistic. If negative, the system displays the output, confirming that the person is not autistic. The process concludes at the "Stop" point, signifying the end of the evaluation. This flowchart provides a structured approach to diagnosing autism through either a photo-based classification method or a predictive analysis based on user input.

4.3 Use Case Diagram

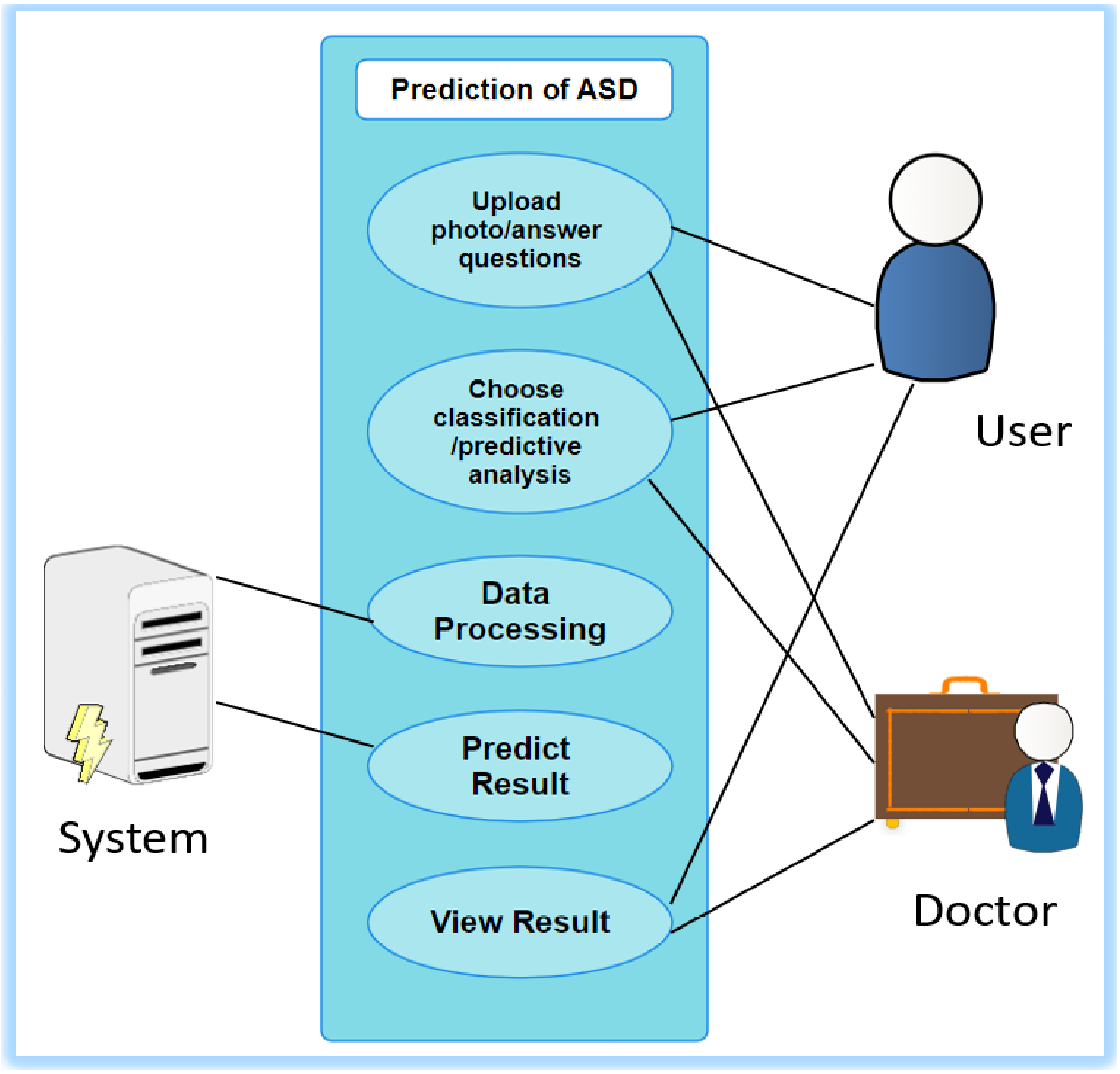


Figure 4.3: Use Case diagram

A use case diagram is used to represent the dynamic behaviour of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system. The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

This illustrates the process for predicting Autism Spectrum Disorder (ASD) involving interactions between the user, the system, and a doctor. The process begins with the user, who can either upload a photo or answer questions. Following this, the user selects between classification or predictive analysis methods. Once the method is chosen, the system takes over, performing data processing based on the provided inputs.

After processing the data, the system predicts the result regarding the likelihood of ASD. This result is then made available for viewing. Both the user and the doctor can access and interpret the results. The doctor's involvement suggests a professional evaluation and confirmation of the system's prediction. This structured process ensures that the ASD prediction is thorough, incorporating both automated analysis and professional medical insight.

4.4 Sequence Case Diagram

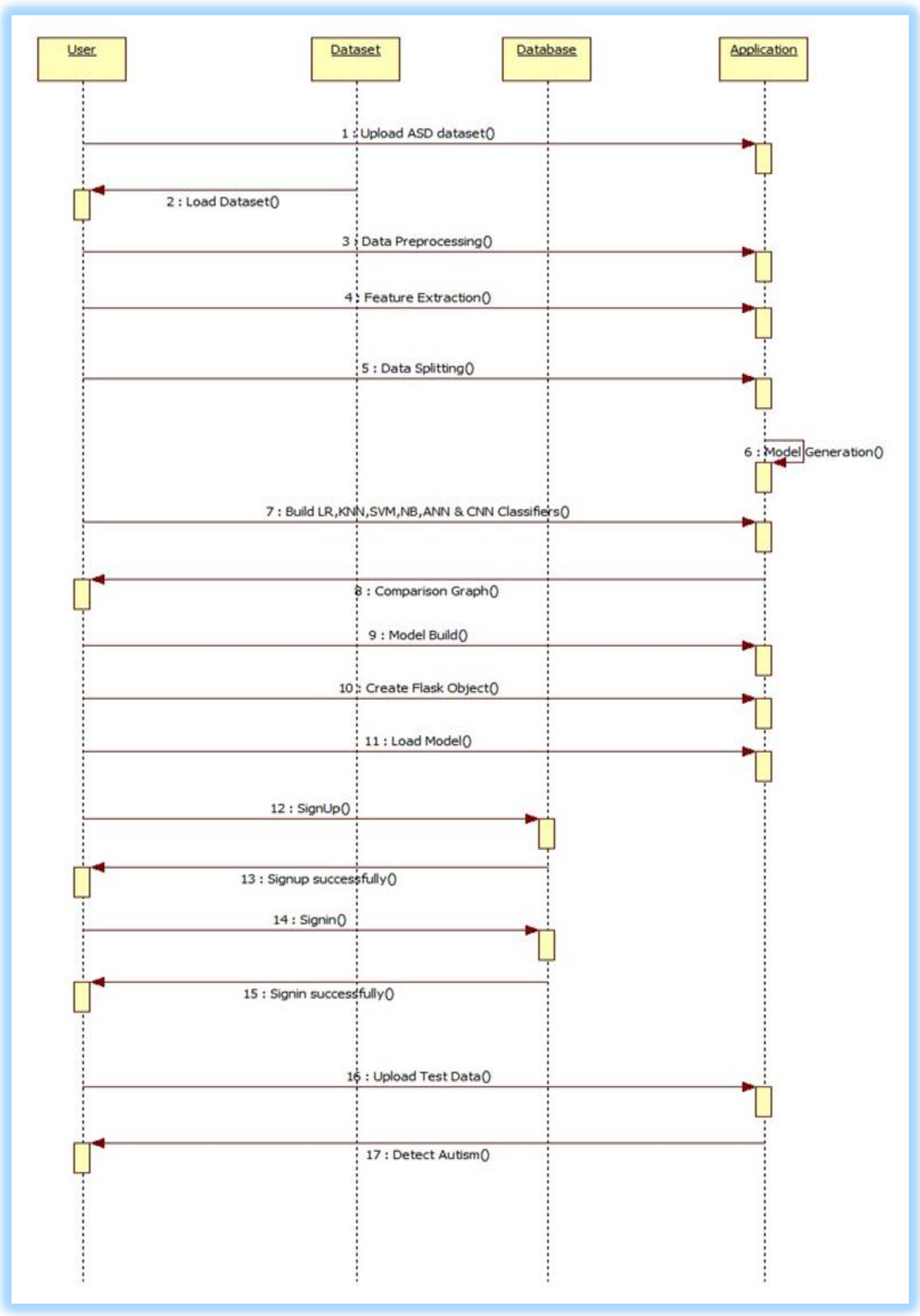


Figure 4.4: Sequence Case diagram

The sequence diagram is a systematic approach to show the relationship and interactions between objects and every object or entity is executed sequentially. Sequence of occurrence of events can also be depicted by using the sequence diagram. Sequence diagram is mostly used in Software Development models, Business Models and making the SRS of the project. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction. A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects. For example, lifelines in a sequence diagram for a banking scenario can represent a customer, bank teller, or bank manager. The communication between the customer, teller, and manager are represented by messages passed between them. The sequence diagram shows the objects and the messages between the objects.

CHAPTER 5

CHAPTER 5

IMPLEMENTATION

Systems implementation is a set of procedures performed to complete the design (as necessary) contained in the approved systems design document and to test, install, and begin to use the new or revised Information System. Implementation allows the users to take over its operation for use and evaluation. It involves training the users to handle the system and plan for a smooth conversion. The purpose of the implementation process is to design and create (or fabricate) a system element conforming to that element's design properties and/or requirements. The element is constructed employing appropriate technologies and industry practices.

5.1 Dataset

A dataset is basically a collection of related data. In this paper, we make use of a publicly available imbalanced dataset. An imbalanced dataset is one in which disparity occurs in the dependent variables. Imbalanced implies that there is an unequal distribution of classes. The particular dataset that we use is also an imbalanced one. In this project, we made use of a publicly available 2 imbalanced dataset. One dataset is used for "Predictive analysis". The data consists of 1985 rows and 28 columns where the last column describes whether the person will develop ASD traits in future(0,1). And also we added our survey data which is of 35 records. This dataset contains factors involved in developing ASD in children. It consists of the following features, A10 Autism Spectrum Quotient(columns A1-A10), Social Responsiveness Scale, Age Years, Qchat_10_Score, Speech Delay/Language Disorder, Learning disorder, Genetic Disorders, Depression, Global developmental delay/intellectual disability, Social / Behavioural Issues, Childhood Autism Rating Scale, Anxiety disorder, Sex, Ethnicity, Jaundice, Family mem with ASD along with various information. We are considering "ASD_traits" as a target feature and we treat Autistic as '0' and Non-Autistic as '1'.

Another dataset is used for "Image Classification". The dataset contains around 2600 records(taken from UCI) which includes Autistic and non-Autistic child face images.

5.1.1 Data Pre - Processing

Data preprocessing is an important step in the data mining process. It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

Some common steps in data preprocessing include:

Data cleaning: This step involves identifying and removing missing, inconsistent, or irrelevant data. This can include removing duplicate records, filling in missing values, and handling outliers.

Data integration: This step involves combining data from multiple sources, such as databases, spreadsheets, and text files. The goal of integration is to create a single, consistent view of the data.

Data transformation: This step involves converting the data into a format that is more suitable for the data mining task. This can include normalizing numerical data, creating dummy variables, and encoding categorical data.

Data reduction: This step is used to select a subset of the data that is relevant to the data mining task. This can include feature selection (selecting a subset of the variables) or feature extraction (extracting new variables from the data).

Data discretization: This step is used to convert continuous numerical data into categorical data, which can be used for decision tree and other categorical data mining techniques.

5.1.2 Feature Engineering

Feature engineering is a cornerstone of the machine learning process, demanding substantial time and attention from data scientists and analysts. This critical step involves crafting and refining features derived from raw data to enhance the performance of machine learning models and unearth valuable insights. Beginning with a comprehensive understanding of the dataset, practitioners meticulously select relevant features while filtering out noise and redundancy. Transformation techniques like normalization and encoding are then applied to preprocess the data, ensuring consistency and compatibility across features. Additionally, new features are often engineered through mathematical transformations or domain-specific knowledge, empowering models to capture intricate relationships and patterns within the data. Handling missing values and reducing dimensionality further refine the feature set, culminating in an iterative process of experimentation and refinement. While labor-intensive, effective feature engineering is essential for optimizing model accuracy, interpretability, and generalization capabilities, ultimately driving informed business decisions and unlocking the full potential of machine learning applications.

5.1.3 Splitting Dataset

Data splitting is a fundamental practice in data science, involving the division of a dataset into multiple subsets. Typically, this involves a two-part split, where one subset is utilized for model training, and the other for evaluation or testing purposes. This division is crucial for ensuring the robustness and generalizability of machine learning models, as it allows for independent validation of model performance on unseen data. By separating the dataset into training and testing sets, data splitting enables data scientists to train models on one subset while objectively assessing their predictive capabilities on another. This process plays a pivotal role in model development, helping to prevent overfitting and providing insights into a model's ability to generalize to new data. Overall, data splitting is a cornerstone of effective model building in data science, facilitating the creation of accurate and reliable predictive models.

5.1.4 Train Data

Training data serves as the bedrock upon which machine learning algorithms build their predictive prowess and refine their decision-making rules. Often referred to as the training dataset, learning set, or training set, this initial collection of data plays a pivotal role in the development of machine learning models. It provides algorithms with real-world examples and corresponding outcomes, enabling them to learn patterns, relationships, and trends inherent in the data. By repeatedly analyzing the training data, models iteratively adjust their parameters and update their internal representations to optimize performance. Essentially, training data acts as a teacher, guiding the model to understand the nuances and complexities of the problem domain. Through exposure to diverse instances and associated labels, machine learning models gain insights into what constitutes the expected output, allowing them to generalize their learning and make accurate predictions or perform desired tasks when presented with new, unseen data. This iterative process of learning from training data is fundamental to the success of machine learning algorithms, empowering them to extract actionable insights and drive meaningful outcomes across various domains and applications.

5.1.5 Test Data

Once we train the model with the training dataset, it's time to test the model with the test dataset. This dataset evaluates the performance of the model and ensures that the model can generalize well with the new or unseen dataset. The test dataset is another subset of original data, which is independent of the training dataset. However, it has some similar types of features and class probability distribution and uses it as a benchmark for model evaluation once the model training is completed. Test data is a well-organized dataset that contains data for each type of scenario for a given problem that the model would be facing when used in the real world. Usually, the test dataset is approximately 20-25% of the total original data for an ML project. At this stage, we can also check and compare the testing accuracy with the training accuracy, which means how accurate our model is with the test dataset against the training dataset. If the accuracy of the model on training data is greater than that on testing data, then the model is said to have overfitting.

The testing data should:

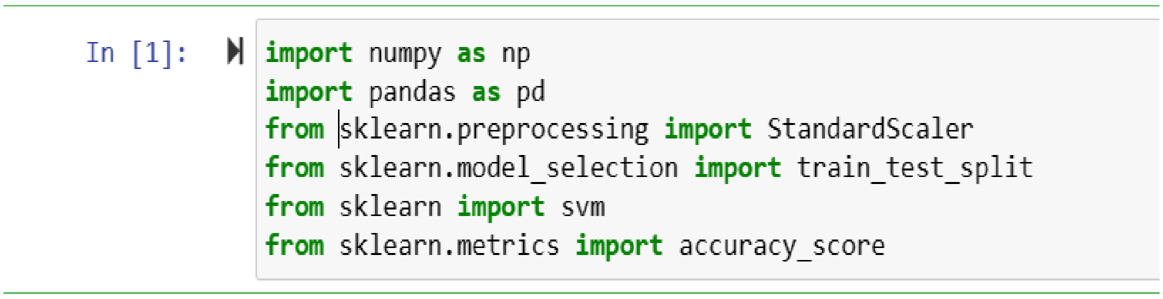
- Represent or part of the original dataset.
- It should be large enough to give meaningful predictions.

5.1.6 Model Evaluation

Model evaluation serves as a critical step in the multi-faceted process of model development, employing various metrics to gauge the performance and generalization capabilities of a machine learning model. Given the importance of ensuring robustness and reliability in predictive tasks, evaluating a model becomes indispensable for assessing its efficacy and identifying potential weaknesses. Metrics such as Accuracy, Precision, Recall, F1 score, Area under Curve, Confusion Matrix, and Mean Square Error provide valuable insights into different aspects of model performance, allowing practitioners to make informed decisions about model selection and refinement. Additionally, techniques like Cross Validation, incorporated during the training phase, serve as effective tools for evaluating model performance by assessing its ability to generalize to unseen data. By systematically analyzing these metrics and techniques, stakeholders can gain a comprehensive understanding of a model's strengths and limitations, paving the way for enhanced predictive capabilities and more informed decision-making in real-world applications.

5.1.7 Prediction

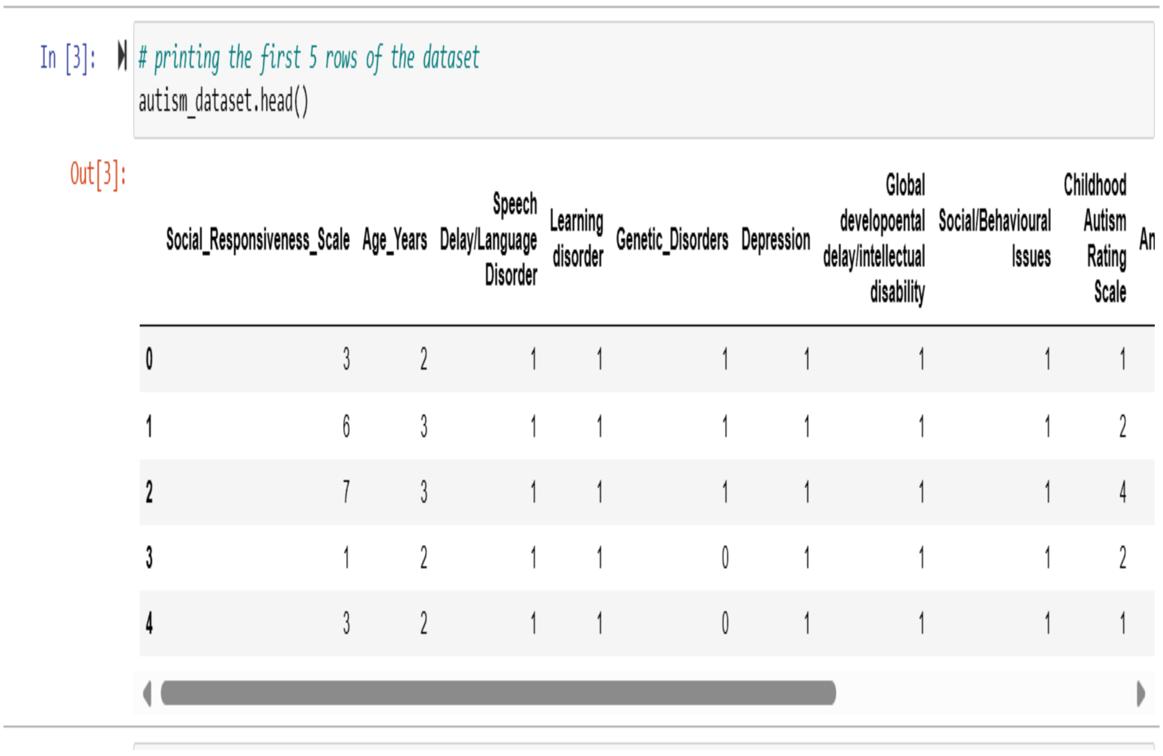
“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be. After training a machine learning model, prediction involves using the trained model to make predictions or infer information from new, unseen data. This process entails inputting new data into the model and obtaining predictions or estimates based on the patterns and relationships learned during training. Depending on the type of model and task, predictions can range from class labels in classification tasks to numerical values in regression tasks. The quality and accuracy of predictions depend on factors such as the quality of the training data, the complexity of the model, and how well the model generalizes to unseen data.



```
In [1]:  import numpy as np
         import pandas as pd
         from sklearn.preprocessing import StandardScaler
         from sklearn.model_selection import train_test_split
         from sklearn import svm
         from sklearn.metrics import accuracy_score
```

Figure 5.1: Importing Packages

Importing packages in programming languages like Python is essential for accessing pre-built functionality and libraries. This process allows developers to use functions, classes, and modules provided by these packages to simplify coding tasks and leverage existing solutions.



In [3]:
```
# printing the first 5 rows of the dataset
autism_dataset.head()
```

Out[3]:

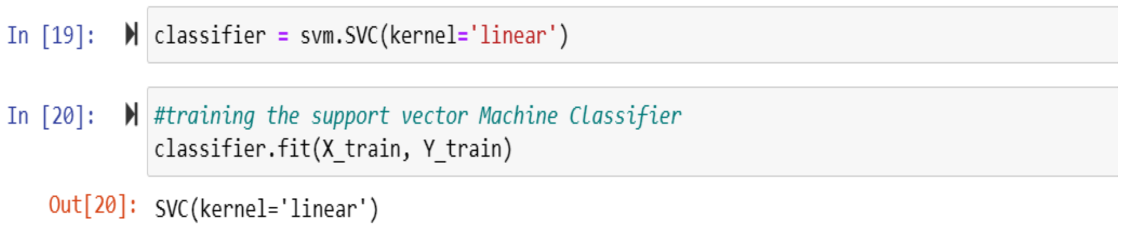
	Social_Responsiveness_Scale	Age_Years	Speech Delay/Language Disorder	Learning disorder	Genetic_Disorders	Depression	Global developoental delay/intellectual disability	Social/Behavioural Issues	Childhood Autism Rating Scale	An
0	3	2	1	1	1	1	1	1	1	
1	6	3	1	1	1	1	1	1	2	
2	7	3	1	1	1	1	1	1	4	
3	1	2	1	1	0	1	1	1	2	
4	3	2	1	1	0	1	1	1	1	

Figure 5.2: Dataset

5.2 Algorithm Used

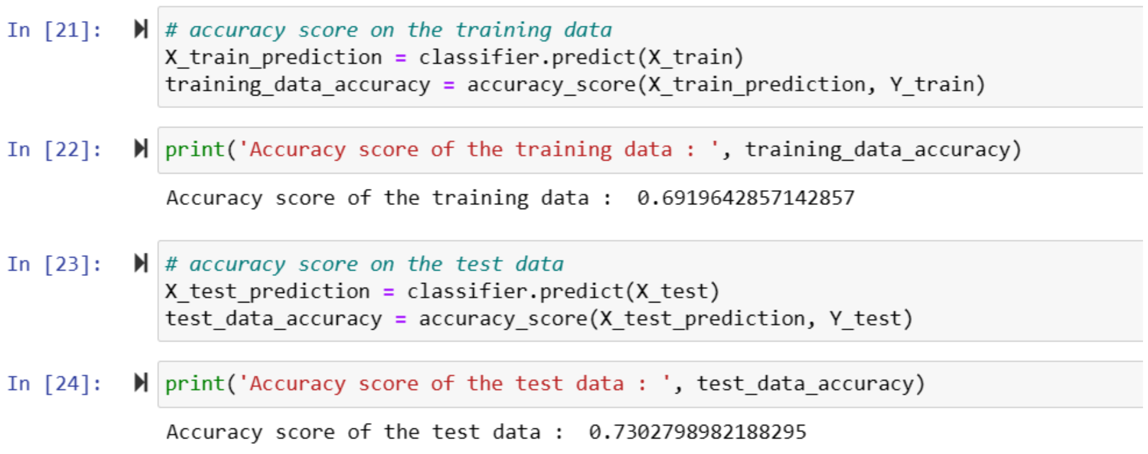
5.2.1 Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane. Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.



```
In [19]:  classifier = svm.SVC(kernel='linear')

In [20]:  #training the support vector Machine Classifier
          classifier.fit(X_train, Y_train)

Out[20]:  SVC(kernel='linear')
```

Figure 5.3: SVM Train Data

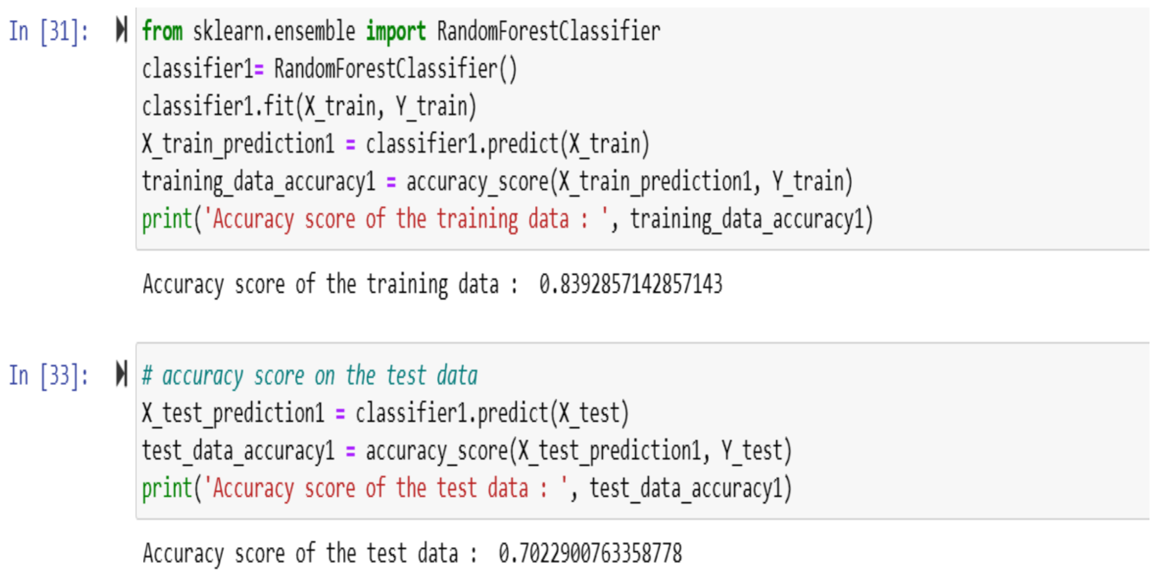


```
In [21]:  # accuracy score on the training data
          X_train_prediction = classifier.predict(X_train)
          training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

In [22]:  print('Accuracy score of the training data : ', training_data_accuracy)

          Accuracy score of the training data :  0.6919642857142857

In [23]:  # accuracy score on the test data
          X_test_prediction = classifier.predict(X_test)
          test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

In [24]:  print('Accuracy score of the test data : ', test_data_accuracy)

          Accuracy score of the test data :  0.7302798982188295
```

Figure 5.4: SVM Test Data

5.2.2 Random Forest

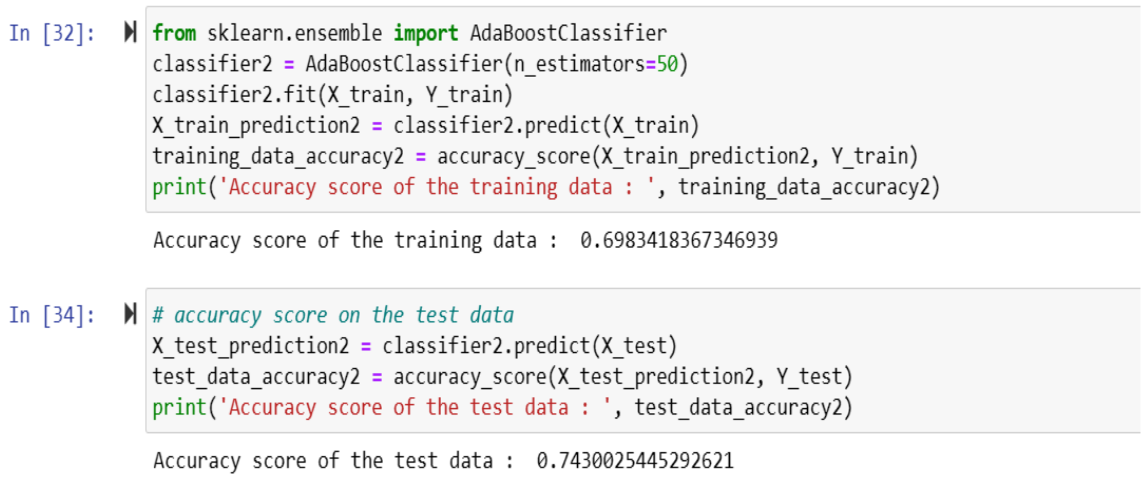
Random Forest is an ensemble learning algorithm widely used for both classification and regression tasks in machine learning. It constructs multiple decision trees during training by randomly sampling subsets of the training data and features. Each decision tree is built independently, recursively partitioning the feature space to maximize information gain (for classification) or decrease in impurity (for regression). During prediction, the outputs of individual trees are aggregated through majority voting for classification or averaging for regression, resulting in a final prediction. Random Forest offers high accuracy and robustness to overfitting, making it suitable for handling large datasets with high dimensionality. Additionally, it provides feature importance scores, aiding in feature selection. However, it may be less interpretable than simpler models and can be computationally expensive, particularly for large datasets. Despite these limitations, Random Forest remains a popular choice due to its versatility and effectiveness in various machine learning tasks.



```
In [31]:  from sklearn.ensemble import RandomForestClassifier
          classifier1= RandomForestClassifier()
          classifier1.fit(X_train, Y_train)
          X_train_prediction1 = classifier1.predict(X_train)
          training_data_accuracy1 = accuracy_score(X_train_prediction1, Y_train)
          print('Accuracy score of the training data : ', training_data_accuracy1)

          Accuracy score of the training data :  0.8392857142857143

In [33]:  # accuracy score on the test data
          X_test_prediction1 = classifier1.predict(X_test)
          test_data_accuracy1 = accuracy_score(X_test_prediction1, Y_test)
          print('Accuracy score of the test data : ', test_data_accuracy1)

          Accuracy score of the test data :  0.7022900763358778
```

Figure 5.5: Random Forest Train & Test Data

5.2.3 AdaBoost



```
In [32]:  from sklearn.ensemble import AdaBoostClassifier
          classifier2 = AdaBoostClassifier(n_estimators=50)
          classifier2.fit(X_train, Y_train)
          X_train_prediction2 = classifier2.predict(X_train)
          training_data_accuracy2 = accuracy_score(X_train_prediction2, Y_train)
          print('Accuracy score of the training data : ', training_data_accuracy2)

          Accuracy score of the training data :  0.6983418367346939

In [34]:  # accuracy score on the test data
          X_test_prediction2 = classifier2.predict(X_test)
          test_data_accuracy2 = accuracy_score(X_test_prediction2, Y_test)
          print('Accuracy score of the test data : ', test_data_accuracy2)

          Accuracy score of the test data :  0.7430025445292621
```

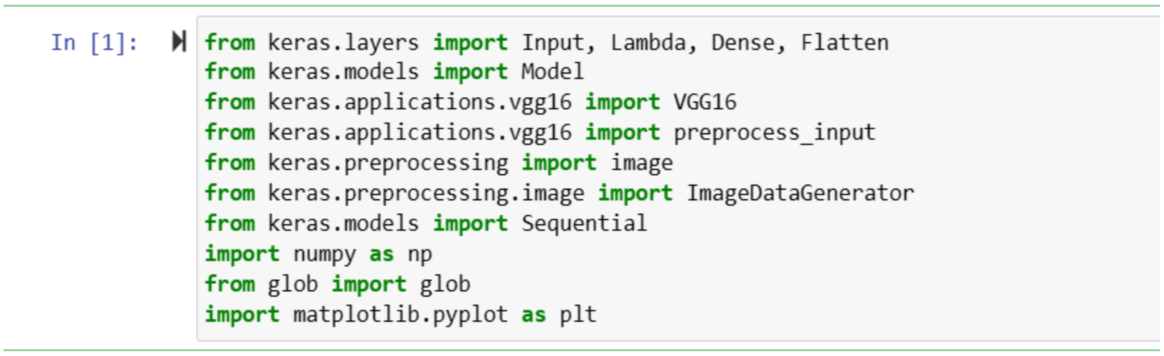
Figure 5.6: AdaBoost Train & Test Data

AdaBoost, short for Adaptive Boosting, is another ensemble learning algorithm commonly used for classification tasks. It works by iteratively training a sequence of weak classifiers, typically decision trees with limited depth, and adjusting the weights of incorrectly classified instances to focus more on difficult examples in subsequent iterations. During each iteration, AdaBoost assigns higher weights to misclassified instances, forcing the subsequent weak learners to focus more on these instances in an attempt to correct the errors made by the previous classifiers. The final prediction is then obtained by combining the individual weak classifiers through a weighted sum, where the weights are determined by the accuracy of each classifier. AdaBoost is known for its ability to improve classification performance by iteratively focusing on the most challenging instances, making it particularly effective in situations where a single classifier may struggle. However, AdaBoost can be sensitive to noisy data and outliers, which may adversely affect its performance. Despite this, AdaBoost remains a popular and widely used algorithm in machine learning due to its simplicity and effectiveness in boosting the performance of weak classifiers.

5.2.4 VGG 16

VGG16, also known as VGGNet, is a 16-layer convolutional neural network (CNN) model used for image recognition. It was developed by the Visual Geometry Group at Oxford University and proposed by K. Simonyan and A. Zisserman in their paper Very Deep Convolutional Networks for Large-Scale Image Recognition. VGG16 is known for its simplicity and uniformity, and is considered one of the best vision model architectures. It has 13 convolutional layers, three fully connected layers, and 138 million parameters. The "16" in VGG16 refers to the total number of layers in the network, including 13 convolutional layers and 3 fully connected layers. The architecture follows a straightforward pattern of stacking multiple convolutional layers with small 3x3 filters, followed by max-pooling layers to reduce spatial dimensions. This pattern is repeated multiple times, leading to a deep network with a uniform architecture.

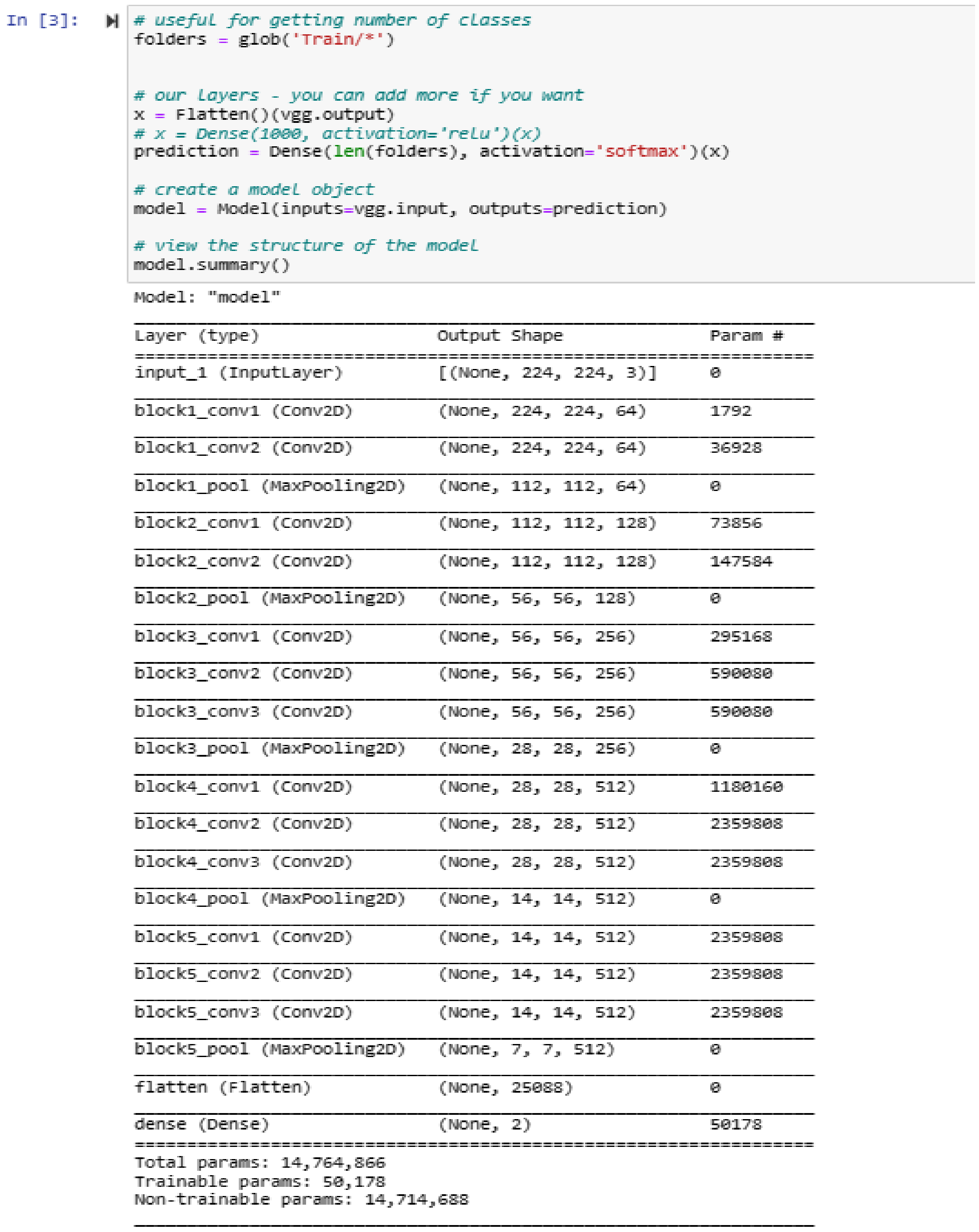
One of the key characteristics of VGG16 is its deep and uniform architecture, which allows it to capture intricate features at different scales in the input image. Despite its simplicity compared to later architectures like ResNet or Inception, VGG16 has been shown to perform well on a variety of image classification tasks, including the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) where it achieved top rankings.



```
In [1]:  from keras.layers import Input, Lambda, Dense, Flatten
         from keras.models import Model
         from keras.applications.vgg16 import VGG16
         from keras.applications.vgg16 import preprocess_input
         from keras.preprocessing import image
         from keras.preprocessing.image import ImageDataGenerator
         from keras.models import Sequential
         import numpy as np
         from glob import glob
         import matplotlib.pyplot as plt
```

Figure 5.7: Importing Packages

Importing packages code builds a convolutional neural network (CNN) for image classification, like training a machine to recognize objects in pictures. It starts by gathering necessary tools (like ingredients) from a library. Then, it sets up the basic structure of the model (like prepping your workspace). The core involves layers that analyze images (like filters) searching for specific patterns. Two sets of these filters, with increasing complexity, are defined. The code then incorporates layers that simplify the analyzed data (like summarizing key details). Following that, it transforms the data into a simpler format (like transforming chopped vegetables into a single pile). Next, a layer connects all the features, similar to combining ingredients in a recipe. Finally, the code creates an output layer with a specific number of categories (like the number of dishes you can cook). This layer ensures the model outputs probabilities that add up to 1 (like assigning portions). Lastly, the code configures how the model learns by setting parameters for error minimization and filter adjustments (like choosing cooking methods and times).

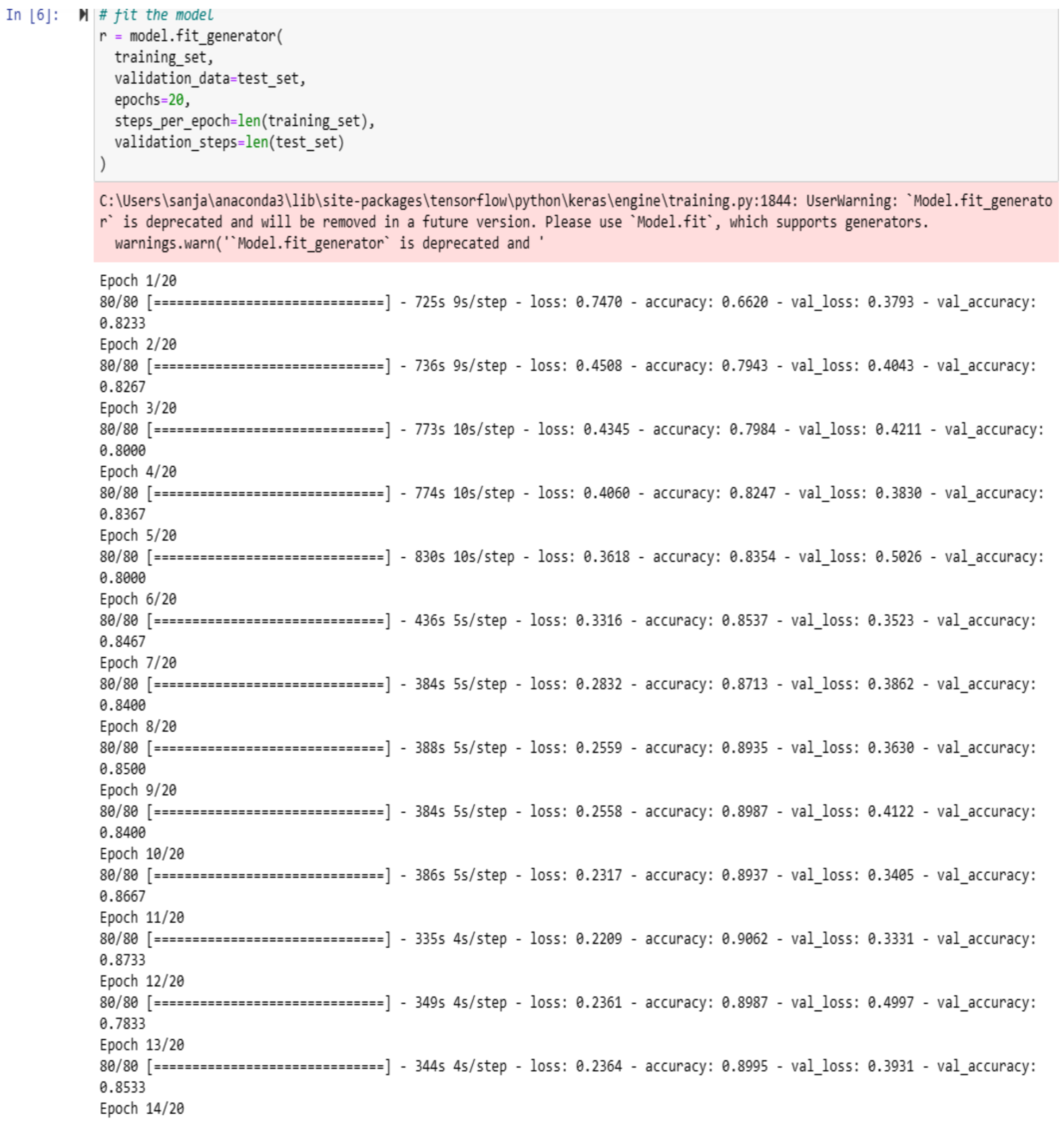


```
In [3]:  # useful for getting number of classes
         folders = glob('Train/*')


         # our layers - you can add more if you want
         x = Flatten()(vgg.output)
         # x = Dense(1000, activation='relu')(x)
         prediction = Dense(len(folders), activation='softmax')(x)

         # create a model object
         model = Model(inputs=vgg.input, outputs=prediction)

         # view the structure of the model
         model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 50178
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 5.8: Creating VGG 16 model

Imagine training a machine to recognize objects in pictures. This code builds a special kind of neural network, a convolutional neural network (CNN), to achieve this. It starts by gathering tools from libraries (like collecting ingredients), then sets up the basic structure of the model (like prepping your workspace). The core involves layers that analyze images like filters (think of sieves separating flour). The code defines two sets of these filters, one after another, to identify increasingly complex features. Additional layers simplify the analyzed data (like summarizing key details after sifting) and transform it into a more manageable format. A layer then connects all the features, similar to combining ingredients in a recipe. Finally, the code creates an output layer with a specific number of categories (like the number of dishes you can cook). This layer uses a special function to ensure probabilities add up to 1 (like assigning portions). Lastly, the code configures how the model learns by setting parameters (like choosing cooking methods and times) to achieve optimal performance. By following these steps, the code builds a CNN that can learn from image data and classify objects into predefined categories.



```
In [6]:  # fit the model
         r = model.fit_generator(
           training_set,
           validation_data=test_set,
           epochs=20,
           steps_per_epoch=len(training_set),
           validation_steps=len(test_set)
         )

C:\Users\sanja\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py:1844: UserWarning: `Model.fit_generato
r` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
  warnings.warn('`Model.fit_generator` is deprecated and '

Epoch 1/20
80/80 [==============================] - 725s 9s/step - loss: 0.7470 - accuracy: 0.6620 - val_loss: 0.3793 - val_accuracy:
0.8233
Epoch 2/20
80/80 [==============================] - 736s 9s/step - loss: 0.4508 - accuracy: 0.7943 - val_loss: 0.4043 - val_accuracy:
0.8267
Epoch 3/20
80/80 [==============================] - 773s 10s/step - loss: 0.4345 - accuracy: 0.7984 - val_loss: 0.4211 - val_accuracy:
0.8000
Epoch 4/20
80/80 [==============================] - 774s 10s/step - loss: 0.4060 - accuracy: 0.8247 - val_loss: 0.3830 - val_accuracy:
0.8367
Epoch 5/20
80/80 [==============================] - 830s 10s/step - loss: 0.3618 - accuracy: 0.8354 - val_loss: 0.5026 - val_accuracy:
0.8000
Epoch 6/20
80/80 [==============================] - 436s 5s/step - loss: 0.3316 - accuracy: 0.8537 - val_loss: 0.3523 - val_accuracy:
0.8467
Epoch 7/20
80/80 [==============================] - 384s 5s/step - loss: 0.2832 - accuracy: 0.8713 - val_loss: 0.3862 - val_accuracy:
0.8400
Epoch 8/20
80/80 [==============================] - 388s 5s/step - loss: 0.2559 - accuracy: 0.8935 - val_loss: 0.3630 - val_accuracy:
0.8500
Epoch 9/20
80/80 [==============================] - 384s 5s/step - loss: 0.2558 - accuracy: 0.8987 - val_loss: 0.4122 - val_accuracy:
0.8400
Epoch 10/20
80/80 [==============================] - 386s 5s/step - loss: 0.2317 - accuracy: 0.8937 - val_loss: 0.3405 - val_accuracy:
0.8667
Epoch 11/20
80/80 [==============================] - 335s 4s/step - loss: 0.2209 - accuracy: 0.9062 - val_loss: 0.3331 - val_accuracy:
0.8733
Epoch 12/20
80/80 [==============================] - 349s 4s/step - loss: 0.2361 - accuracy: 0.8987 - val_loss: 0.4997 - val_accuracy:
0.7833
Epoch 13/20
80/80 [==============================] - 344s 4s/step - loss: 0.2364 - accuracy: 0.8995 - val_loss: 0.3931 - val_accuracy:
0.8533
Epoch 14/20
```

Figure 5.9: Training VGG 16 model

CHAPTER 6

CHAPTER 6

RESULTS AND DISCUSSION

6.1 Testing

Testing is extremely important for quality assurance and ensuring the products reliability. The success of testing for programmer flaws in largely determined by the experience. Testing might be a crucial component in ensuring the proposed systems quality and efficiency in achieving its goal. Testing is carried out at various phases during the system design and implementation process with the goal of creating a system that is visible, adaptable and secure. Testing is an important element of the software development process. The testing procedure verifies whether the generated product meets the requirements for which it was intended.

6.1.1 Test Objectives

Testing may be a defined as a process of running a programme with the goal of detecting a flaw.

- An honest case is one in which there is a good chance of discovering a mistake that hasn't been detected yet.
- A successful test is one that uncovers previously unknown flaw. If testing is done correctly, problems in the programme will be discovered. Testing cannot reveal whether or not flaws are present. It can only reveal the presence of software flaws.

6.1.2 Testing Principles

A programmer must first grasp the fundamental idea that governs software testing before applying the methodologies to create successful test cases. All testing must be able to be tracked back to the customer's specification.

6.1.3 Testing Design

Any engineering product is frequently put to the test in one of two ways:

- **White Box Testing**

  Glass container checking out is every other call for this kind of checking out. By understanding the necessary characteristic that the product has been supposed to do, checking out is regularly accomplished that proves every characteristic is absolutely operational at the same time as additionally checking for faults in every characteristic.

- **Black Box Testing**

  Tests are regularly finished on this checking out via way of means of understanding the indoors operation of a product to make certain that each one gears mesh, that the indoors operation operates reliably in step with specification, and that each one inner additive had been nicely exercised. It is in most cases worried with the software's practical needs.

6.1.4 Testing Techniques

A software testing template should be established as a set of stages in which particular test suit design techniques are defined for the software engineering process.

The following characteristics should be included in every software testing strategy:

 Testing begins with the modules and extends to the mixing of the full computer-based system.

 At different periods in time, different testing approaches are applicable.

 Testing is carried out by the software's developer and an independent test group. A software developer can use a software testing strategy as a route map. Testing might be a collection of actions that are prepared ahead of time and carried out in a methodical manner. As a result, a software testing template should be established as a set of stages in which particular test suit design techniques are defined for the software engineering process. The following characteristics should be included in every software testing strategy: Testing begins at the module level and progresses to entire computer-based system are mixing.

6.1.5 Levels of Testing

Testing is frequently omitted at various stages of the SDLC. They are as follows:

- **Unit Testing:**

  Unit testing checks the tiny piece of software that makes up the module. The white box orientation of the unit test is maintained throughout. Different modules are tested alongside the requirements created throughout the module design process. The aim of unit testing is to inspect the inner logic of the modules, and it is used to verify the code created during development phase. It is usually done by the module's developer. The coding phase is sometimes referred to as coding and unit testing because of its tight association with coding. Unit tests for many modules are frequently run in simultaneously.

- **Integration Testing:**

  Integration testing is the second level of quality assurance. This type of testing integrates different components in program like modules also to check the interface problems. Many tested modules are combined into subsystems and tested as a result of this. The purpose of this test is to see if all of the modules are properly integrated. Integration testing may be divided into three categories:

- **Top-Down Integration:**

  Top-Down integration is a method of gradually constructing a Programme structures. Modules are connected by working their way down the control Hierarchy, starting with the module having the most control. Bottom-Up Integration:

  Construction and testing using autonomous modules begin with Bottom-up integration, as the name suggests.

- **Regression Testing:**

  It is a subset of previously executed tests to ensure that Modifications have not propagated unexpected side effects during this competition of an Integration test strategy.

- **Functional Testing:**

  The business and technical requirements, system documentation, and user guides all specify that functional tests must be conducted to ensure that the functions being tested are available. The following items are the focus of functional testing:

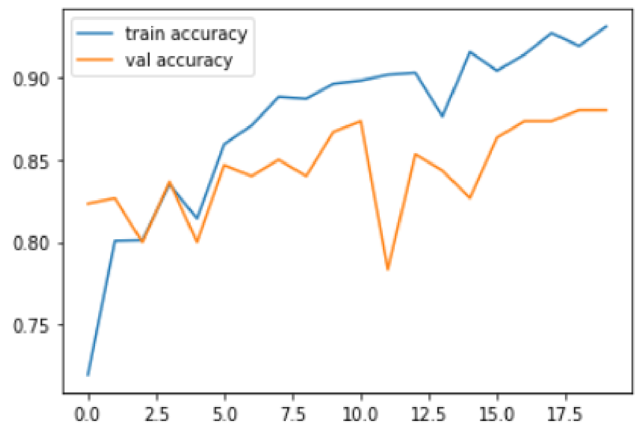
- **Validation Testing:**

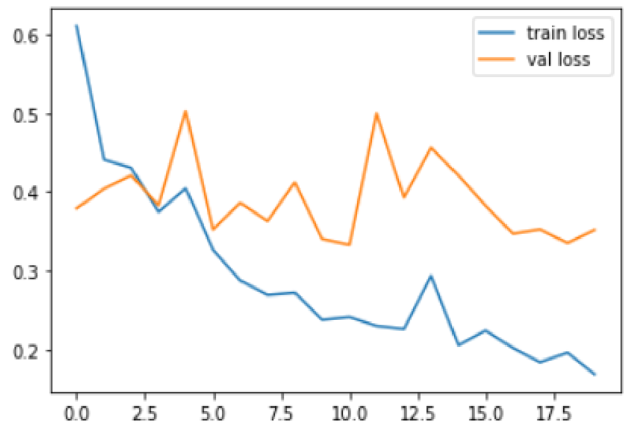
  Validation may be characterized in a lot of ways; however, one easy definition is that validation is a hit whilst software program plays in a manner that clients may fairly expect. The affordable expectation is said with inside the software program requirement specification that is a record that lists all the software program’s user-seen attributes. Validation standards are a segment of the specification. The statistics on this component serves as the premise for the validation trying out strategy.

- **Alpha Testing:**

  Software developer can’t know how a customer will utilize a programme ahead of time. Instructions to be utilized could be misconstrued, a peculiar combination of knowledge could be employed on a regular basis, and a result that was clear to the tester could be unclear to a field user. It's impractical to conduct a formal acceptance test with all users if the programmed is designed as a product that will be used by many people. Most software developers utilize alpha and beta testing to detect bugs that only the most experienced users seem to be aware of. At the developer's premises, a customer does the trial.

6.2 Results





```
<Figure size 432x288 with 0 Axes>
```

Figure 6.1: Training accuracy v/s Validation accuracy and Training loss v/s Validation loss in Image Classification model

The graph offers insights into how a machine learning model is learning. The x-axis tracks training epochs, which are essentially rounds of training where the model sees the entire training data set. The y-axis represents accuracy, indicating how well the model performs its task. Two lines are plotted: training accuracy and validation accuracy. As expected, the training accuracy typically increases as the model is exposed to more data during training.

The validation accuracy, however, reflects how well the model performs on unseen data. A significant gap between these lines indicates overfitting, where the model memorizes the training data but struggles with new information. Imagine a student who aces a practice test by rote memorization but might fail when encountering new questions. By monitoring both accuracies, developers can identify and address overfitting to ensure the model generalizes well to real-world scenarios. In essence, this graph is a valuable tool for evaluating a machine learning model's learning progress and ability to handle unseen data.
By analyzing this graph, we can assess the model's learning progress and identify potential issues like overfitting. Here are some key observations:

- **Upward Trends:** If both training and validation accuracy exhibit upward trends, it suggests the model is learning effectively from the training data.
- **Convergence:** Ideally, the validation accuracy should converge towards a stable value after a sufficient number of epochs. This indicates the model has achieved a good balance between learning from the training data and generalizing to unseen data.
- **Gap Analysis:** A small gap between the training and validation accuracy lines is desirable. A large and persistent gap suggests potential overfitting, requiring adjustments to the model or training process to improve generalization.

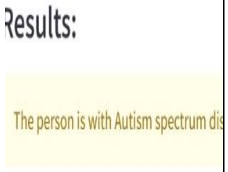
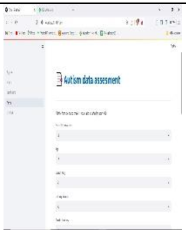
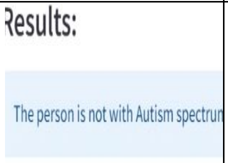
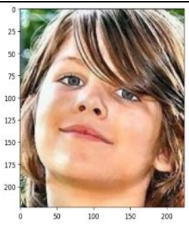
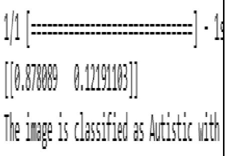
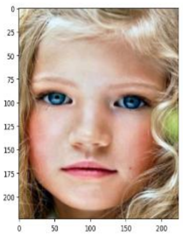
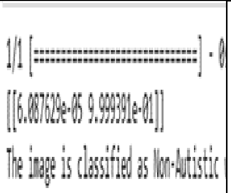
Test Case ID	**Test case Objective**	**Prerequisites**	**Steps**	**Input Data**	**Expected Output**	**Actual Output**	**Status**
1	Input the text	Read text	Enter Number	Autism data assesment	Autism Detected	 Results: The person is with Autism spectrum dis	Passed
2	Input the text	Read text	Enter Number	 Autism data assesment	Autism not Detected	 Results: The person is not with Autism spectrum	Passed
3	Input the image	Read image	Upload image		Autism Detected	 1/1 [==============================] - 1s [[0.878089 0.12191103]] The image is classified as Autistic with	Passed
4	Input the image	Read image	Upload image		Autism not Detected	 1/1 [==============================] - 0 [[6.087629e-05 9.999391e-01]] The image is classified as Non-Autistic	Passed

Figure 6.2: Test Cases of Output

6.3 Snapshots

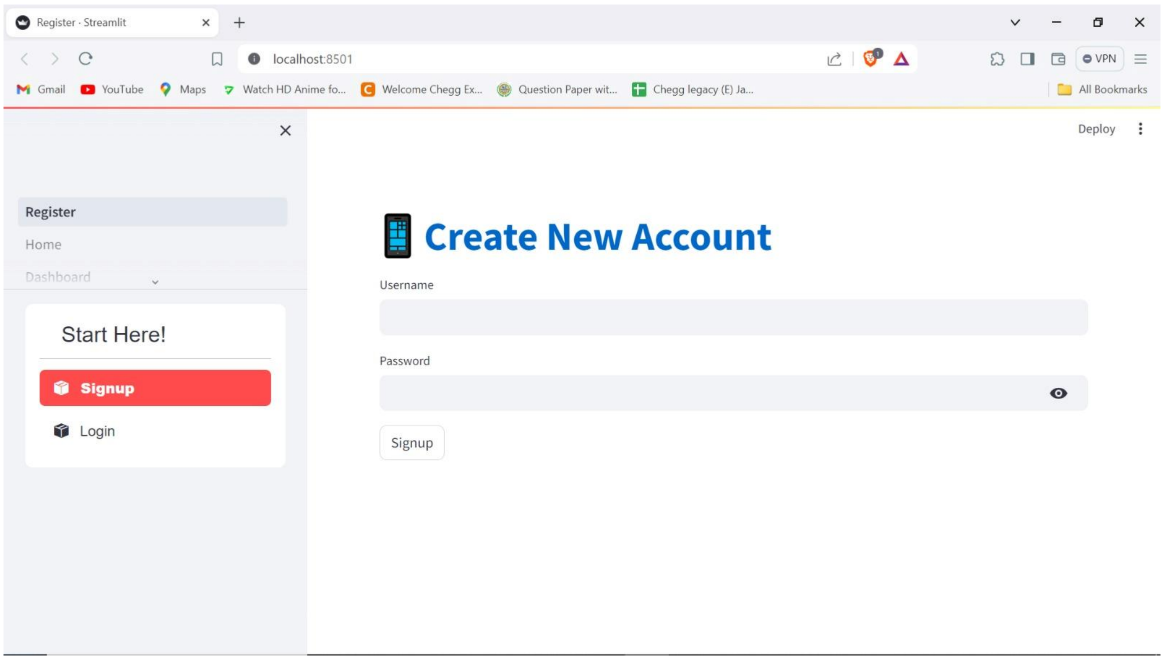


Figure 6.3: Create Account

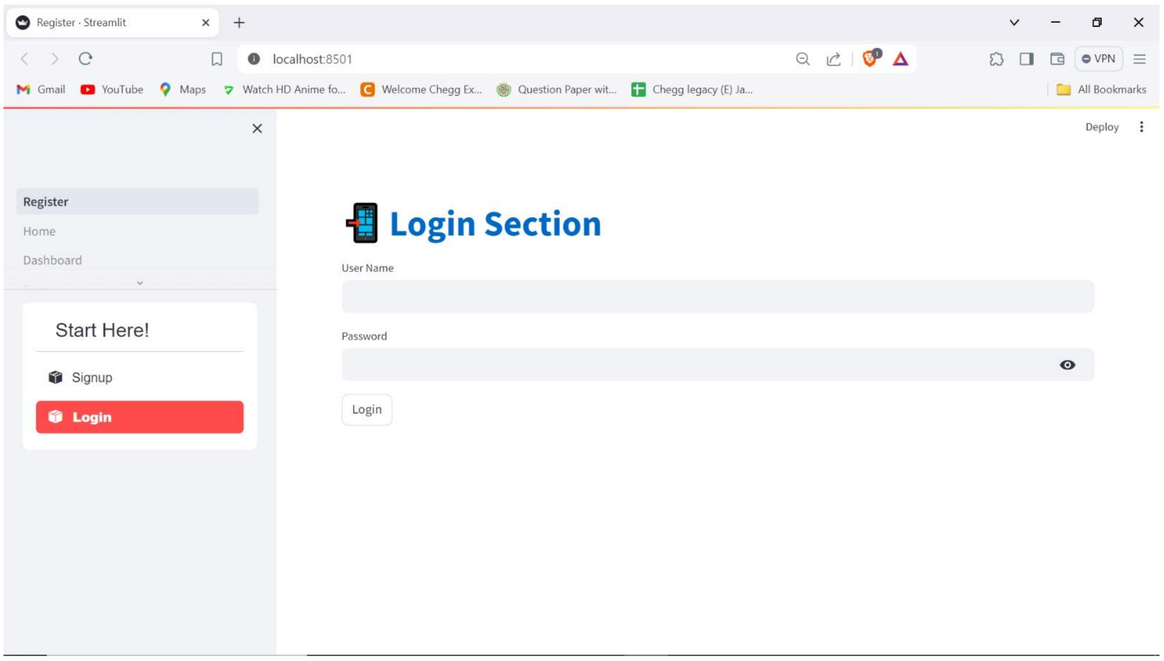
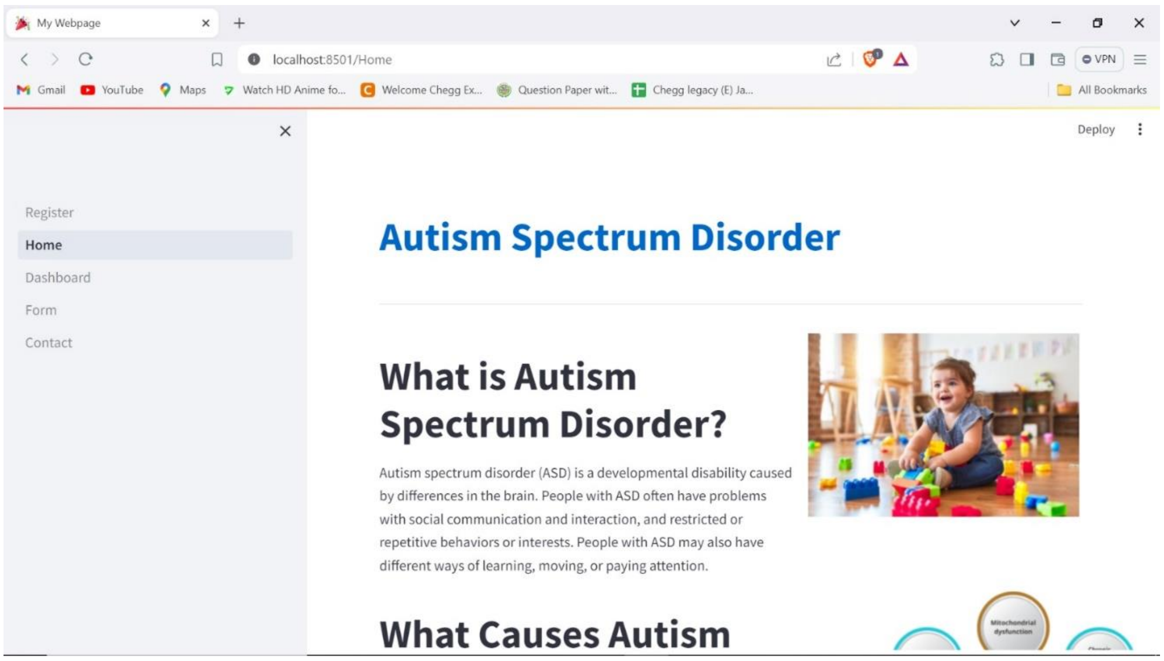


Figure 6.4: Login to Account

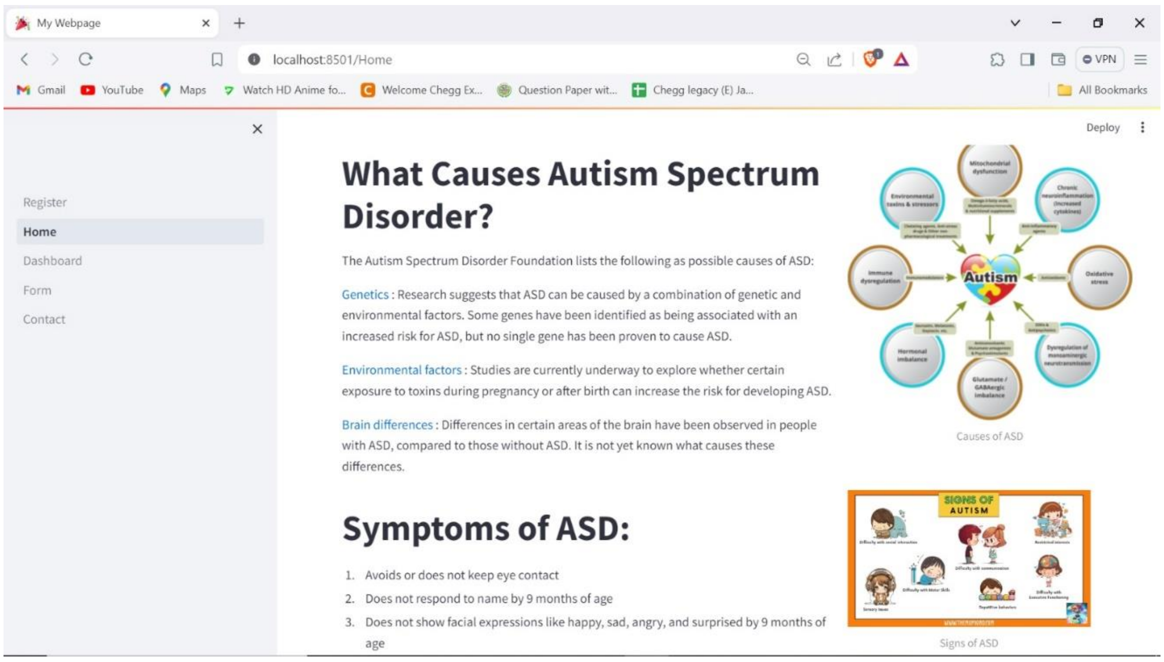


Autism Spectrum Disorder

What is Autism Spectrum Disorder?

Autism spectrum disorder (ASD) is a developmental disability caused by differences in the brain. People with ASD often have problems with social communication and interaction, and restricted or repetitive behaviors or interests. People with ASD may also have different ways of learning, moving, or paying attention.

What Causes Autism



What Causes Autism Spectrum Disorder?

The Autism Spectrum Disorder Foundation lists the following as possible causes of ASD:

Genetics : Research suggests that ASD can be caused by a combination of genetic and environmental factors. Some genes have been identified as being associated with an increased risk for ASD, but no single gene has been proven to cause ASD.

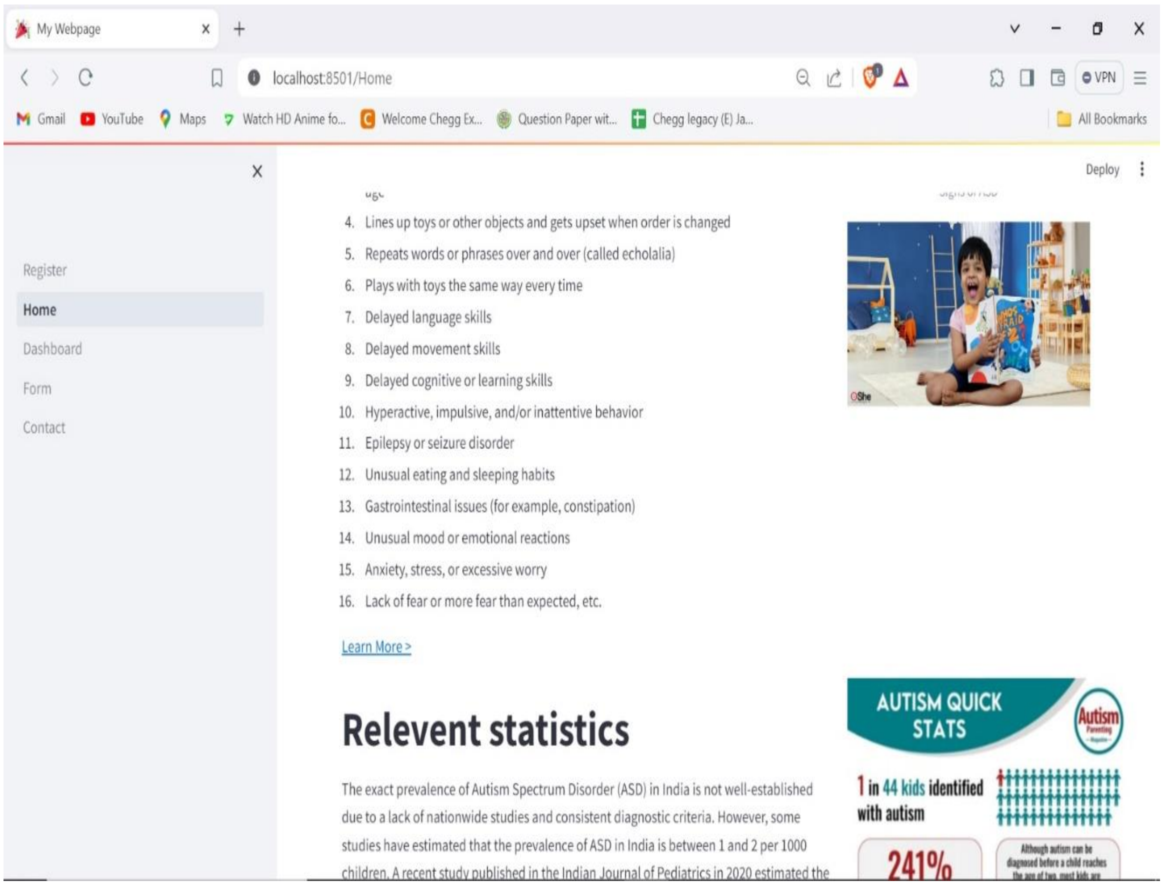
Environmental factors : Studies are currently underway to explore whether certain exposure to toxins during pregnancy or after birth can increase the risk for developing ASD.

Brain differences : Differences in certain areas of the brain have been observed in people with ASD, compared to those without ASD. It is not yet known what causes these differences.

Symptoms of ASD:

1. Avoids or does not keep eye contact
2. Does not respond to name by 9 months of age
3. Does not show facial expressions like happy, sad, angry, and surprised by 9 months of age

Figure 6.5: Description on ASD

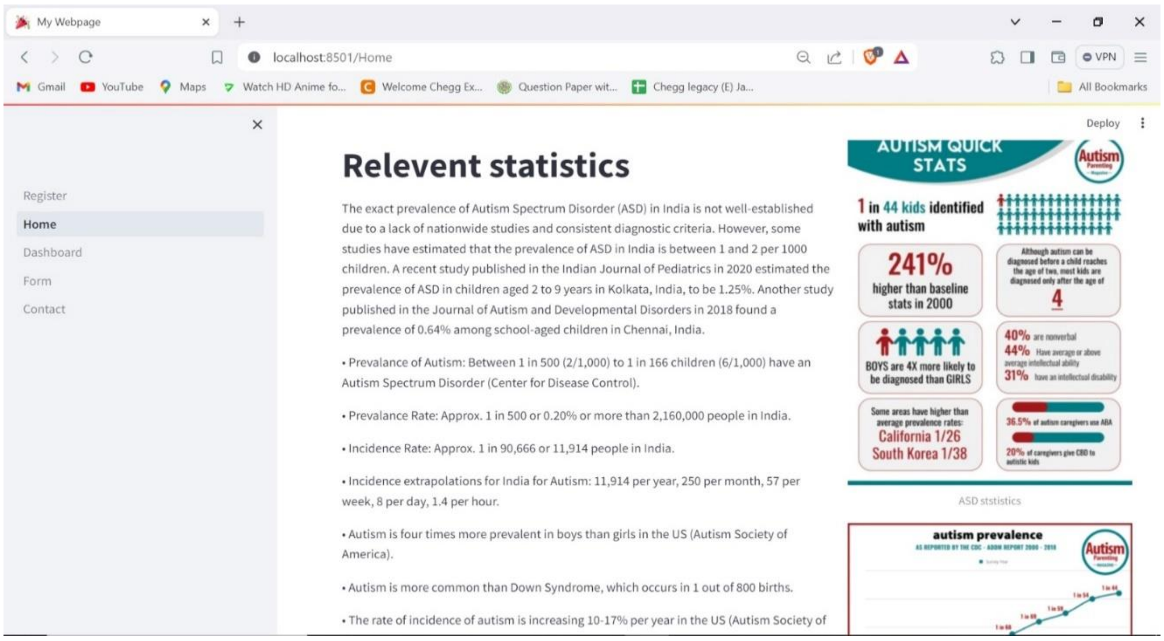


4. Lines up toys or other objects and gets upset when order is changed
5. Repeats words or phrases over and over (called echolalia)
6. Plays with toys the same way every time
7. Delayed language skills
8. Delayed movement skills
9. Delayed cognitive or learning skills
10. Hyperactive, impulsive, and/or inattentive behavior
11. Epilepsy or seizure disorder
12. Unusual eating and sleeping habits
13. Gastrointestinal issues (for example, constipation)
14. Unusual mood or emotional reactions
15. Anxiety, stress, or excessive worry
16. Lack of fear or more fear than expected, etc.

Learn More >

Relevent statistics

The exact prevalence of Autism Spectrum Disorder (ASD) in India is not well-established due to a lack of nationwide studies and consistent diagnostic criteria. However, some studies have estimated that the prevalence of ASD in India is between 1 and 2 per 1000 children. A recent study published in the Indian Journal of Pediatrics in 2020 estimated the


My Webpage
localhost:8501/Home
Gmail
YouTube
Maps
Watch HD Anime fo...
Welcome Chegg Ex...
Question Paper wit...
Chegg legacy (E) Ja...
All Bookmarks
Deploy
Register
Home
Dashboard
Form
Contact
AUTISM QUICK STATS
1 in 44 kids identified with autism
241% higher than baseline stats in 2000
Although autism can be diagnosed before a child reaches the age of two, most kids are diagnosed only after the age of 4
BOYS are 4X more likely to be diagnosed than GIRLS
40% are nonverbal
44% Have average or above average intellectual ability
31% have an intellectual disability
Some areas have higher than average prevalence rates: California 1/26 South Korea 1/38
36.5% of autism caregivers use ABA
20% of caregivers give CBD to autistic kids
ASD ststistics
autism prevalence
AS REPORTED BY THE CDC - ADDM REPORT 2000 - 2018
1 in 68
1 in 69
1 in 59
1 in 54
1 in 44

Relevent statistics

The exact prevalence of Autism Spectrum Disorder (ASD) in India is not well-established due to a lack of nationwide studies and consistent diagnostic criteria. However, some studies have estimated that the prevalence of ASD in India is between 1 and 2 per 1000 children. A recent study published in the Indian Journal of Pediatrics in 2020 estimated the prevalence of ASD in children aged 2 to 9 years in Kolkata, India, to be 1.25%. Another study published in the Journal of Autism and Developmental Disorders in 2018 found a prevalence of 0.64% among school-aged children in Chennai, India.

• Prevalance of Autism: Between 1 in 500 (2/1,000) to 1 in 166 children (6/1,000) have an Autism Spectrum Disorder (Center for Disease Control).

• Prevalance Rate: Approx. 1 in 500 or 0.20% or more than 2,160,000 people in India.

• Incidence Rate: Approx. 1 in 90,666 or 11,914 people in India.

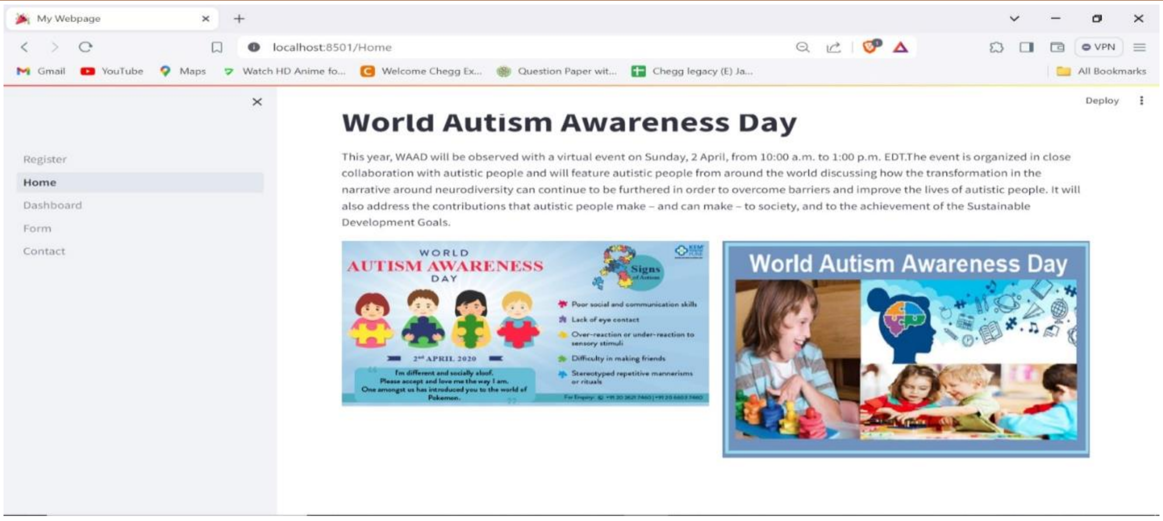
• Incidence extrapolations for India for Autism: 11,914 per year, 250 per month, 57 per week, 8 per day, 1.4 per hour.

• Autism is four times more prevalent in boys than girls in the US (Autism Society of America).

• Autism is more common than Down Syndrome, which occurs in 1 out of 800 births.

• The rate of incidence of autism is increasing 10-17% per year in the US (Autism Society of

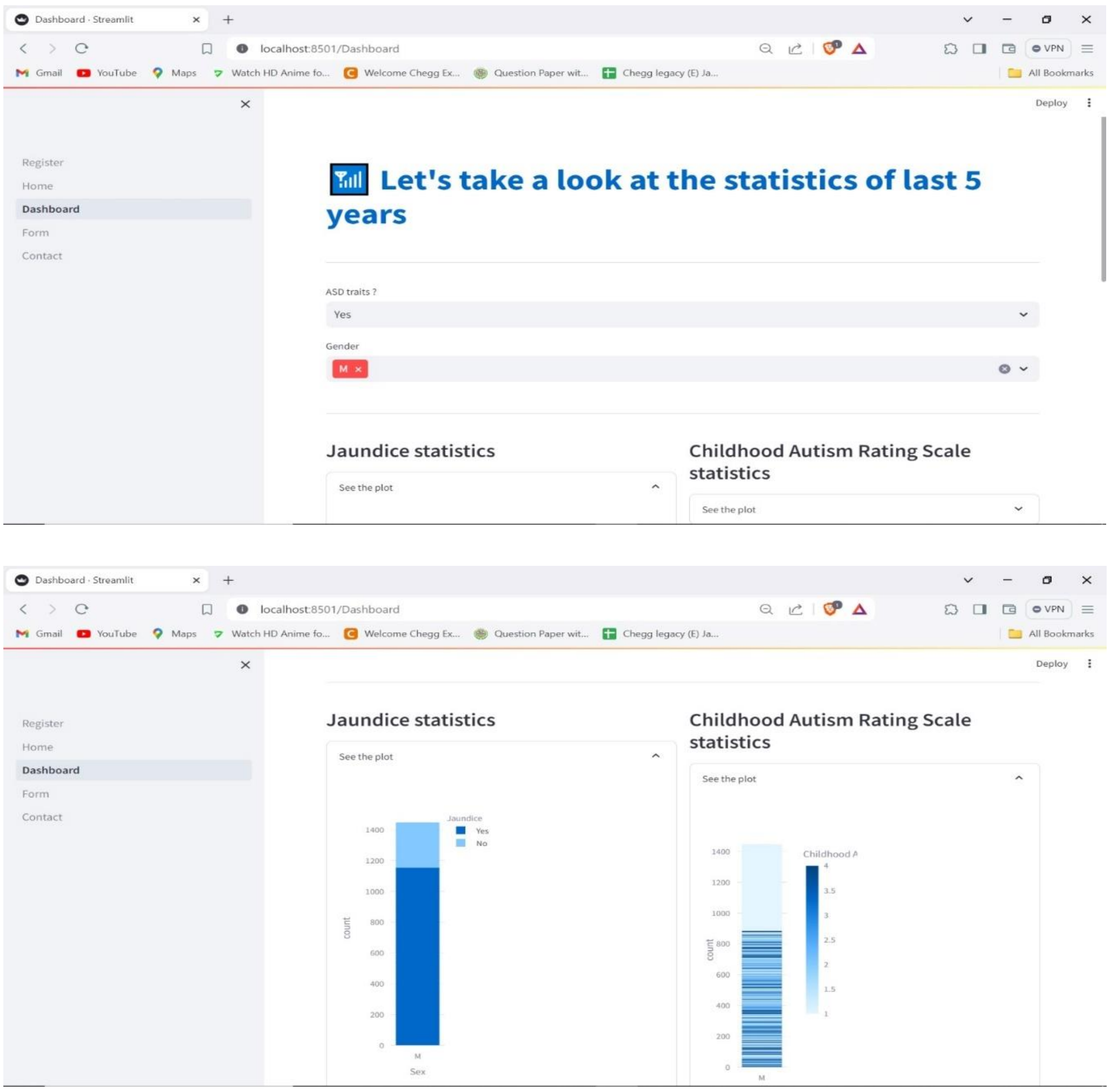
Figure 6.6: Description on Autism



World Autism Awareness Day

This year, WAAD will be observed with a virtual event on Sunday, 2 April, from 10:00 a.m. to 1:00 p.m. EDT.The event is organized in close collaboration with autistic people and will feature autistic people from around the world discussing how the transformation in the narrative around neurodiversity can continue to be furthered in order to overcome barriers and improve the lives of autistic people. It will also address the contributions that autistic people make – and can make – to society, and to the achievement of the Sustainable Development Goals.

Figure 6.7: Description on ASD



Let's take a look at the statistics of last 5 years

ASD traits ?

Yes

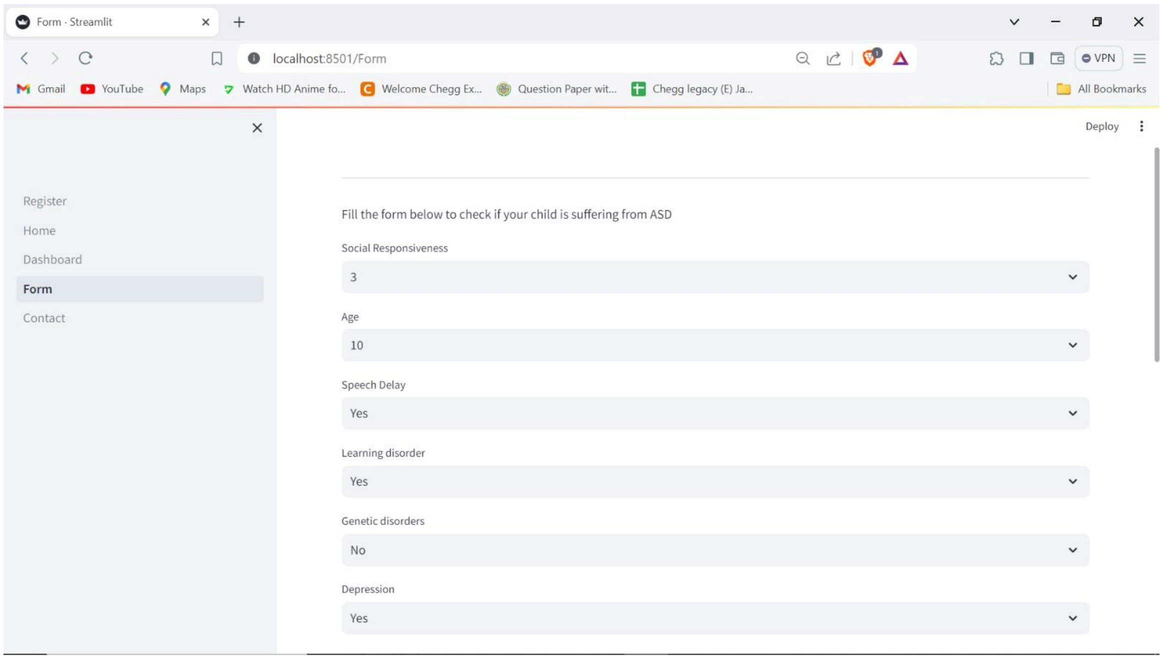
Gender

M ×

Jaundice statistics

Childhood Autism Rating Scale statistics

Figure 6.8: ASD Statistics



Fill the form below to check if your child is suffering from ASD

Social Responsiveness: 3

Age: 10

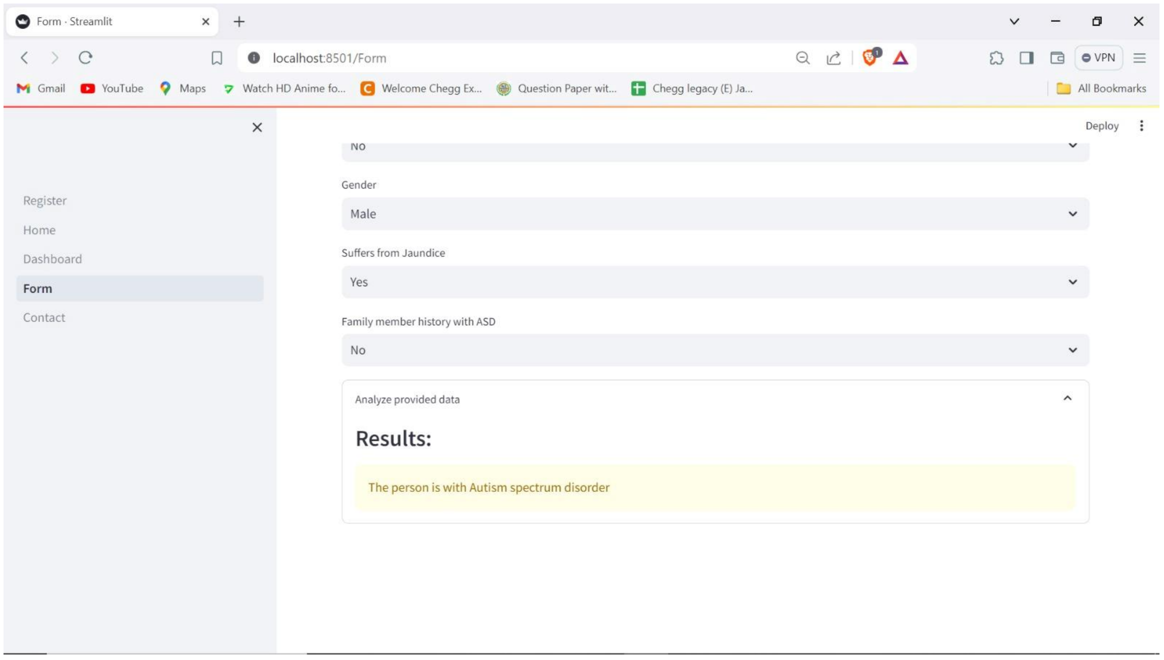
Speech Delay: Yes

Learning disorder: Yes

Genetic disorders: No

Depression: Yes

Figure 6.9: Predictive Analysis Form



No

Gender: Male

Suffers from Jaundice: Yes

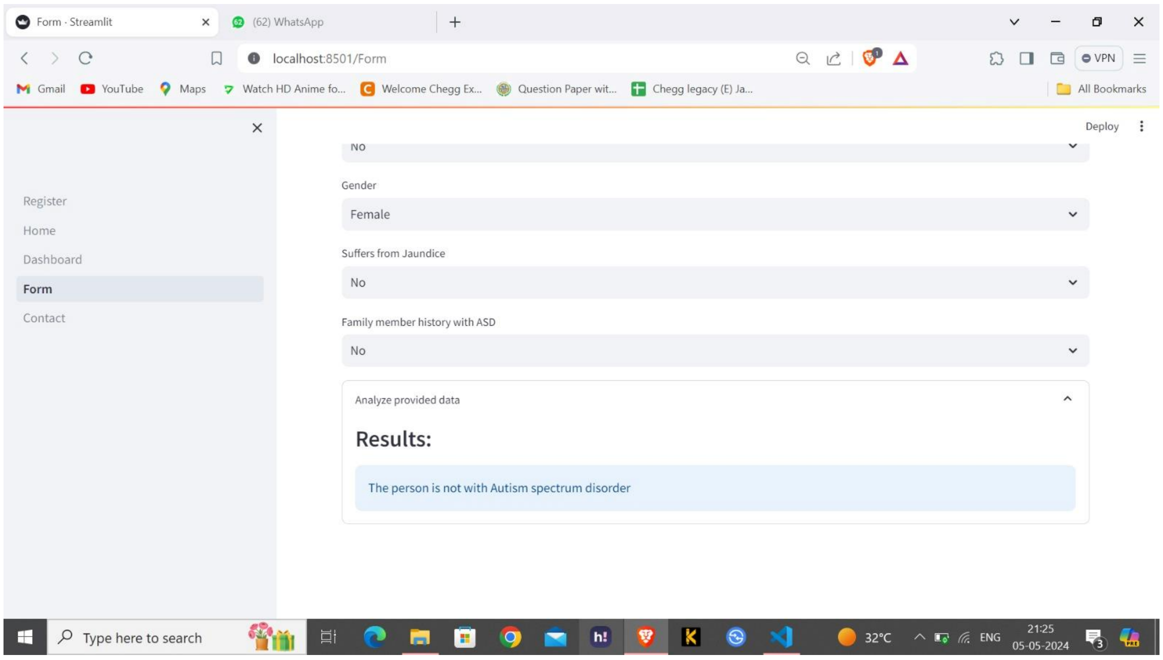
Family member history with ASD: No

Analyze provided data

Results:

The person is with Autism spectrum disorder

Figure 6.10: Predictive Analysis Output - 1



No

Gender

Female

Suffers from Jaundice

No

Family member history with ASD

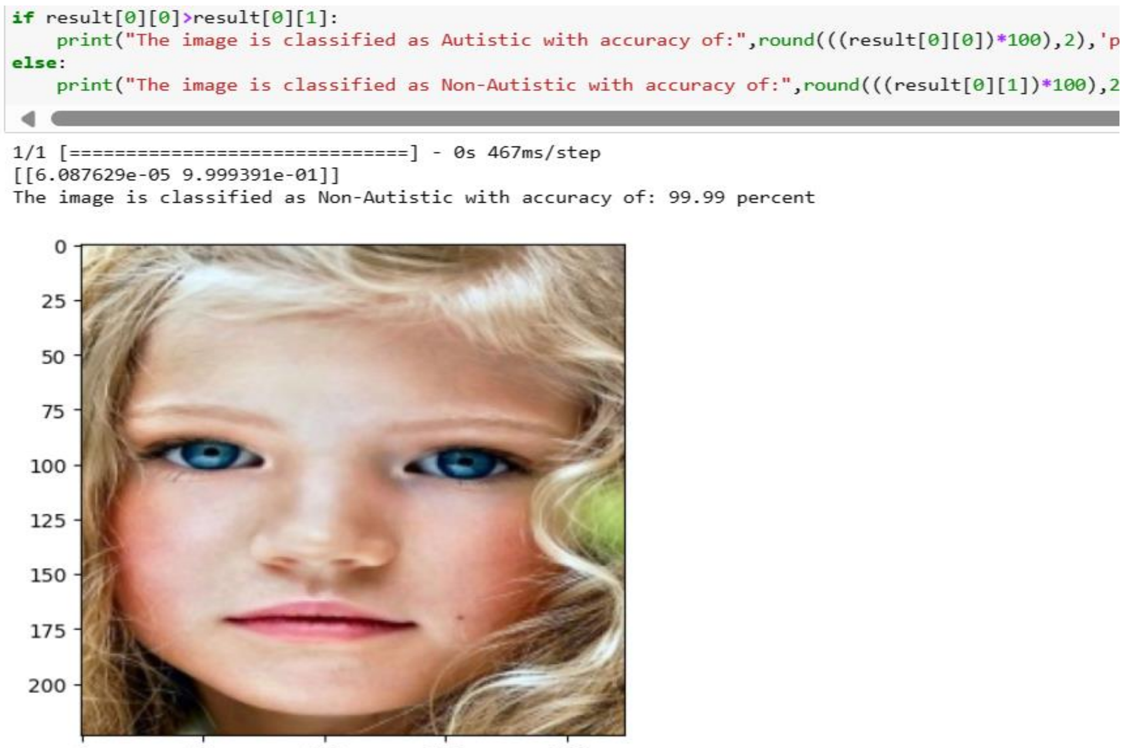
No

Analyze provided data

Results:

The person is not with Autism spectrum disorder

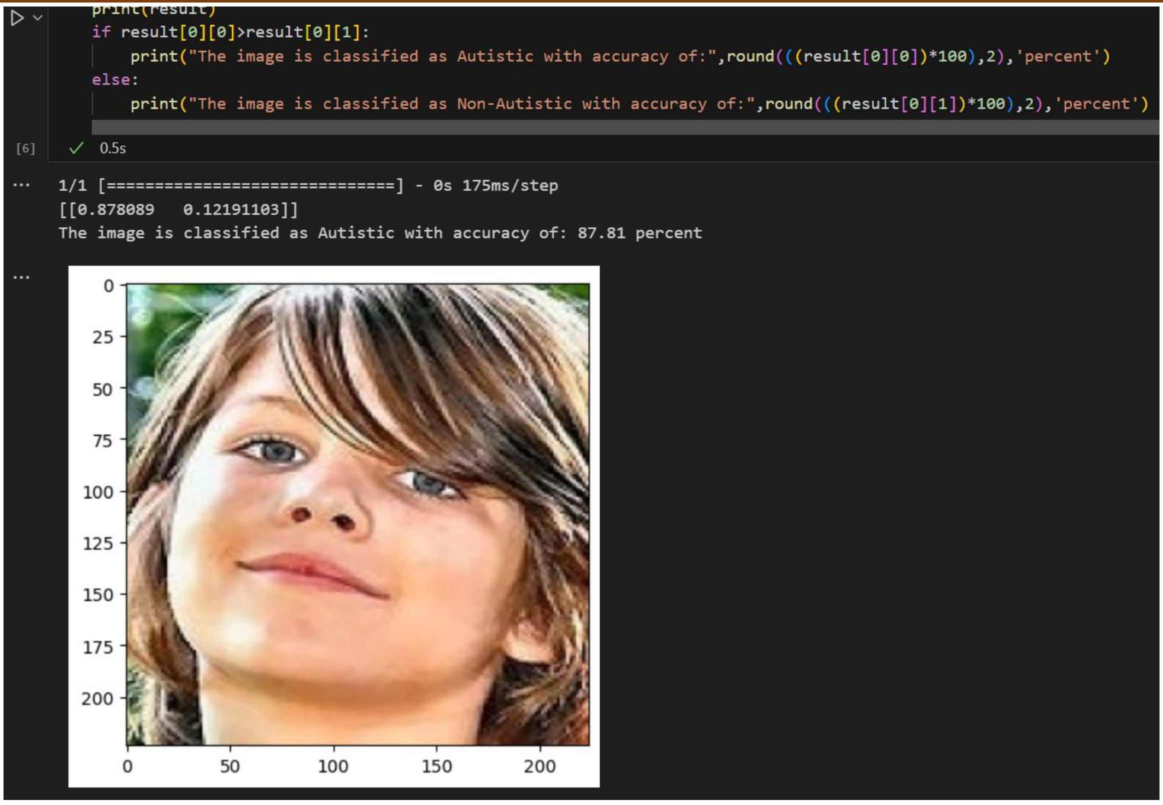
Figure 6.11: Predictive Analysis Output - 2



```
if result[0][0]>result[0][1]:
    print("The image is classified as Autistic with accuracy of:",round(((result[0][0])*100),2),'p
else:
    print("The image is classified as Non-Autistic with accuracy of:",round(((result[0][1])*100),2
```

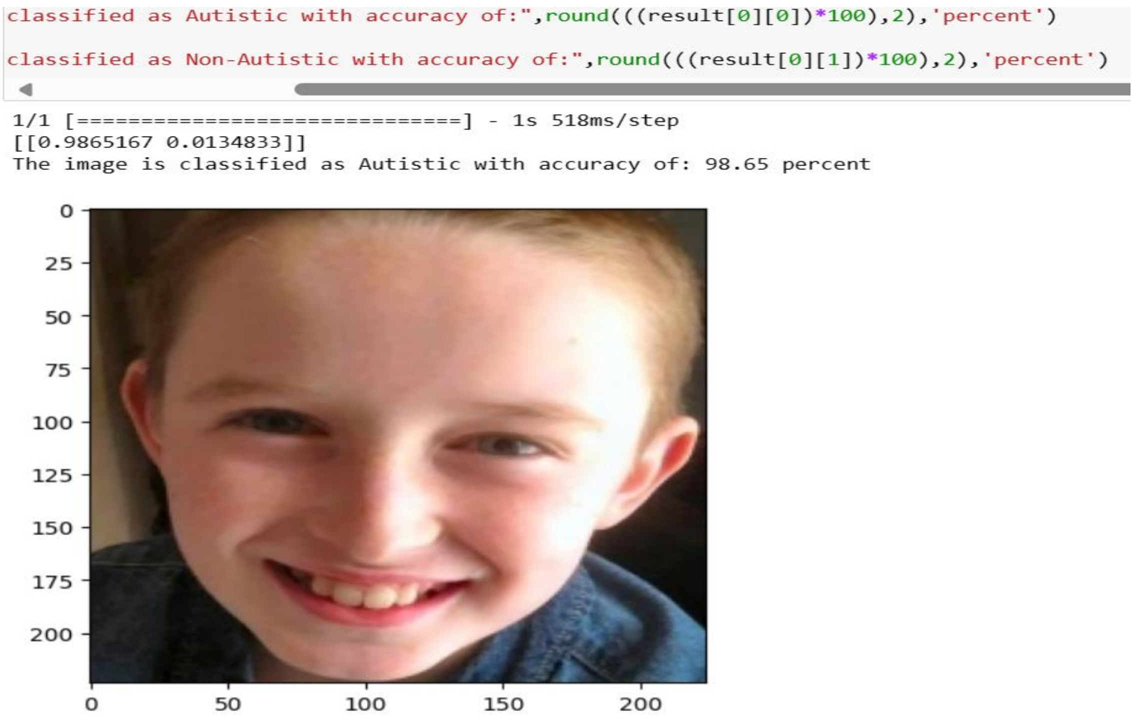
```
1/1 [==============================] - 0s 467ms/step
[[6.087629e-05 9.999391e-01]]
The image is classified as Non-Autistic with accuracy of: 99.99 percent
```

Figure 6.12: Image Recognition Output - 1



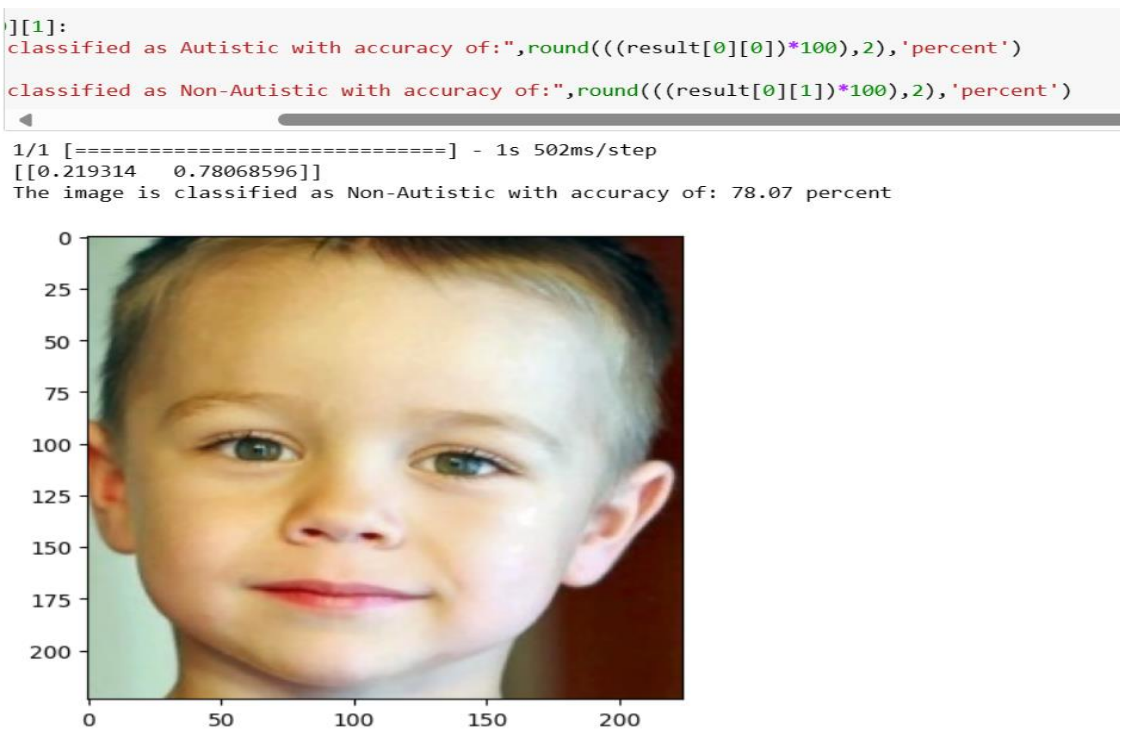
```
    print(result)
    if result[0][0]>result[0][1]:
        print("The image is classified as Autistic with accuracy of:",round(((result[0][0])*100),2),'percent')
    else:
        print("The image is classified as Non-Autistic with accuracy of:",round(((result[0][1])*100),2),'percent')
[6]  ✓ 0.5s

1/1 [==============================] - 0s 175ms/step
[[0.878089   0.12191103]]
The image is classified as Autistic with accuracy of: 87.81 percent
```

Figure 6.13: Image Recognition Output - 2



```
classified as Autistic with accuracy of:",round(((result[0][0])*100),2),'percent')

classified as Non-Autistic with accuracy of:",round(((result[0][1])*100),2),'percent')

1/1 [==============================] - 1s 518ms/step
[[0.9865167 0.0134833]]
The image is classified as Autistic with accuracy of: 98.65 percent
```

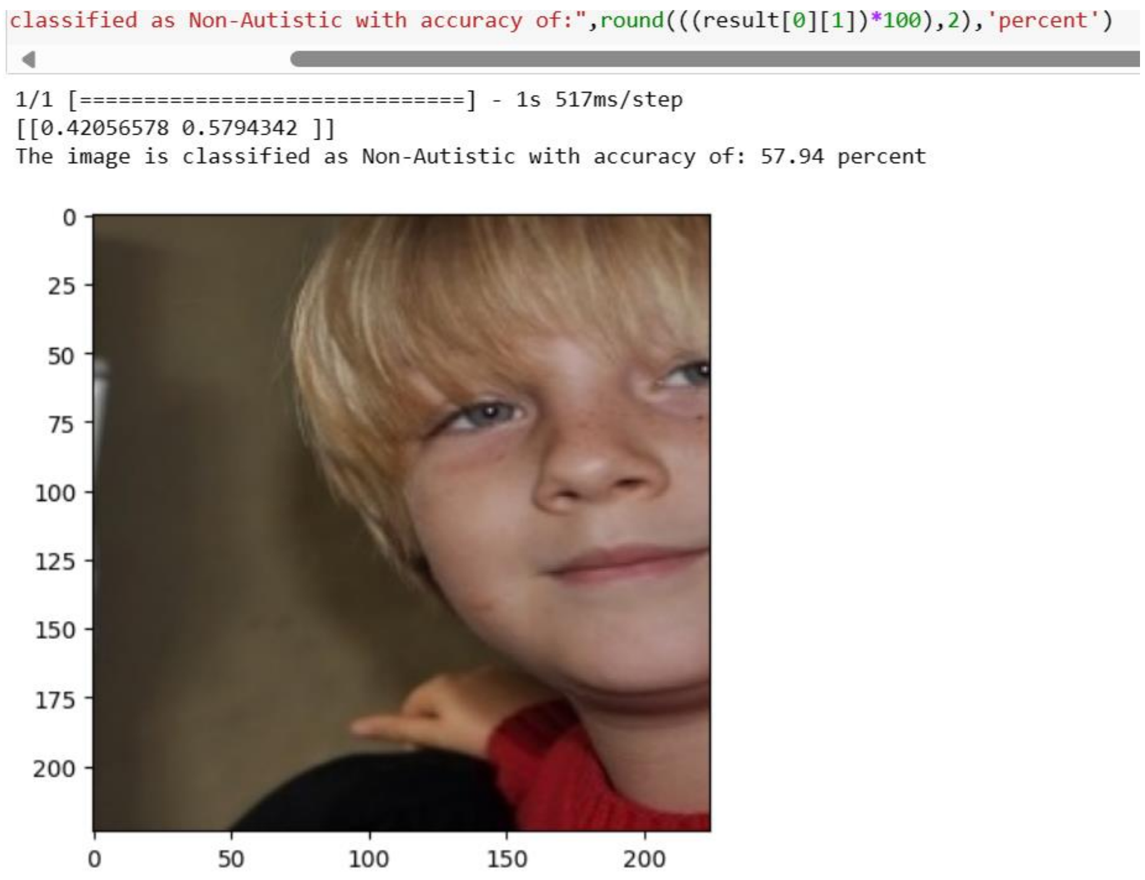
Figure 6.14: Image Recognition Output - 3



```
)][1]:
classified as Autistic with accuracy of:",round(((result[0][0])*100),2),'percent')

classified as Non-Autistic with accuracy of:",round(((result[0][1])*100),2),'percent')
```

```
1/1 [==============================] - 1s 502ms/step
[[0.219314   0.78068596]]
The image is classified as Non-Autistic with accuracy of: 78.07 percent
```

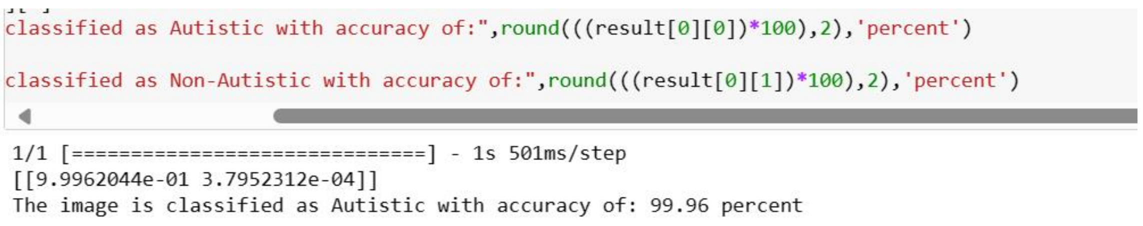
Figure 6.15: Image Recognition Output – 4



```
classified as Non-Autistic with accuracy of:",round(((result[0][1])*100),2),'percent')
```

```
1/1 [==============================] - 1s 517ms/step
[[0.42056578 0.5794342 ]]
The image is classified as Non-Autistic with accuracy of: 57.94 percent
```

Figure 6.16: Image Recognition Output - 5



```
classified as Autistic with accuracy of:",round(((result[0][0])*100),2),'percent')

classified as Non-Autistic with accuracy of:",round(((result[0][1])*100),2),'percent')
```

```
1/1 [==============================] - 1s 501ms/step
[[9.9962044e-01 3.7952312e-04]]
The image is classified as Autistic with accuracy of: 99.96 percent
```

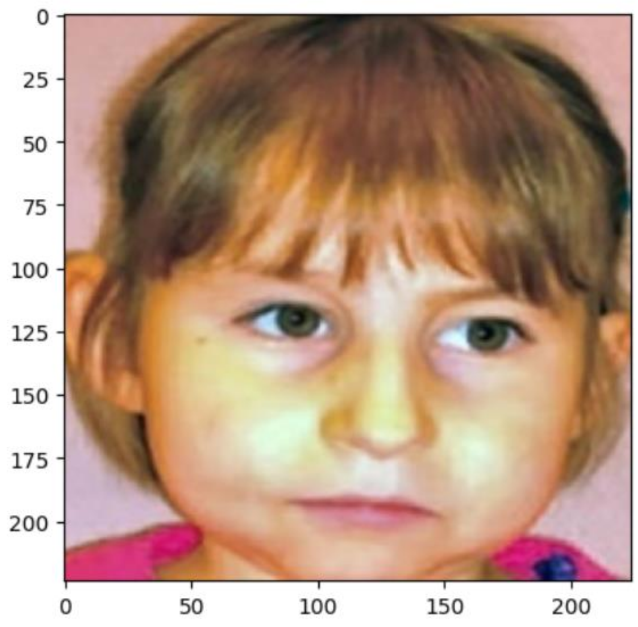


Figure 6.17: Image Recognition Output - 6

CONCLUSION AND FUTURE WORK

In this study, multiple machine learning and deep learning approaches were used to try and detect autism spectrum disorder. The effectiveness of the models employed for ASD identification on non-clinical datasets from three sets of age groups, namely children, adolescents, and adults, was examined using a variety of performance evaluation measures. Our project represents a significant step towards enhancing the detection and management of Autism Spectrum Disorder (ASD) through the utilization of machine learning techniques.

By addressing the limitations of existing diagnostic methods, we strive to provide a more objective, scalable approach to ASD detection. Through the development of this innovative system, we aim to improve outcomes for individuals with ASD, and ultimately contribute to a better understanding and support system for those on the autism spectrum.

Future researchers in this field may apply the resampling techniques to the respective datasets being used. This technique helps to reduce the imbalance ratio of datasets which in turn produces better classification results.

All the algorithms performed almost same with less difference, but we can consider that if these algorithms are trained with some more real-world data then the efficiency and prediction will increase. As we could not reach 100% accuracy even after making many data mining techniques, but we are trying to get more accuracy we will work on combining different algorithms which can give us better accuracy and with respect to data also if we get more data, we can gain more accuracy as learning increases, we will try to decrease False Negatives. We can be still able to get better results if train models with more data and use genetic algorithms which we will test in our future work. For the Image Classification, as for the future enhancement we can deploy the built vgg16 model to the web so that user will be comfortable in using it.

REFERENCES

[1] American Psychiatric Association. (2013). Diagnostic and statistical manual of mental disorders (5th ed.). Arlington, VA: American Psychiatric Publishing.

[2] Baio, J. (2014). Prevalence of Autism Spectrum Disorder Among Children Aged 8 Years — Autism and Developmental Disabilities Monitoring Network, 11 Sites, United States, 2010. MMWR. Surveillance Summaries, 63(2), 1–21.

[3] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, 321–357.

[4] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[5] Lord, C., Elsabbagh, M., Baird, G., & Veenstra-Vanderweele, J. (2018). Autism spectrum disorder. The Lancet, 392(10146), 508–520.

[6] Miodovnik, A., Harstad, E., Sideridis, G., & Huntington, N. (2015). Timing of the Diagnosis of Attention-Deficit/Hyperactivity Disorder and Autism Spectrum Disorder. Pediatrics, 136(4), e830–e837.

[7] Thabtah, F. (2020). Machine Learning in Autistic Spectrum Disorder Behavioral Research: A Review and Ways Forward. Informatics, 7(4), 52.

[8] Thabtah, F., & Peebles, D. (2018). Detecting Autism Spectrum Disorder Using Extreme Learning Machines with Subjective Questionnaire Data. Neurocomputing, 275, 3051–3059.

[9] Thabtah, F., & Kamalov, F. (2019). ASD-Fuse: Autism Spectrum Disorder Diagnosis from Imbalanced EEG Data Using Hybrid Deep Learning Models.

Computers in Biology and Medicine, 107, 197–210.

[10] Zhou, T., Chen, H., Liu, W., Peng, C., & Liu, C. (2020). Convolutional Neural Networks-Based Autism Spectrum Disorder Classification: Feature Importance Analysis. IEEE Access, 8, 219212–219224.

Name: Akshay Kumar P **USN: 1EE20CS001** **SEM: 8** **Phone No.: 9611436560** **Email ID: akshaypreeths@gmail.com**	**Name: Basavaraj Angadi** **USN:1EE20CS005** **SEM:8** **Phone No.:7411946909** **Email ID: bhargav.angadi@gmail.com**
Name: Darshan Gowda C **USN: 1EE20CS011** **SEM: 8** **Phone No.: 9380988058** **Email ID: darshanc4696@gmail.com**	**Name: Harish S** **USN: 1EE20CS014** **SEM: 8** **Phone No.: 6361718521** **Email ID: acharyaisback746@gmail.com**