Detection Of Autistic Spectrum Disorder: Classification

Introduction:
Autism Spectrum Disorder (ASD) is a chronic condition that significantly influences an individual's behavior and socialization skills. While it manifests in early childhood, it often goes undiagnosed until school age. Early detection of ASD is crucial for both families and affected children. This study aims to explore the impact of individual characteristics on ASD detection and assess whether these characteristics can effectively predict ASD cases.

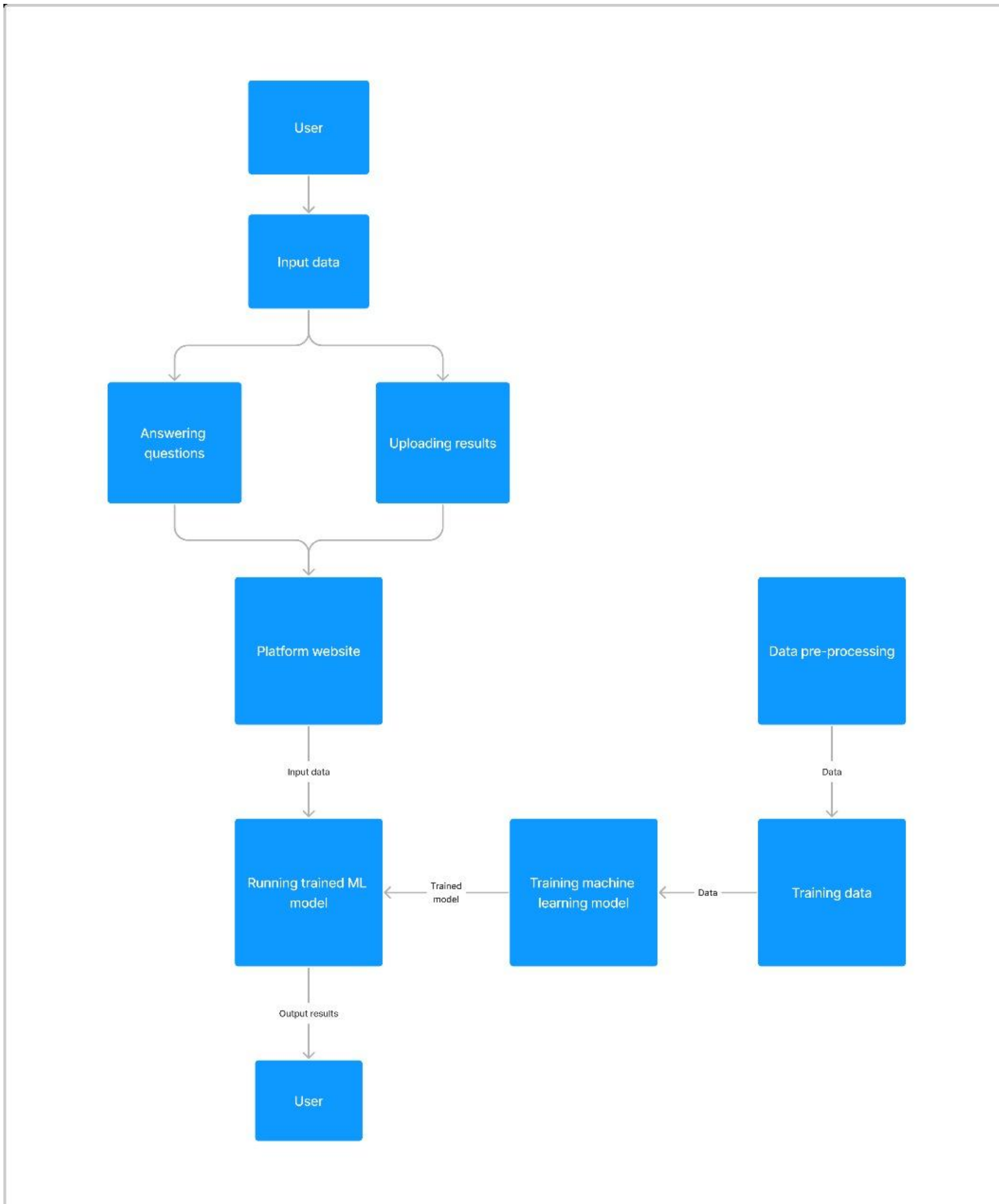
Healthcare Costs and Timely Diagnosis:
ASD carries substantial healthcare costs, emphasizing the importance of early diagnosis in reducing these financial burdens. Unfortunately, the existing waiting times for ASD diagnoses are prolonged, and current procedures lack cost-effectiveness. The economic implications of autism, coupled with the rising number of ASD cases worldwide, underscore the urgent need for easily implementable and efficient screening methods.

Challenges in Dataset Availability:
The increasing prevalence of ASD worldwide highlights the necessity for datasets related to behavioral traits. However, acquiring such datasets is challenging, as they are scarce, hindering in-depth analyses to enhance the efficiency, sensitivity, specificity, and predictive accuracy of ASD screening. Currently, the available autism datasets are primarily genetic in nature, limiting comprehensive exploration.

Proposed Solution:
In response to the shortage of suitable datasets, we propose a new dataset specifically focused on autism screening in adults. This dataset comprises 20 features, including ten behavioral features (A1-A10-Adult) and ten individual characteristics. The goal is to utilize this dataset for further analysis, specifically in identifying influential autistic traits and enhancing the classification of ASD cases.

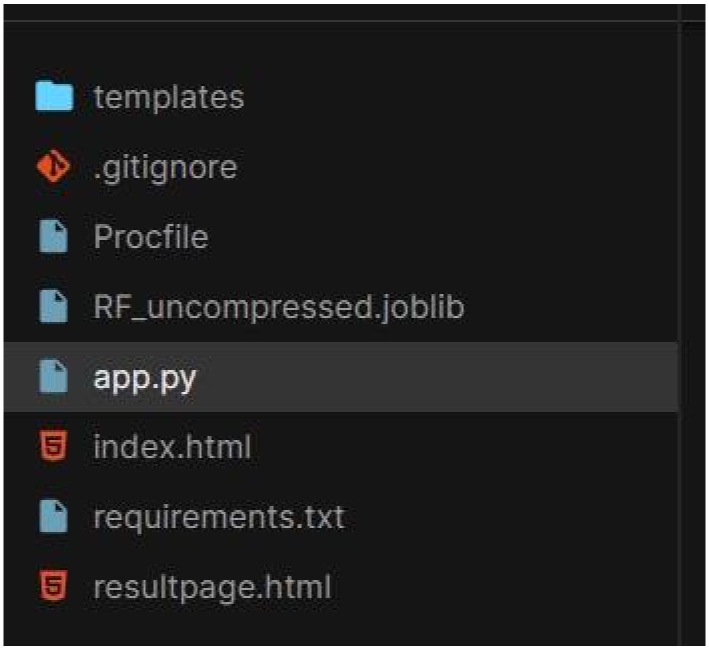
By addressing these aspects, the study aims to contribute to the development of a time-efficient and accessible ASD screening tool. Such a tool would aid healthcare professionals in making informed decisions and provide individuals with valuable insights on whether to pursue formal clinical diagnosis.

TECHNICAL ARCHITECTURE:



PROJECT FLOW:

Website flow



Model flow

Table of contents

Data preprocessing is a crucial step in preparing raw data for analysis or machine learning. It involves several tasks to clean, organize, and transform the data into a format suitable for further processing. Here are some common steps in data preprocessing:

1. Exploratory Data Analysis (EDA):
   - Conducting a thorough exploration of the dataset to understand its characteristics.
   - Visualizing data distributions, relationships, and patterns using charts and graphs.
   - Identifying outliers, trends, and potential insights that guide further preprocessing decisions.
   - EDA helps in making informed choices during data cleaning, transformation, and feature engineering.
   - Techniques such as summary statistics, histograms, scatter plots, and correlation analysis are commonly employed in EDA.

Certainly! Here's an added point emphasizing visualization in the data preprocessing phase:

2. Data Visualization:
   - Conducting visual exploration: Utilizing charts, graphs, and plots to visually understand the distribution of data, spot outliers, and identify trends during EDA.
   - Visualizing relationships: Creating scatter plots, pair plots, and correlation matrices to visually assess the relationships between variables, aiding in feature selection and engineering decisions.

   - Insightful data representation: Utilizing visualizations to communicate findings and insights to stakeholders, facilitating a better understanding of the dataset's nuances.
   - Visual diagnostics: Using visualizations to diagnose issues in the data, such as skewed distributions or the presence of outliers, guiding the subsequent preprocessing steps.

3. Data Cleaning:
   - Handling missing values: Either removing or imputing missing data.
   - Removing duplicates: Eliminating identical records to avoid redundancy.
   - Correcting errors: Addressing any inaccuracies or inconsistencies in the data.

4. Data Transformation:
   - Encoding categorical variables: Converting categorical data into numerical format for analysis.

- Scaling features: Standardizing or normalizing numerical features to a consistent scale.
- Data discretization: Converting continuous data into discrete categories.

5. Data Reduction:
   - Feature selection: Choosing the most relevant features for analysis to reduce dimensionality.
   - Dimensionality reduction: Techniques like Principal Component Analysis (PCA) to reduce the number of features while retaining essential information.

6. Handling Outliers:
   - Identifying and dealing with outliers that can skew analysis or machine learning models.

After data preprocessing, the next steps involve preparing the data for model building. Here are some common tasks in this phase:

1. Data Splitting:
   - Dividing the dataset into training and testing sets(validation set) . The training set is used to train the model, and the testing set is used to evaluate its performance.

2. Model Selection:
   - Choosing an appropriate machine learning or statistical model based on the nature of the problem (classification, regression, neural networks, etc.) and the characteristics of the data.

3. Model Training:
   - Using the training set to teach the model to make predictions. The model learns the patterns and relationships present in the data.

4. Hyperparameter Tuning(done for random forests):
   - Adjusting the hyperparameters of the model to optimize its performance. This often involves using techniques like grid search or randomized search.

5. Validation Set (The testing set):
   - Creating a separate validation set to fine-tune the model and prevent overfitting. This set is used to evaluate the model's performance during training.

6. Model Evaluation:

- Assessing the model's performance using the test set. Common evaluation metrics include accuracy, precision, recall, F1 score (for classification), and confusion matrix.

7. Deployment using Flask:
    - If the model meets the desired performance and is ready for deployment, one common approach is using Flask, a web framework in Python.
    - Flask allows for the creation of a web application to serve the model predictions in real-time.
    - The trained model is integrated into the Flask application, creating an API endpoint where users can input data and receive predictions.
    - This API can be hosted on a server, making the model accessible to users or other systems over the internet.
    - Flask provides a straightforward way to handle HTTP requests, making it a popular choice for deploying machine learning models in production.
    - Additionally, Flask allows for easy integration with front-end applications, providing a user-friendly interface for interacting with the model.

These steps were followed by us to build, test and deploy the model, the complexity of the problem, and the goals of the analysis. The overall aim is to create a robust and effective model that can make accurate predictions or provide valuable insights.

Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.
o ML Concepts
o Supervised learning: https://www.javatpoint.com/supervised-machine-learning
o Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning

o Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

o Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
o KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning
o Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/

o Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/

o Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the Business Problem

Autism Spectrum Disorder (ASD) poses a significant challenge in terms of early detection and diagnosis. The current scenario is marked by lengthy waiting times for ASD diagnoses, resulting in substantial healthcare costs. The lack of cost-effective procedures compounds the issue, necessitating a solution that addresses both the economic and health-related aspects of ASD. The overarching business problem is to develop a streamlined and efficient ASD screening method that reduces waiting times, and healthcare costs
and offers a reliable tool for early detection.

Activity 2: Business Requirements

To address the identified business problem effectively, several key business requirements need consideration:

1. **Time Efficiency:** The screening method must be time-efficient, significantly reducing the current waiting times associated with ASD diagnoses. This is crucial for ensuring early intervention and support for affected individuals.

2. **Cost-Effectiveness:** The proposed solution should be economically viable, aiming to minimize healthcare costs associated with ASD. This involves streamlining procedures and utilizing resources efficiently to make the screening accessible to a broader population.

3. **Reliability and Accuracy:** The screening tool must demonstrate high sensitivity and specificity, ensuring reliable detection of ASD cases while minimizing false positives and negatives. This is essential for providing accurate information to healthcare professionals and individuals seeking guidance.

4. **Accessibility:** The solution should be easily accessible, catering to a diverse demographic. Accessibility is not just about physical availability but also ensuring that the screening tool is user-friendly and understandable for various stakeholders, including healthcare professionals and individuals.

5. **Data Inclusivity:** The dataset used for screening should encompass a comprehensive range of behavioural traits and individual characteristics. This inclusivity is vital for improving the screening process's efficiency and enhancing its predictive accuracy.

6. **Scalability:** The proposed solution should be scalable to accommodate the increasing number of ASD cases globally. This involves designing a system that can handle a growing dataset and adapt to evolving diagnostic needs.

By addressing these business requirements, the goal is to develop a solution that not only meets the immediate needs of efficient ASD screening but also establishes a foundation for ongoing improvements and adaptations in the future.

Activity 3: Literature review

Autism Spectrum Disorder (ASD) is a complex neurodevelopmental disorder affecting social communication and behaviour. Diagnosable at any age, it is predominantly identified in early childhood, presenting symptoms such as challenges in social interaction, repetitive behaviours, and sensory sensitivities. The aetiology of ASD remains unknown but is believed to result from a combination of genetic and environmental factors. While there is no cure, evidence-based treatments, including applied behaviour analysis (ABA), speech therapy, occupational therapy, and social skills training, aim to improve symptoms and enhance the quality of life for individuals with ASD.

References to the Diagnostic and Statistical Manual of Mental Disorders (DSM-5), the Centers for Disease Control and Prevention, and diagnostic tools like the Autism Diagnostic Interview-Revised (ADI-R) and Gilliam Autism Rating Scale (GARS-3) contribute to the foundational understanding of ASD.

Problem Statement Definition - Autism Spectrum Disorder:
The identified problem in the literature is framed as a comprehensive literature survey. This survey aims to explore and summarize the current state of knowledge on ASD. The defined objectives include understanding the prevalence, causes, symptoms, and diagnostic methods associated with ASD. Additionally, the literature review seeks to identify effective treatments, challenges, and opportunities faced by individuals with ASD and their families. Furthermore, it aims to pinpoint gaps in existing research on ASD.

Methodology:
The literature survey will utilize various databases, such as PubMed, PsycINFO, and Google Scholar. Employing search terms like "autism spectrum disorder," "autism," "ASD," "neurodevelopmental disorders," "developmental disabilities," "social communication disorder," and "restricted and repetitive behaviours," the review aims to provide a comprehensive overview of the current state of research on ASD.

This literature review serves as a valuable resource for gaining insights into the multifaceted aspects of ASD, from its clinical manifestations to the existing gaps in knowledge, providing a foundation for future research endeavours in the field.

Activity 4: Social or Business Impact

The impact of Autism Spectrum Disorder (ASD) extends beyond its clinical manifestations, influencing both the social and business domains. Understanding and addressing this impact is essential for devising effective strategies and solutions.

Social Impact:
ASD significantly affects individuals' social lives, presenting challenges in communication and interaction. Families of individuals with ASD often face unique struggles, including navigating healthcare systems, accessing appropriate educational resources, and fostering social integration. The societal impact involves promoting inclusivity, understanding, and support for individuals with ASD, contributing to a more compassionate and empathetic community.

Business Impact:
From a business perspective, the economic implications of ASD are considerable. The costs associated with healthcare, educational support, and therapeutic interventions pose financial challenges for families and strain public resources. Developing efficient screening methods and evidence-based treatments not only improves the quality of life for individuals with ASD but also has the potential to reduce long-term societal costs. Additionally, businesses that embrace inclusivity and accommodate employees with ASD may benefit from a diverse workforce, fostering innovation and creativity.

In summary, the social impact emphasizes the need for a compassionate and supportive society, while the business impact highlights the economic considerations and potential benefits of addressing ASD through effective screening and treatment methods. By acknowledging and addressing both aspects, the goal is to create a more holistic approach to understanding and mitigating the impact of ASD on individuals and society at large.

Milestone 2: Data Collection & Preparation

Activity 1: Collect the dataset
There are many popular open sources for collecting the data.
E.g.: kaggle.com, UCI repository, GitHub etc.
In this project we have used from Kaggle and dataset was in .csv format. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset. Link: https://www.kaggle.com/code/faizunnabi/autism-screening-classification
As the dataset is downloaded. Let us read and understand the data.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 704 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   A1_Score         704 non-null    int64  
 1   A2_Score         704 non-null    int64  
 2   A3_Score         704 non-null    int64  
 3   A4_Score         704 non-null    int64  
 4   A5_Score         704 non-null    int64  
 5   A6_Score         704 non-null    int64  
 6   A7_Score         704 non-null    int64  
 7   A8_Score         704 non-null    int64  
 8   A9_Score         704 non-null    int64  
 9   A10_Score        704 non-null    int64  
 10  age              702 non-null    float64
 11  gender           704 non-null    object 
 12  ethnicity        704 non-null    object 
 13  jundice          704 non-null    object 
 14  austim           704 non-null    object 
 15  contry_of_res    704 non-null    object 
 16  used_app_before  704 non-null    object 
 17  result           704 non-null    float64
 18  age_desc         704 non-null    object 
 19  relation         704 non-null    object 
 20  Class/ASD        704 non-null    object 
dtypes: float64(2), int64(10), object(9)
memory usage: 115.6+ KB
```

THE CODE:

The Code below is a representation of all the steps mentioned above in the “project flow”:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

IMPORTING DATASET

```
df = pd.read_csv("https://raw.githubusercontent.com/smartinternz02/SI-GuidedProj
```

```
df.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_S
0	1	1	1	1	0	0	1	
1	1	1	0	1	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	0	1	0	0	1	

DATA PREPROCESSING

EDA

```
df.shape
```

```
(704, 21)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 704 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   A1_Score         704 non-null    int64  
 1   A2_Score         704 non-null    int64  
 2   A3_Score         704 non-null    int64  
 3   A4_Score         704 non-null    int64  
 4   A5_Score         704 non-null    int64  
 5   A6_Score         704 non-null    int64  
 6   A7_Score         704 non-null    int64  
 7   A8_Score         704 non-null    int64  
 8   A9_Score         704 non-null    int64  
 9   A10_Score        704 non-null    int64  
 10  age              702 non-null    float64
 11  gender           704 non-null    object 
 12  ethnicity        704 non-null    object 
 13  jundice          704 non-null    object 
 14  austim           704 non-null    object 
 15  contry_of_res    704 non-null    object 
 16  used_app_before  704 non-null    object 
 17  result           704 non-null    float64
 18  age_desc         704 non-null    object 
 19  relation         704 non-null    object 
 20  Class/ASD        704 non-null    object 
dtypes: float64(2), int64(10), object(9)
memory usage: 115.6+ KB
```

```
df.isnull().any()
```

```
A1_Score           False
A2_Score           False
A3_Score           False
A4_Score           False
A5_Score           False
A6_Score           False
A7_Score           False
A8_Score           False
A9_Score           False
A10_Score          False
age                 True
gender             False
ethnicity          False
jundice            False
austim             False
contry_of_res      False
used_app_before    False
result             False
age_desc           False
relation           False
Class/ASD          False
dtype: bool
```

```
df.isnull().sum()
```

```
A1_Score           0
A2_Score           0
A3_Score           0
A4_Score           0
A5_Score           0
A6_Score           0
A7_Score           0
A8_Score           0
A9_Score           0
A10_Score          0
age                2
gender             0
ethnicity          0
jundice            0
austim             0
contry_of_res      0
used_app_before    0
result             0
age_desc           0
relation           0
Class/ASD          0
dtype: int64
```

```
df.describe()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Sc
count	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	704.00
mean	0.721591	0.453125	0.457386	0.495739	0.498580	0.284091	0.41
std	0.448535	0.498152	0.498535	0.500337	0.500353	0.451301	0.49
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

```
data = df.dropna()
data.isnull().sum()
```

```
A1_Score           0
A2_Score           0
A3_Score           0
A4_Score           0
A5_Score           0
A6_Score           0
A7_Score           0
A8_Score           0
A9_Score           0
A10_Score          0
age                0
gender             0
ethnicity          0
jundice            0
austim             0
contry_of_res      0
used_app_before    0
result             0
age_desc           0
relation           0
Class/ASD          0
dtype: int64
```

```
data.shape
```

```
(702, 21)
```

```
data.describe()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Sc
count	702.000000	702.000000	702.000000	702.000000	702.000000	702.000000	702.00
mean	0.723647	0.452991	0.458689	0.497151	0.498575	0.284900	0.41
std	0.447512	0.498140	0.498646	0.500348	0.500354	0.451689	0.49
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 702 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   A1_Score         702 non-null    int64
 1   A2_Score         702 non-null    int64
 2   A3_Score         702 non-null    int64
 3   A4_Score         702 non-null    int64
 4   A5_Score         702 non-null    int64
 5   A6_Score         702 non-null    int64
 6   A7_Score         702 non-null    int64
 7   A8_Score         702 non-null    int64
 8   A9_Score         702 non-null    int64
 9   A10_Score        702 non-null    int64
 10  age              702 non-null    float64
 11  gender           702 non-null    object
 12  ethnicity        702 non-null    object
 13  jundice          702 non-null    object
 14  austim           702 non-null    object
 15  contry_of_res    702 non-null    object
 16  used_app_before  702 non-null    object
 17  result           702 non-null    float64
 18  age_desc         702 non-null    object
 19  relation         702 non-null    object
 20  Class/ASD        702 non-null    object
dtypes: float64(2), int64(10), object(9)
memory usage: 120.7+ KB
```

```
data.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_S
0	1	1	1	1	0	0	1	
1	1	1	0	1	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	0	1	0	0	1	

```
data.ethnicity.nunique()
```

```
12
```

```
data.ethnicity.unique()
```

```
array(['White-European', 'Latino', '?', 'Others', 'Black', 'Asian',
       'Middle Eastern ', 'Pasifika', 'South Asian', 'Hispanic',
       'Turkish', 'others'], dtype=object)
```

```
data.ethnicity.value_counts()
```

```
White-European     233
Asian              123
?                   93
Middle Eastern      92
Black               43
South Asian         36
Others              30
Latino              20
Hispanic            13
Pasifika            12
Turkish              6
others               1
Name: ethnicity, dtype: int64
```

```
data.jundice.unique()
```

```
array(['no', 'yes'], dtype=object)
```

```
data.austim.value_counts()
```

```
no     611
yes     91
Name: austim, dtype: int64
```

```
data.contry_of_res.nunique()
```

```
67
```

```
print(data.contry_of_res.value_counts())
```

```
United States           113
United Arab Emirates     82
New Zealand              81
India                    81
United Kingdom           77
                       ...
China                     1
Chile                     1
Lebanon                   1
Burundi                   1
Cyprus                    1
Name: contry_of_res, Length: 67, dtype: int64
```

```
data.used_app_before.value_counts()
```

```
no     690
yes     12
Name: used_app_before, dtype: int64
```

```
data.age_desc.value_counts()
```

```
18 and more    702
Name: age_desc, dtype: int64
```

```
data.relation.value_counts()
```

```
Self                        522
?                            93
Parent                       50
Relative                     28
Others                        5
Health care professional      4
Name: relation, dtype: int64
```

```
data['Class/ASD'].value_counts()
```

```
NO     513
YES    189
Name: Class/ASD, dtype: int64
```

DATA CLEANING

Removing columns ["contry_of_res","age_desc"]

- country_of_res - Describe the country of residence of indivisuals , not enough significant data to create conclusion
- age_desc - Describes if someone is above 18 or not, which is redundant as not only is age given but also that everyone is above 18

```
clean_data = data.drop(columns = ["contry_of_res","age_desc"],axis=1)
clean_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 702 entries, 0 to 703
Data columns (total 19 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   A1_Score         702 non-null    int64  
 1   A2_Score         702 non-null    int64  
 2   A3_Score         702 non-null    int64  
 3   A4_Score         702 non-null    int64  
 4   A5_Score         702 non-null    int64  
 5   A6_Score         702 non-null    int64  
 6   A7_Score         702 non-null    int64  
 7   A8_Score         702 non-null    int64  
 8   A9_Score         702 non-null    int64  
 9   A10_Score        702 non-null    int64  
 10  age              702 non-null    float64
 11  gender           702 non-null    object 
 12  ethnicity        702 non-null    object 
 13  jundice          702 non-null    object 
 14  austim           702 non-null    object 
 15  used_app_before  702 non-null    object 
 16  result           702 non-null    float64
 17  relation         702 non-null    object 
 18  Class/ASD        702 non-null    object 
dtypes: float64(2), int64(10), object(7)
memory usage: 109.7+ KB
```

```
clean_data.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_S
0	1	1	1	1	0	0	1	
1	1	1	0	1	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	0	1	0	0	1	

Label Encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
clean_data.gender = le.fit_transform(clean_data.gender)
clean_data.ethnicity = le.fit_transform(clean_data.ethnicity)
clean_data.jundice = le.fit_transform(clean_data.jundice)
clean_data.austim = le.fit_transform(clean_data.austim)
clean_data.used_app_before = le.fit_transform(clean_data.used_app_before)
clean_data.relation = le.fit_transform(clean_data.relation)
clean_data["Class/ASD"] = le.fit_transform(clean_data["Class/ASD"])
clean_data.head(25)
```

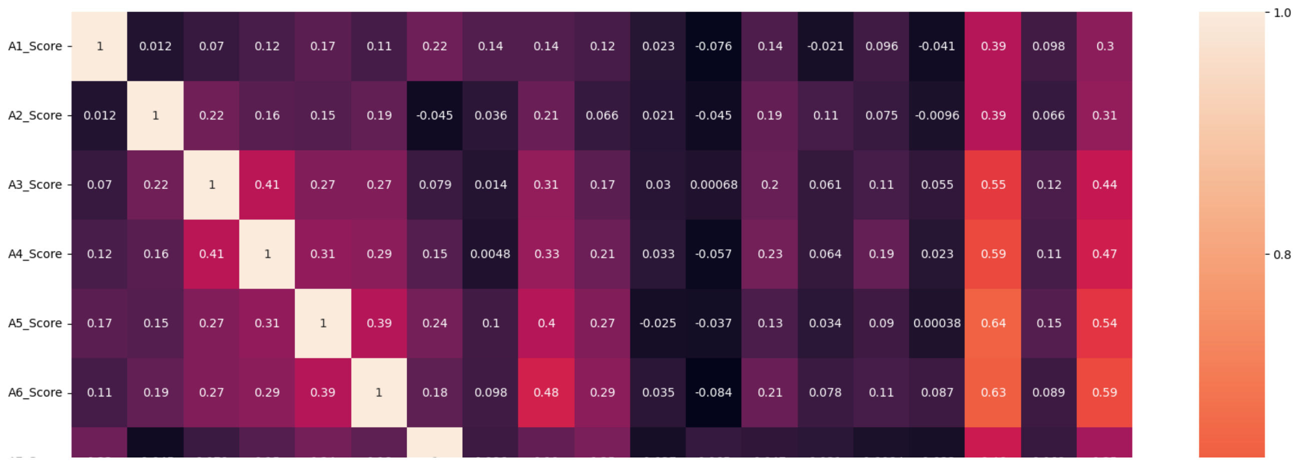
	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_
0	1	1	1	1	0	0	1	
1	1	1	0	1	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	0	1	0	0	1	
4	1	0	0	0	0	0	0	
5	1	1	1	1	1	0	1	
6	0	1	0	0	0	0	0	
7	1	1	1	1	0	0	0	
8	1	1	0	0	1	0	0	
9	1	1	1	1	0	1	1	
10	1	1	1	1	1	1	1	
11	0	1	0	1	1	1	1	
12	0	1	1	1	1	1	0	
13	1	0	0	0	0	0	1	
14	1	0	0	0	0	0	1	
15	1	1	0	1	1	0	0	
16	1	0	0	0	0	0	1	
17	0	0	0	0	0	0	0	
18	0	0	1	0	1	1	0	
19	0	0	0	0	0	0	1	
20	0	1	1	1	0	0	0	
21	0	0	0	0	0	0	0	
22	0	0	0	1	0	0	1	
23	0	0	0	0	0	0	0	
24	1	1	1	1	0	0	0	

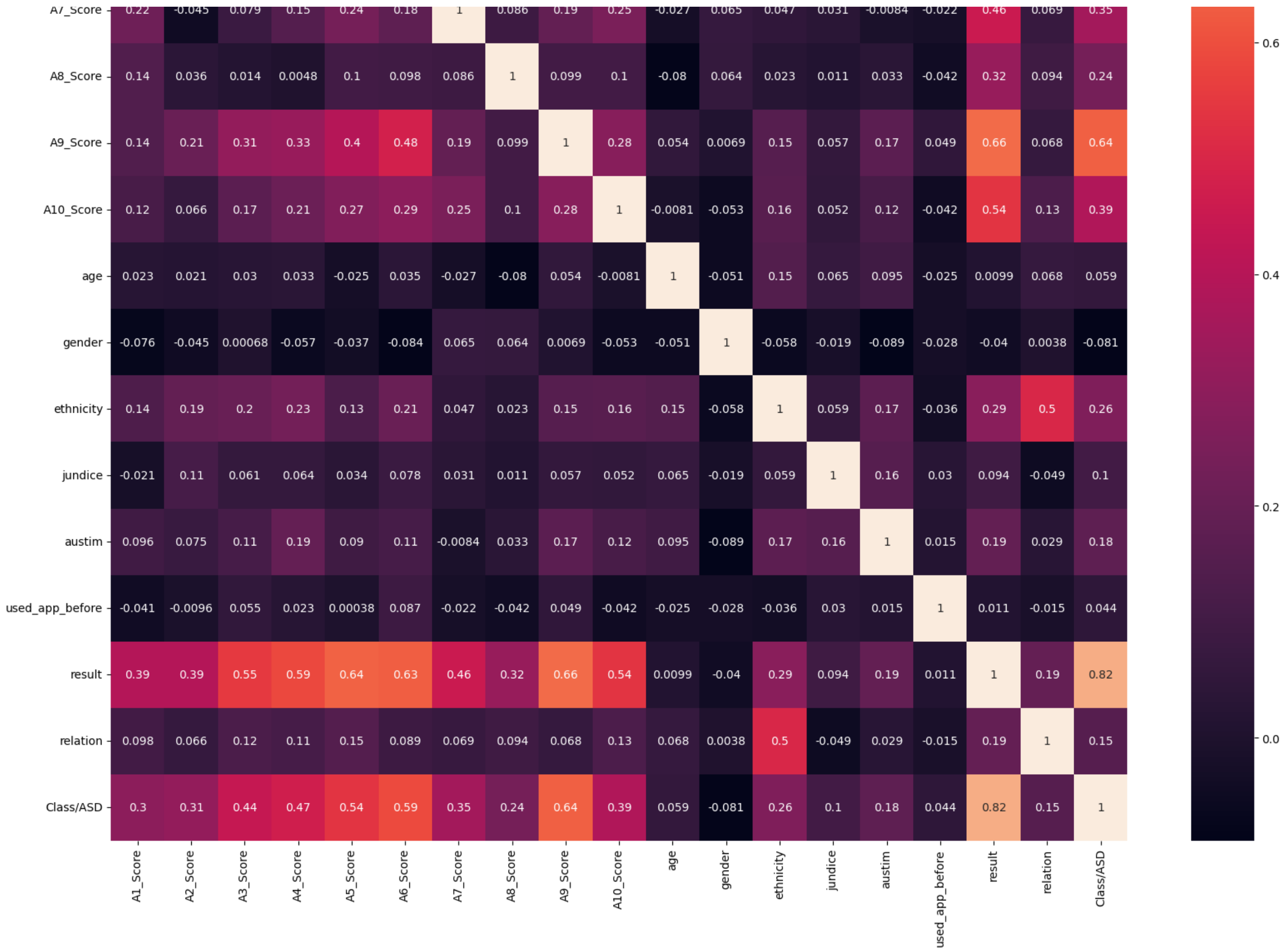
```
clean_data.corr()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score
A1_Score	1.000000	0.012033	0.070229	0.123898	0.170253	0.107769
A2_Score	0.012033	1.000000	0.224762	0.159718	0.151401	0.186408
A3_Score	0.070229	0.224762	1.000000	0.411198	0.265631	0.267671
A4_Score	0.123898	0.159718	0.411198	1.000000	0.307682	0.293951
A5_Score	0.170253	0.151401	0.265631	0.307682	1.000000	0.393140
A6_Score	0.107769	0.186408	0.267671	0.293951	0.393140	1.000000
A7_Score	0.219444	-0.044838	0.078866	0.152150	0.236398	0.176153
A8_Score	0.142301	0.035919	0.014268	0.004794	0.102513	0.097996
A9_Score	0.142904	0.206045	0.313894	0.326397	0.397423	0.478777
A10_Score	0.118341	0.066231	0.168516	0.211155	0.265461	0.294771
age	0.023059	0.020824	0.029504	0.032539	-0.025095	0.034705
gender	-0.075594	-0.044654	0.000685	-0.056789	-0.036949	-0.083858
ethnicity	0.135900	0.190571	0.201709	0.233603	0.126319	0.208435
jundice	-0.020668	0.112884	0.060981	0.064086	0.034435	0.077831
austim	0.096239	0.074783	0.112848	0.193043	0.090159	0.113444
used_app_before	-0.041378	-0.009623	0.055043	0.022731	0.000376	0.087193
result	0.394739	0.392229	0.551552	0.585232	0.640141	0.630066
relation	0.098463	0.065516	0.124116	0.112183	0.153108	0.088850
Class/ASD	0.296099	0.312159	0.440248	0.469136	0.538055	0.591647

```
plt.figure(figsize=(20,20))
sns.heatmap(clean_data.corr(),annot=True)
```

<Axes: >




A7_Score
A8_Score
A9_Score
A10_Score
age
gender
ethnicity
jundice
austim
used_app_before
result
relation
Class/ASD
A1_Score
A2_Score
A3_Score
A4_Score
A5_Score
A6_Score
A7_Score
A8_Score
A9_Score
A10_Score
age
gender
ethnicity
jundice
austim
used_app_before
result
relation
Class/ASD
0.6
0.4
0.2
0.0

```
main_df = clean_data.drop(columns = ["gender"],axis=1)
main_df
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_
0	1	1	1	1	0	0	1	
1	1	1	0	1	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	0	1	0	0	1	
4	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	
699	0	1	0	1	1	0	1	
700	1	0	0	0	0	0	0	
701	1	0	1	1	1	0	1	
702	1	0	0	1	1	0	1	
703	1	0	1	1	1	0	1	

702 rows × 18 columns

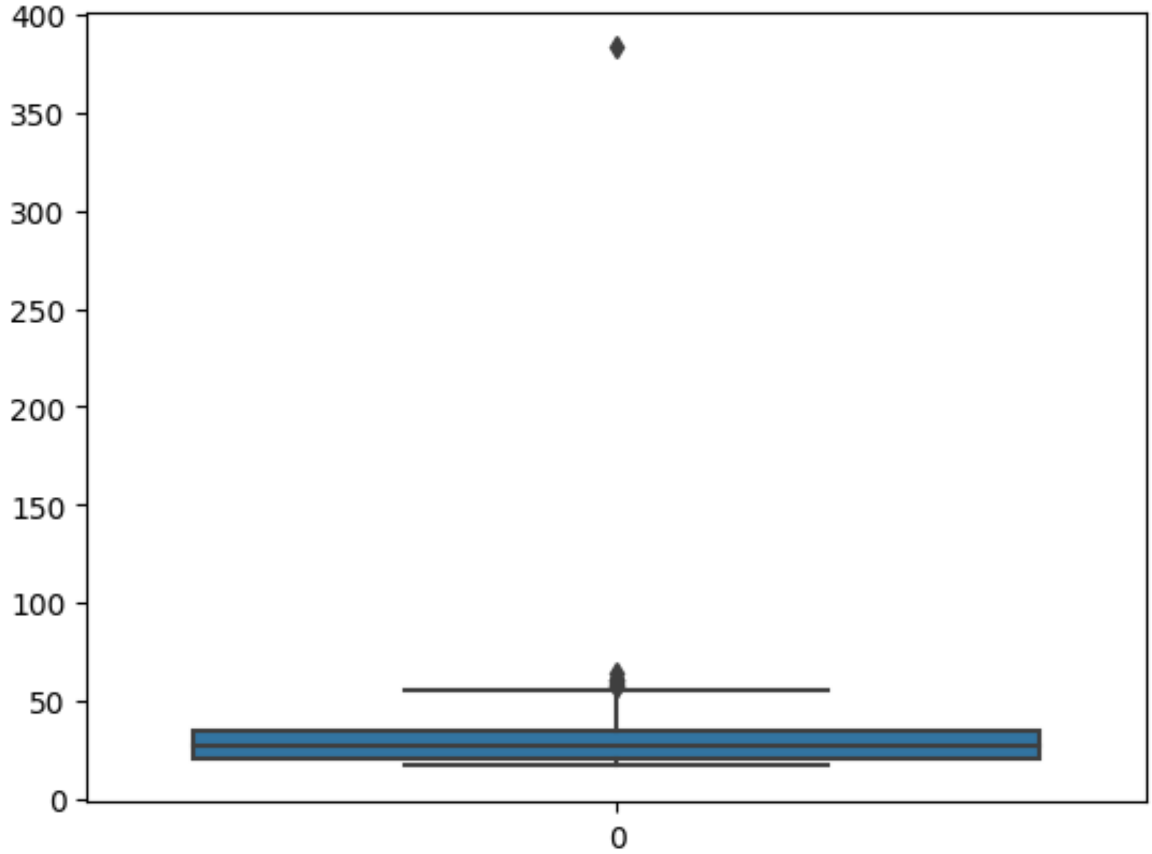
```
main_df.describe()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Sc
count	702.000000	702.000000	702.000000	702.000000	702.000000	702.000000	702.00
mean	0.723647	0.452991	0.458689	0.497151	0.498575	0.284900	0.41
std	0.447512	0.498140	0.498646	0.500348	0.500354	0.451689	0.49
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

Outlier Detection and removal

```
sns.boxplot(main_df.age)
```

```
<Axes: >
```



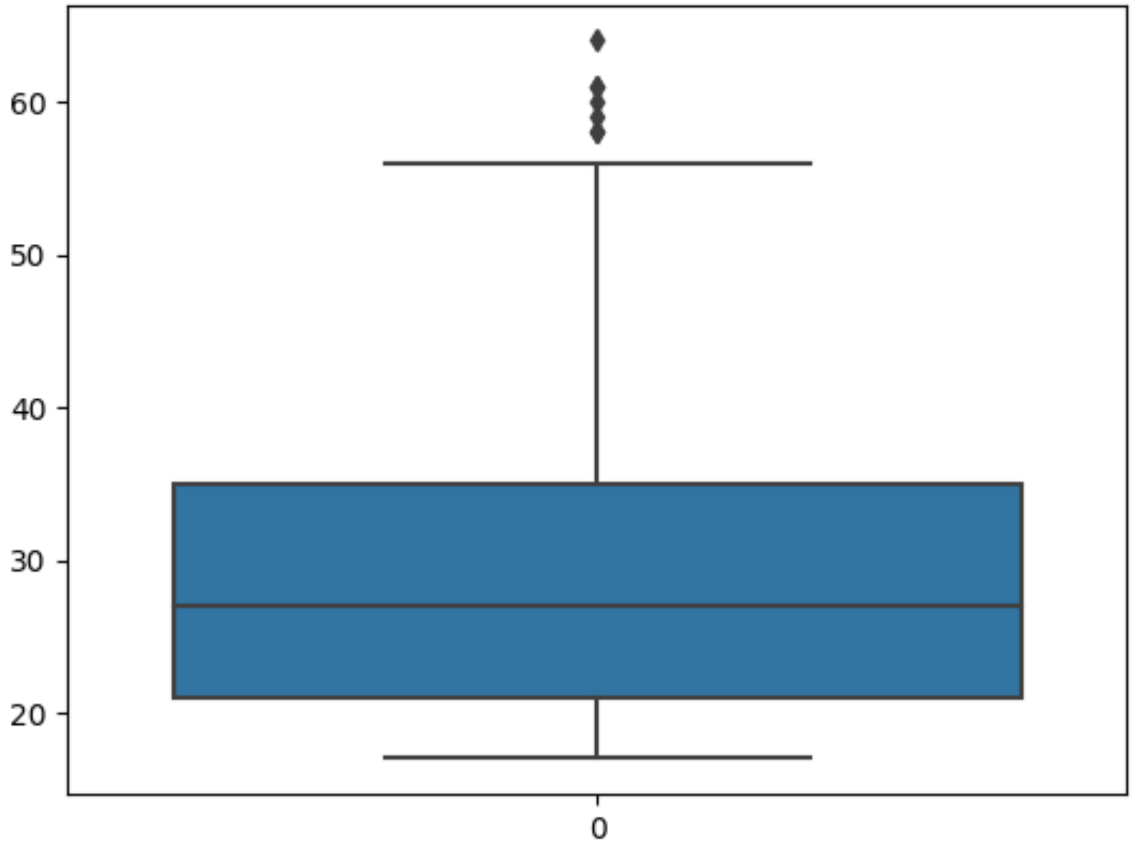

```
## using Z - score
from scipy import stats
```

```
age_zscore = stats.zscore(main_df.age)
age_zscore
```

```
0     -0.224180
1     -0.345424
2     -0.163558
3      0.321417
4      0.624526
         ...
699   -0.284802
700    0.260795
701   -0.345424
702    0.321417
703   -0.224180
Name: age, Length: 702, dtype: float64
```

```
df_z = main_df[np.abs(age_zscore)<=3]
sns.boxplot(df_z.age)
```

```
<Axes: >
```



SPLITTING DATA INTO X AND Y

```
X = df_z.drop(columns = ['Class/ASD'],axis = 1)
X.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_S
0	1	1	1	1	0	0	1	
1	1	1	0	1	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	0	1	0	0	1	
4	1	0	0	0	0	0	0	

```
y = df_z["Class/ASD"]
y.head()
```

```
0    0
1    0
2    1
3    0
4    0
Name: Class/ASD, dtype: int64
```

Train test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train,y_test = train_test_split(X,y,test_size = 0.2, random_s
```

```
X_train.shape, X_test.shape, y_train.shape ,y_test.shape
```

```
((560, 17), (141, 17), (560,), (141,))
```

```
X_train.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_
168	0	0	1	1	1	0	0	
545	1	0	1	1	1	1	1	
287	1	0	0	1	1	0	0	
363	0	1	0	1	0	0	0	
113	0	1	0	0	0	0	0	

MODEL TRAINING

Logistic Regression

```
import os
import cv2
import numpy as np
from skimage import io, color, segmentation
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import numpy as np
import pandas as pd
```

```
class LogisticRegression:
    def __init__(self, lr=0.01, num_iter=100000, fit_intercept=True, verbose=Fal
        self.lr = lr
        self.num_iter = num_iter
        self.fit_intercept = fit_intercept
        self.verbose = verbose
```

```
   def __add_intercept(self, X):
        intercept = np.ones((X.shape[0], 1))
        return np.concatenate((intercept, X), axis=1)
```

```
import numpy as np

class YourClassName:
    def __sigmoid(self, z):
        return 1 / (1 + np.exp(-z))

    def __loss(self, h, y):
        return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()
```

```
 def fit(self, X, y):
        if self.fit_intercept:
            X = self.__add_intercept(X)
```

```
class YourClass:
    def __init__(self, X):
        # Initialize self.theta with np.zeros
        self.theta = np.zeros(X.shape[1])
```

```
class YourClass:
    def __init__(self, X, num_iter, lr):
        self.X = X
        self.num_iter = num_iter
        self.lr = lr
        self.theta = np.zeros(X.shape[1])

    def __sigmoid(self, z):
        # Implementation of the sigmoid function
        pass

    def __loss(self, h, y):
        # Implementation of the loss function
        pass

    def fit(self, y):
        for i in range(self.num_iter):
            z = np.dot(self.X, self.theta)
            h = self.__sigmoid(z)
            gradient = np.dot(self.X.T, (h - y)) / y.size
            self.theta -= self.lr * gradient
            z = np.dot(self.X, self.theta)
            h = self.__sigmoid(z)
            loss = self.__loss(h, y)
```

```
class LinearRegression:
    def __init__(self, X, y):
        self.theta = np.zeros(X.shape[1])
```

```
class YourClass:
    def __init__(self, X, num_iter, lr, verbose=True):
        self.X = X
        self.num_iter = num_iter
        self.lr = lr
        self.theta = np.zeros(X.shape[1])
        self.verbose = verbose  # Assuming verbose is an attribute of the class.

    def __sigmoid(self, z):
        # Implementation of the sigmoid function
        pass

    def __loss(self, h, y):
        # Implementation of the loss function
        pass

    def fit(self, y):
        for i in range(self.num_iter):
            z = np.dot(self.X, self.theta)
            h = self.__sigmoid(z)
            gradient = np.dot(self.X.T, (h - y)) / y.size
            self.theta -= self.lr * gradient
            z = np.dot(self.X, self.theta)
            h = self.__sigmoid(z)
            loss = self.__loss(h, y)

            if self.verbose and i % 10000 == 0:
                print(f'loss: {loss} \t')
```

```
    def predict_prob(self, X):
        if self.fit_intercept:
            X = self.__add_intercept(X)

        return self.__sigmoid(np.dot(X, self.theta))

    def predict(self, X):
        return self.predict_prob(X).round()
```

```
#read data
df=pd.read_csv("https://raw.githubusercontent.com/smartinternz02/SI-GuidedProjec
```

```
print(df.columns)
```

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Scor
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jundice', 'austim', 'contry_of_res', 'used_app_before'
       'result', 'age_desc', 'relation', 'Class/ASD'],
      dtype='object')
```

```
X = np.array(X)
```

```
print(len(X))
```

```
701
```

```
print("Length of y")
print(len(y))

print("DataFrame head")
print(df.head())
```

```
Length of y
701
DataFrame head
   A1_Score  A2_Score  A3_Score  A4_Score  A5_Score  A6_Score  A7_Score  \
0         1         1         1         1         0         0         1
1         1         1         0         1         0         0         0
2         1         1         0         1         1         0         1
3         1         1         0         1         0         0         1
4         1         0         0         0         0         0         0

   A8_Score  A9_Score  A10_Score  ...  gender       ethnicity jundice austi
0         1         0          0  ...       f  White-European      no     n
1         1         0          1  ...       m          Latino      no    ye
2         1         1          1  ...       m          Latino     yes    ye
3         1         0          1  ...       f  White-European      no    ye
4         1         0          0  ...       f               ?      no     n

   contry_of_res used_app_before result     age_desc relation Class/ASD
0  United States              no    6.0  18 and more     Self        NO
1         Brazil              no    5.0  18 and more     Self        NO
2          Spain              no    8.0  18 and more   Parent       YES
3  United States              no    6.0  18 and more     Self        NO
4          Egypt              no    2.0  18 and more        ?        NO

[5 rows x 21 columns]
```

```
import numpy as np

# Convert the list 'y' to a NumPy array
y = np.array(y)

# Now you can access the shape attribute
print("Shape of y:", y.shape)
print("Shape of X:", X.shape)
```

```
Shape of y: (701,)
Shape of X: (701, 17)
```

```
import numpy as np
from sklearn.model_selection import train_test_split

# Create some sample data
X = np.array(X)
y = np.array(y)

# Split the data into training and test sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_


# Print the shape of the training and test sets
print(X_train.shape)
print(X_test.shape)
```

```
(560, 17)
(141, 17)
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# Split your data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.2, random_

# Create and fit your scikit-learn logistic regression model
modelSklearn = LogisticRegression()
modelSklearn.fit(X_train, Y_train)

# Make predictions and evaluate the scikit-learn model
predsSklearn = modelSklearn.predict(X_test)
print("Sklearn predictions: ", predsSklearn)
print("Sklearn model score: ", (predsSklearn == Y_test).mean())
```

```
Sklearn predictions:  [0 0 1 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 0 0
 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0 1 0 1 0 0 0 1 0 1 0 0 1 0
 1 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1 1 0 1 0 1
 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 1]
Sklearn model score:  1.0
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:4
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regr
  n_iter_i = _check_optimize_result(
```

```
import os

import glob

path = 'C:/Users/mail2/Desktop/flippin/pics/'
i = 0

for infile in glob.glob( os.path.join(path, '*.jpg') ):

    img = Image.open(infile)
    print ("current file is: " + infile)
    img.transpose(Image.FLIP_LEFT_RIGHT)
    img.transpose(Image.FLIP_TOP_BOTTOM)
    img.transpose(
        Image.FLIP_LEFT_RIGHT).transpose(
        Image.FLIP_TOP_BOTTOM).save("combined%s.jpg")
print('done')
```

```
done
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
```

```
# Initialize the model

knn = KNeighborsClassifier()
```

```
knn.fit(X_train,y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
ypred = knn.predict(X_test)
```

```
ypred
```

```
array([0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1])
```

```
y_test
```

```
array([0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 1])
```

```
from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,ypred))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97        98
           1       0.93      0.93      0.93        43

    accuracy                           0.96       141
   macro avg       0.95      0.95      0.95       141
weighted avg       0.96      0.96      0.96       141
```

```
confusion_matrix(y_test,ypred)
```

```
array([[95,  3],
       [ 3, 40]])
```

Random Forests

```
import pandas as pd
import numpy as np
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 704 entries, 0 to 703
Data columns (total 21 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   A1_Score         704 non-null    int64  
 1   A2_Score         704 non-null    int64  
 2   A3_Score         704 non-null    int64  
 3   A4_Score         704 non-null    int64  
 4   A5_Score         704 non-null    int64  
 5   A6_Score         704 non-null    int64  
 6   A7_Score         704 non-null    int64  
 7   A8_Score         704 non-null    int64  
 8   A9_Score         704 non-null    int64  
 9   A10_Score        704 non-null    int64  
 10  age              702 non-null    float64
 11  gender           704 non-null    object 
 12  ethnicity        704 non-null    object 
 13  jundice          704 non-null    object 
 14  austim           704 non-null    object 
 15  contry_of_res    704 non-null    object 
 16  used_app_before  704 non-null    object 
 17  result           704 non-null    float64
 18  age_desc         704 non-null    object 
 19  relation         704 non-null    object 
 20  Class/ASD        704 non-null    object 
dtypes: float64(2), int64(10), object(9)
memory usage: 115.6+ KB
```

```
from sklearn.ensemble import RandomForestClassifier
model2 =RandomForestClassifier(criterion='entropy')
```

```
model2.fit(X_train,y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(criterion='entropy')
```

```
r_y_predict = model2.predict(X_test)
r_y_predict_train = model2.predict(X_train)
```

```
print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict))
print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train))
```

```
Testing Accuracy =  1.0
Training Accuracy =  1.0
```

```
pd.crosstab(y_test,r_y_predict)
```

col_0	0	1
row_0		
0	98	0
1	0	43

```
print(classification_report(y_test,r_y_predict))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        98
           1       1.00      1.00      1.00        43

    accuracy                           1.00       141
   macro avg       1.00      1.00      1.00       141
weighted avg       1.00      1.00      1.00       141
```

NEURAL NETWORKS

```
import os

def create_model_checkpoint(model_name, save_path="model_experiments"):
  return tf.keras.callbacks.ModelCheckpoint(filepath=os.path.join(save_path, mod
                                            monitor="val_loss",
                                            verbose=0,
                                            save_best_only=True)
```

```
import tensorflow as tf
tf.random.set_seed(42)

model_1 = tf.keras.Sequential([
    tf.keras.layers.Dense(10),
    tf.keras.layers.Dense(1)
])

model_1.compile(loss=tf.keras.losses.BinaryCrossentropy(), # binary since we are
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_1 = model_1.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_te
```

```
Epoch 1/10
18/18 [==============================] - 12s 31ms/step - loss: 4.0215 - acc
Epoch 2/10
18/18 [==============================] - 0s 12ms/step - loss: 4.0215 - accu
Epoch 3/10
18/18 [==============================] - 0s 12ms/step - loss: 4.0215 - accu
Epoch 4/10
18/18 [==============================] - 0s 8ms/step - loss: 4.0215 - accur
Epoch 5/10
18/18 [==============================] - 0s 13ms/step - loss: 4.0215 - accu
Epoch 6/10
18/18 [==============================] - 0s 9ms/step - loss: 4.0215 - accur
Epoch 7/10
18/18 [==============================] - 0s 12ms/step - loss: 4.0215 - accu
Epoch 8/10
18/18 [==============================] - 0s 10ms/step - loss: 4.0215 - accu
Epoch 9/10
18/18 [==============================] - 0s 16ms/step - loss: 4.0215 - accu
Epoch 10/10
18/18 [==============================] - 0s 14ms/step - loss: 4.0215 - accu
```

```
tf.random.set_seed(42)

model_2 = tf.keras.Sequential([
    tf.keras.layers.Dense(10,activation="relu"),
    tf.keras.layers.Dense(1,activation="sigmoid")
])

model_2.compile(loss=tf.keras.losses.BinaryCrossentropy(), # binary since we are
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_2 = model_2.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_te
```

```
Epoch 1/10
18/18 [==============================] - 2s 18ms/step - loss: 1.8055 - accu
Epoch 2/10
18/18 [==============================] - 0s 6ms/step - loss: 0.4498 - accur
Epoch 3/10
18/18 [==============================] - 0s 6ms/step - loss: 0.4073 - accur
Epoch 4/10
18/18 [==============================] - 0s 12ms/step - loss: 0.3975 - accu
Epoch 5/10
18/18 [==============================] - 0s 7ms/step - loss: 0.4029 - accur
Epoch 6/10
18/18 [==============================] - 0s 5ms/step - loss: 0.3865 - accur
Epoch 7/10
18/18 [==============================] - 0s 9ms/step - loss: 0.3741 - accur
Epoch 8/10
18/18 [==============================] - 0s 11ms/step - loss: 0.3688 - accu
Epoch 9/10
18/18 [==============================] - 0s 14ms/step - loss: 0.3657 - accu
Epoch 10/10
18/18 [==============================] - 0s 11ms/step - loss: 0.3558 - accu
```

```
tf.random.set_seed(42)

model_3 = tf.keras.Sequential([
    tf.keras.layers.Dense(10,activation="relu"),
    tf.keras.layers.Dense(10,activation="relu"),
    tf.keras.layers.Dense(10,activation="relu"),
    tf.keras.layers.Dense(10,activation="relu"),
    tf.keras.layers.Dense(1,activation="sigmoid")
])

model_3.compile(loss=tf.keras.losses.BinaryCrossentropy(), # binary since we are
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_3 = model_3.fit(X_train, y_train, epochs=10,validation_data=(X_test,y_te
```

```
    Epoch 1/10
    18/18 [==============================] - 4s 23ms/step - loss: 0.8206 - accu
    Epoch 2/10
    18/18 [==============================] - 0s 10ms/step - loss: 0.6595 - accu
    Epoch 3/10
    18/18 [==============================] - 0s 10ms/step - loss: 0.6027 - accu
    Epoch 4/10
    18/18 [==============================] - 0s 8ms/step - loss: 0.6039 - accur
    Epoch 5/10
    18/18 [==============================] - 0s 10ms/step - loss: 0.6015 - accu
    Epoch 6/10
    18/18 [==============================] - 0s 12ms/step - loss: 0.5647 - accu
    Epoch 7/10
    18/18 [==============================] - 0s 12ms/step - loss: 0.5500 - accu
    Epoch 8/10
    18/18 [==============================] - 0s 12ms/step - loss: 0.5302 - accu
    Epoch 9/10
    18/18 [==============================] - 0s 10ms/step - loss: 0.5066 - accu
    Epoch 10/10
    18/18 [==============================] - 0s 18ms/step - loss: 0.4726 - accu
```

```
tf.random.set_seed(42)

model_4 = tf.keras.Sequential([
    tf.keras.layers.Dense(50,activation="relu"),
    tf.keras.layers.Dense(20,activation="relu"),
    tf.keras.layers.Dense(1,activation="sigmoid")
])

model_4.compile(loss=tf.keras.losses.BinaryCrossentropy(), # binary since we are
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_4 = model_4.fit(X_train, y_train, epochs=150,validation_data=(X_test,y_t
```

```
    Epoch 1/150
```

```
18/18 [==============================] - 3s 28ms/step - loss: 0.7482 - accu
Epoch 2/150
18/18 [==============================] - 0s 10ms/step - loss: 0.5059 - accu
Epoch 3/150
18/18 [==============================] - 0s 13ms/step - loss: 0.4259 - accu
Epoch 4/150
18/18 [==============================] - 0s 8ms/step - loss: 0.3987 - accur
Epoch 5/150
18/18 [==============================] - 0s 9ms/step - loss: 0.4885 - accur
Epoch 6/150
18/18 [==============================] - 0s 13ms/step - loss: 0.4057 - accu
Epoch 7/150
18/18 [==============================] - 0s 12ms/step - loss: 0.3623 - accu
Epoch 8/150
18/18 [==============================] - 0s 9ms/step - loss: 0.3565 - accur
Epoch 9/150
18/18 [==============================] - 0s 8ms/step - loss: 0.3192 - accur
Epoch 10/150
18/18 [==============================] - 0s 8ms/step - loss: 0.3129 - accur
Epoch 11/150
18/18 [==============================] - 0s 9ms/step - loss: 0.3212 - accur
Epoch 12/150
18/18 [==============================] - 0s 11ms/step - loss: 0.2864 - accu
Epoch 13/150
18/18 [==============================] - 0s 8ms/step - loss: 0.2731 - accur
Epoch 14/150
18/18 [==============================] - 0s 10ms/step - loss: 0.3017 - accu
Epoch 15/150
18/18 [==============================] - 0s 12ms/step - loss: 0.2724 - accu
Epoch 16/150
18/18 [==============================] - 0s 9ms/step - loss: 0.2753 - accur
Epoch 17/150
18/18 [==============================] - 0s 8ms/step - loss: 0.2637 - accur
Epoch 18/150
18/18 [==============================] - 0s 7ms/step - loss: 0.2424 - accur
Epoch 19/150
18/18 [==============================] - 0s 8ms/step - loss: 0.2425 - accur
Epoch 20/150
18/18 [==============================] - 0s 8ms/step - loss: 0.2714 - accur
Epoch 21/150
18/18 [==============================] - 0s 10ms/step - loss: 0.2634 - accu
Epoch 22/150
18/18 [==============================] - 0s 10ms/step - loss: 0.2386 - accu
Epoch 23/150
18/18 [==============================] - 0s 11ms/step - loss: 0.2287 - accu
Epoch 24/150
18/18 [==============================] - 0s 17ms/step - loss: 0.2231 - accu
Epoch 25/150
18/18 [==============================] - 0s 13ms/step - loss: 0.2530 - accu
Epoch 26/150
18/18 [==============================] - 0s 16ms/step - loss: 0.3027 - accu
Epoch 27/150
18/18 [==============================] - 0s 15ms/step - loss: 0.2247 - accu
Epoch 28/150
18/18 [==============================] - 0s 17ms/step - loss: 0.2193 - accu
Epoch 29/150
18/18 [==============================] - 0s 13ms/step - loss: 0.2485 - accu
```

Epoch 30/150

```
tf.random.set_seed(42)

model_4 = tf.keras.Sequential([
    tf.keras.layers.Dense(150,activation="relu"),
    tf.keras.layers.Dense(50,activation="relu"),
    tf.keras.layers.Dense(20,activation="relu"),
    tf.keras.layers.Dense(1,activation="sigmoid")
],name="model_4")

model_4.compile(loss=tf.keras.losses.BinaryCrossentropy(), # binary since we are
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_4 = model_4.fit(X_train, y_train, epochs=150,validation_data=(X_test,y_t
```

```
18/18 [==============================] - 0s 6ms/step - loss: 0.1340 - accur
Epoch 99/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1402 - accur
Epoch 100/150
18/18 [==============================] - 1s 33ms/step - loss: 0.1199 - accu
Epoch 101/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1480 - accur
Epoch 102/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1162 - accur
Epoch 103/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1245 - accur
Epoch 104/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1069 - accur
Epoch 105/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1293 - accur
Epoch 106/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1424 - accur
Epoch 107/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1130 - accur
Epoch 108/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1108 - accur
Epoch 109/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1185 - accur
Epoch 110/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1250 - accur
Epoch 111/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1087 - accur
Epoch 112/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1213 - accur
Epoch 113/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1151 - accur
Epoch 114/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1339 - accur
Epoch 115/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1576 - accur
Epoch 116/150
18/18 [==============================] - 0s 5ms/step - loss: 0.1508 - accur
Epoch 117/150
18/18 [==============================] - 1s 33ms/step - loss: 0.1058 - accu
```

```
Epoch 118/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1194 - accur
Epoch 119/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1323 - accur
Epoch 120/150
18/18 [==============================] - 1s 34ms/step - loss: 0.1156 - accu
Epoch 121/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1107 - accur
Epoch 122/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1244 - accur
Epoch 123/150
18/18 [==============================] - 0s 7ms/step - loss: 0.1086 - accur
Epoch 124/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1587 - accur
Epoch 125/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1247 - accur
Epoch 126/150
18/18 [==============================] - 1s 33ms/step - loss: 0.1002 - accu
Epoch 127/150
18/18 [==============================] - 0s 6ms/step - loss: 0.1006 - accur
```

```
model_4.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.1020 - accurac
[0.10199040919542313, 0.9503546357154846]
```

```
model_4 = tf.keras.models.load_model("model_experiments/model_4/")
model_4.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0774 - accurac
[0.07742727547883987, 0.978723406791687]
```

```
tf.random.set_seed(42)

model_5 = tf.keras.Sequential([
    tf.keras.layers.Dense(250,activation="relu"),
    tf.keras.layers.Dense(150,activation="relu"),
    tf.keras.layers.Dense(120,activation="relu"),
    tf.keras.layers.Dense(1,activation="sigmoid")
],name="model_5")

model_5.compile(loss=tf.keras.losses.BinaryCrossentropy(), # binary since we are
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_5 = model_5.fit(X_train, y_train, epochs=250,validation_data=(X_test,y_t
```

```
Epoch 1/250
18/18 [==============================] - 2s 50ms/step - loss: 0.7174 - accu
Epoch 2/250
18/18 [==============================] - 1s 50ms/step - loss: 0.5486 - accu
Epoch 3/250
```

```
18/18 [==============================] - 2s 100ms/step - loss: 0.4360 - acc
Epoch 4/250
18/18 [==============================] - 0s 14ms/step - loss: 0.4167 - accu
Epoch 5/250
18/18 [==============================] - 0s 13ms/step - loss: 0.4765 - accu
Epoch 6/250
18/18 [==============================] - 1s 48ms/step - loss: 0.4324 - accu
Epoch 7/250
18/18 [==============================] - 0s 18ms/step - loss: 0.3674 - accu
Epoch 8/250
18/18 [==============================] - 0s 13ms/step - loss: 0.4116 - accu
Epoch 9/250
18/18 [==============================] - 1s 48ms/step - loss: 0.3144 - accu
Epoch 10/250
18/18 [==============================] - 1s 35ms/step - loss: 0.3154 - accu
Epoch 11/250
18/18 [==============================] - 1s 35ms/step - loss: 0.3330 - accu
Epoch 12/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3016 - accur
Epoch 13/250
18/18 [==============================] - 1s 32ms/step - loss: 0.2588 - accu
Epoch 14/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3530 - accur
Epoch 15/250
18/18 [==============================] - 0s 5ms/step - loss: 0.2659 - accur
Epoch 16/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2899 - accur
Epoch 17/250
18/18 [==============================] - 1s 33ms/step - loss: 0.2552 - accu
Epoch 18/250
18/18 [==============================] - 0s 5ms/step - loss: 0.2408 - accur
Epoch 19/250
18/18 [==============================] - 0s 7ms/step - loss: 0.2374 - accur
Epoch 20/250
18/18 [==============================] - 0s 5ms/step - loss: 0.2875 - accur
Epoch 21/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2890 - accur
Epoch 22/250
18/18 [==============================] - 1s 32ms/step - loss: 0.2381 - accu
Epoch 23/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2251 - accur
Epoch 24/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2073 - accur
Epoch 25/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2610 - accur
Epoch 26/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3184 - accur
Epoch 27/250
18/18 [==============================] - 1s 34ms/step - loss: 0.2327 - accu
Epoch 28/250
18/18 [==============================] - 0s 5ms/step - loss: 0.2033 - accur
Epoch 29/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2586 - accur
Epoch 30/250
```

```
model_5.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 3ms/step - loss: 0.0667 - accurac
[0.06672469526529312, 0.978723406791687]
```

```
model_5 = tf.keras.models.load_model("model_experiments/model_5/")
model_5.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0612 - accurac
[0.06118996813893318, 0.978723406791687]
```

```
tf.random.set_seed(42)

model_6 = tf.keras.Sequential([
    tf.keras.layers.Dense(250,activation="relu"),
    tf.keras.layers.Dense(150,activation="relu"),
    tf.keras.layers.Dense(120,activation="relu"),
    tf.keras.layers.Dense(20,activation="relu"),
    tf.keras.layers.Dense(1,activation="sigmoid")
],name="model_6")

model_6.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_6 = model_6.fit(X_train, y_train, epochs=250,validation_data=(X_test,y_t
```

```
Epoch 1/250
18/18 [==============================] - 2s 72ms/step - loss: 0.5845 - accu
Epoch 2/250
18/18 [==============================] - 1s 53ms/step - loss: 0.5134 - accu
Epoch 3/250
18/18 [==============================] - 1s 56ms/step - loss: 0.4297 - accu
Epoch 4/250
18/18 [==============================] - 0s 8ms/step - loss: 0.4030 - accur
Epoch 5/250
18/18 [==============================] - 0s 8ms/step - loss: 0.4598 - accur
Epoch 6/250
18/18 [==============================] - 1s 42ms/step - loss: 0.4664 - accu
Epoch 7/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3887 - accur
Epoch 8/250
18/18 [==============================] - 0s 6ms/step - loss: 0.4232 - accur
Epoch 9/250
18/18 [==============================] - 1s 39ms/step - loss: 0.3278 - accu
Epoch 10/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3391 - accur
Epoch 11/250
18/18 [==============================] - 1s 40ms/step - loss: 0.3678 - accu
Epoch 12/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3206 - accur
Epoch 13/250
```

```
18/18 [==============================] - 1s 38ms/step - loss: 0.2840 - accu
Epoch 14/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3874 - accur
Epoch 15/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2818 - accur
Epoch 16/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3199 - accur
Epoch 17/250
18/18 [==============================] - 1s 39ms/step - loss: 0.3049 - accu
Epoch 18/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2507 - accur
Epoch 19/250
18/18 [==============================] - 0s 7ms/step - loss: 0.2533 - accur
Epoch 20/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3167 - accur
Epoch 21/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2885 - accur
Epoch 22/250
18/18 [==============================] - 1s 40ms/step - loss: 0.2523 - accu
Epoch 23/250
18/18 [==============================] - 0s 7ms/step - loss: 0.2401 - accur
Epoch 24/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2093 - accur
Epoch 25/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2864 - accur
Epoch 26/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3172 - accur
Epoch 27/250
18/18 [==============================] - 1s 39ms/step - loss: 0.2390 - accu
Epoch 28/250
18/18 [==============================] - 1s 39ms/step - loss: 0.2167 - accu
Epoch 29/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2568 - accur
Epoch 30/250
```

```
model_6.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0658 - accurac
[0.06581150740385056, 0.9716312289237976]
```

```
model_6 = tf.keras.models.load_model("model_experiments/model_6/")
model_6.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0532 - accurac
[0.0531863234937191, 0.978723406791687]
```

```
tf.random.set_seed(42)

model_7 = tf.keras.Sequential([
    tf.keras.layers.Dense(250,activation="linear"),
    tf.keras.layers.Dense(150,activation="relu"),
    tf.keras.layers.Dense(120,activation="linear"),
    tf.keras.layers.Dense(20,activation="tanh"),
    tf.keras.layers.Dense(1,activation="sigmoid")
```

```
],name="model_7")

model_7.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                optimizer=tf.keras.optimizers.Adam(lr=0.0001),
                metrics=['accuracy'])

history_7 = model_7.fit(X_train, y_train, epochs=250,validation_data=(X_test,y_t
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_ra
Epoch 1/250
18/18 [==============================] - 4s 96ms/step - loss: 0.5567 - accu
Epoch 2/250
18/18 [==============================] - 1s 61ms/step - loss: 0.4466 - accu
Epoch 3/250
18/18 [==============================] - 1s 45ms/step - loss: 0.3959 - accu
Epoch 4/250
18/18 [==============================] - 1s 42ms/step - loss: 0.3420 - accu
Epoch 5/250
18/18 [==============================] - 0s 6ms/step - loss: 0.3165 - accur
Epoch 6/250
18/18 [==============================] - 1s 42ms/step - loss: 0.3049 - accu
Epoch 7/250
18/18 [==============================] - 1s 43ms/step - loss: 0.3088 - accu
Epoch 8/250
18/18 [==============================] - 1s 42ms/step - loss: 0.2548 - accu
Epoch 9/250
18/18 [==============================] - 1s 43ms/step - loss: 0.2410 - accu
Epoch 10/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2227 - accur
Epoch 11/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2176 - accur
Epoch 12/250
18/18 [==============================] - 1s 42ms/step - loss: 0.2122 - accu
Epoch 13/250
18/18 [==============================] - 1s 43ms/step - loss: 0.2296 - accu
Epoch 14/250
18/18 [==============================] - 0s 6ms/step - loss: 0.1967 - accur
Epoch 15/250
18/18 [==============================] - 1s 42ms/step - loss: 0.2142 - accu
Epoch 16/250
18/18 [==============================] - 1s 43ms/step - loss: 0.1898 - accu
Epoch 17/250
18/18 [==============================] - 1s 68ms/step - loss: 0.1608 - accu
Epoch 18/250
18/18 [==============================] - 1s 43ms/step - loss: 0.1661 - accu
Epoch 19/250
18/18 [==============================] - 0s 5ms/step - loss: 0.1537 - accur
Epoch 20/250
18/18 [==============================] - 0s 5ms/step - loss: 0.2290 - accur
Epoch 21/250
18/18 [==============================] - 0s 6ms/step - loss: 0.2103 - accur
Epoch 22/250
18/18 [==============================] - 0s 9ms/step - loss: 0.1903 - accur
Epoch 23/250
18/18 [==============================] - 0s 8ms/step - loss: 0.1691 - accur
Epoch 24/250
```

```
18/18 [==============================] - 1s 63ms/step - loss: 0.1553 - accu
Epoch 25/250
18/18 [==============================] - 0s 8ms/step - loss: 0.1636 - accur
Epoch 26/250
18/18 [==============================] - 1s 56ms/step - loss: 0.1790 - accu
Epoch 27/250
18/18 [==============================] - 0s 6ms/step - loss: 0.1451 - accur
Epoch 28/250
18/18 [==============================] - 1s 44ms/step - loss: 0.1277 - accu
Epoch 29/250
18/18 [==============================] - 1s 42ms/step - loss: 0.1221 - accu
```

```
model_7.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 6ms/step - loss: 0.0513 - accurac
[0.05128715559840202, 0.9929078221321106]
```

```
model_7 = tf.keras.models.load_model("model_experiments/model_7/")
model_7.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0276 - accurac
[0.027588030323386192, 0.9929078221321106]
```

```
tf.random.set_seed(42)

model_8 = tf.keras.Sequential([
    tf.keras.layers.Dense(250,activation="relu"),
    tf.keras.layers.Dense(150,activation="relu"),
    tf.keras.layers.Dense(120,activation="relu"),
    tf.keras.layers.Dense(20,activation="tanh"),
    tf.keras.layers.Dense(1,activation="sigmoid")
],name="model_8")

model_8.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                optimizer=tf.keras.optimizers.SGD(),
                metrics=['accuracy'])

history_8 = model_8.fit(X_train, y_train, epochs=350,validation_data=(X_test,y_t
                        callbacks=[create_model_checkpoint(model_name=model_8.na
```

```
Epoch 1/350
18/18 [==============================] - 2s 51ms/step - loss: 0.5878 - accu
Epoch 2/350
18/18 [==============================] - 1s 39ms/step - loss: 0.4567 - accu
Epoch 3/350
18/18 [==============================] - 1s 39ms/step - loss: 0.3957 - accu
Epoch 4/350
18/18 [==============================] - 0s 6ms/step - loss: 0.3728 - accur
Epoch 5/350
18/18 [==============================] - 0s 6ms/step - loss: 0.4199 - accur
Epoch 6/350
18/18 [==============================] - 1s 39ms/step - loss: 0.3798 - accu
Epoch 7/350
```

```
18/18 [==============================] - 0s 6ms/step - loss: 0.3396 - accur
Epoch 8/350
18/18 [==============================] - 0s 6ms/step - loss: 0.3493 - accur
Epoch 9/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2957 - accur
Epoch 10/350
18/18 [==============================] - 1s 38ms/step - loss: 0.3122 - accu
Epoch 11/350
18/18 [==============================] - 1s 38ms/step - loss: 0.3030 - accu
Epoch 12/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2744 - accur
Epoch 13/350
18/18 [==============================] - 0s 5ms/step - loss: 0.2794 - accur
Epoch 14/350
18/18 [==============================] - 0s 6ms/step - loss: 0.3056 - accur
Epoch 15/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2887 - accur
Epoch 16/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2742 - accur
Epoch 17/350
18/18 [==============================] - 1s 38ms/step - loss: 0.2791 - accu
Epoch 18/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2341 - accur
Epoch 19/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2404 - accur
Epoch 20/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2708 - accur
Epoch 21/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2573 - accur
Epoch 22/350
18/18 [==============================] - 1s 38ms/step - loss: 0.2470 - accu
Epoch 23/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2215 - accur
Epoch 24/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2119 - accur
Epoch 25/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2665 - accur
Epoch 26/350
18/18 [==============================] - 0s 6ms/step - loss: 0.2948 - accur
Epoch 27/350
18/18 [==============================] - 1s 71ms/step - loss: 0.2260 - accu
Epoch 28/350
18/18 [==============================] - 0s 11ms/step - loss: 0.2111 - accu
Epoch 29/350
18/18 [==============================] - 0s 9ms/step - loss: 0.2508 - accur
Epoch 30/350
```

```
model_8.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.1486 - accurac
[0.1485973447561264, 0.9290780425071716]
```

```
model_8 = tf.keras.models.load_model("model_experiments/model_8/")
model_8.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 5ms/step - loss: 0.0483 - accurac
[0.04832042381167412, 0.978723406791687]
```

```

```

```
tf.random.set_seed(42)

final = tf.keras.Sequential([
    tf.keras.layers.Dense(250,activation="linear"),
    tf.keras.layers.Dense(50,activation="relu"),
    tf.keras.layers.Dense(120,activation="linear"),
    tf.keras.layers.Dense(20,activation="tanh"),
    tf.keras.layers.Dense(1,activation="sigmoid")
],name="final")

final.compile(loss=tf.keras.losses.BinaryCrossentropy(),
                optimizer=tf.keras.optimizers.Adam(lr=0.0001),
                metrics=['accuracy'])

final_history = final.fit(X_train, y_train, epochs=300,validation_data=(X_test,y
                        callbacks=[create_model_checkpoint(model_name=final.name
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_ra
Epoch 1/300
18/18 [==============================] - 3s 56ms/step - loss: 0.4843 - accu
Epoch 2/300
18/18 [==============================] - 1s 46ms/step - loss: 0.3269 - accu
Epoch 3/300
18/18 [==============================] - 1s 43ms/step - loss: 0.2479 - accu
Epoch 4/300
18/18 [==============================] - 0s 7ms/step - loss: 0.2859 - accur
Epoch 5/300
18/18 [==============================] - 0s 7ms/step - loss: 0.2880 - accur
Epoch 6/300
18/18 [==============================] - 1s 45ms/step - loss: 0.2164 - accu
Epoch 7/300
18/18 [==============================] - 0s 6ms/step - loss: 0.2175 - accur
Epoch 8/300
18/18 [==============================] - 1s 43ms/step - loss: 0.1890 - accu
Epoch 9/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1677 - accur
Epoch 10/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1671 - accur
Epoch 11/300
18/18 [==============================] - 0s 7ms/step - loss: 0.1529 - accur
Epoch 12/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1745 - accur
Epoch 13/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1505 - accur
```

```
Epoch 14/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1781 - accur
Epoch 15/300
18/18 [==============================] - 1s 43ms/step - loss: 0.1388 - accu
Epoch 16/300
18/18 [==============================] - 1s 43ms/step - loss: 0.1368 - accu
Epoch 17/300
18/18 [==============================] - 1s 42ms/step - loss: 0.1127 - accu
Epoch 18/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1157 - accur
Epoch 19/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1129 - accur
Epoch 20/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1217 - accur
Epoch 21/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1336 - accur
Epoch 22/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1174 - accur
Epoch 23/300
18/18 [==============================] - 0s 7ms/step - loss: 0.1493 - accur
Epoch 24/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1179 - accur
Epoch 25/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1124 - accur
Epoch 26/300
18/18 [==============================] - 0s 6ms/step - loss: 0.1886 - accur
Epoch 27/300
18/18 [==============================] - 1s 53ms/step - loss: 0.1203 - accu
Epoch 28/300
18/18 [==============================] - 0s 8ms/step - loss: 0.0949 - accur
Epoch 29/300
18/18 [==============================] - 0s 8ms/step - loss: 0.0942 - accur
```

```
final.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0050 - accurac
[0.004958361387252808, 1.0]
```

```
final = tf.keras.models.load_model("model_experiments/final/")
final.evaluate(X_test,y_test)
```

```
5/5 [==============================] - 0s 4ms/step - loss: 0.0039 - accurac
[0.003913191147148609, 1.0]
```

```
final.save("Final_3.h5")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3079:
  saving_api.save_model(
```

4. FLASK INTEGRATION

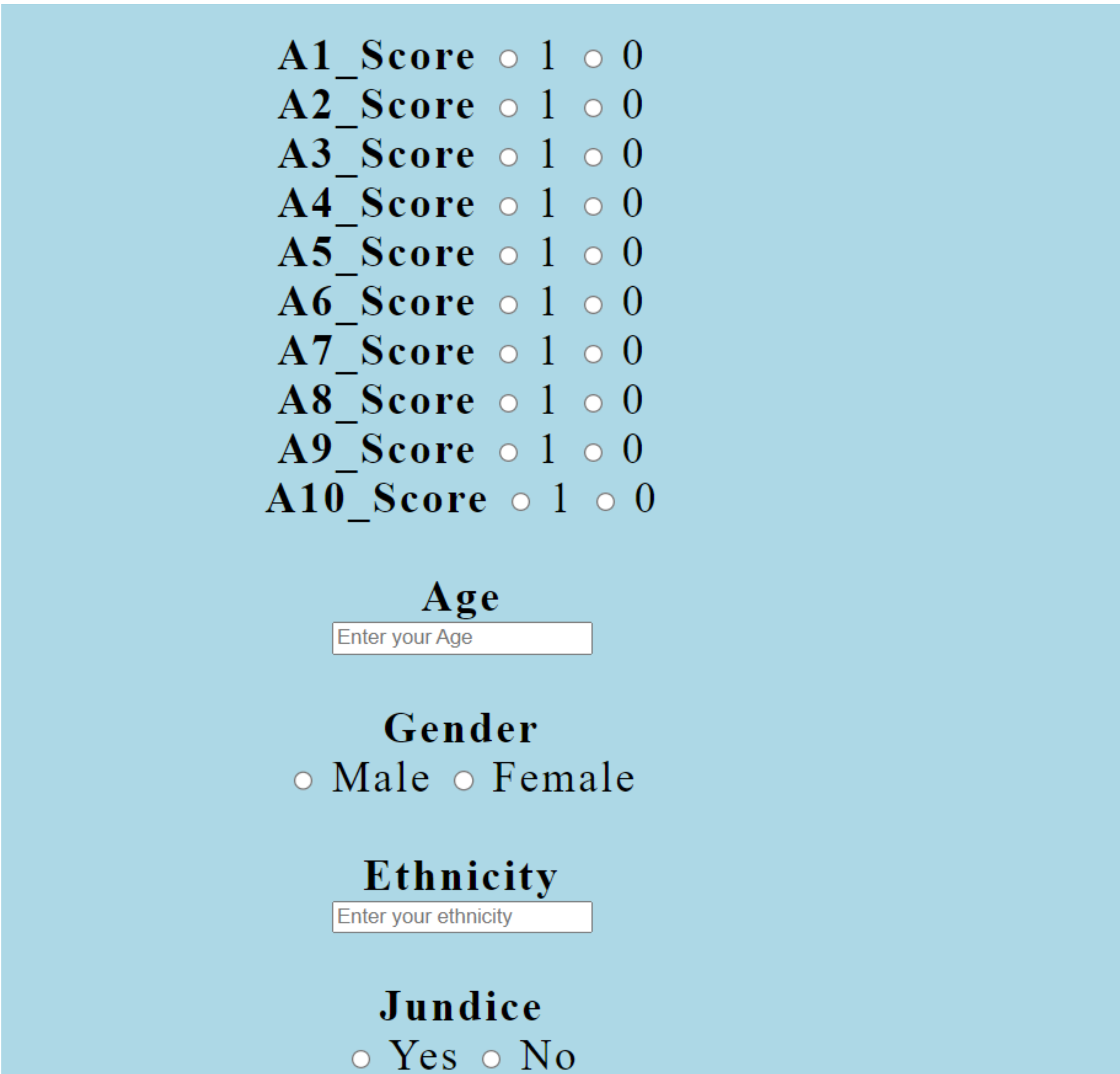
```
import joblib
from flask import Flask, jsonify, abort, make_response, request,
render_template
from flask import
Flask,render_template,url_for,request,send_from_directory

model = joblib.load("RF_uncompressed.joblib")

app = Flask(__name__)

tasks = [
    {
        'id': 1,
        'title': u'Buy groceries',
        'description': u'Milk, Cheese, Pizza, Fruit, Tylenol',
        'done': False
    },
    {
        'id': 2,
        'title': u'Learn Python',
        'description': u'Need to find a good Python tutorial on the
web',
        'done': False
    }
]

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

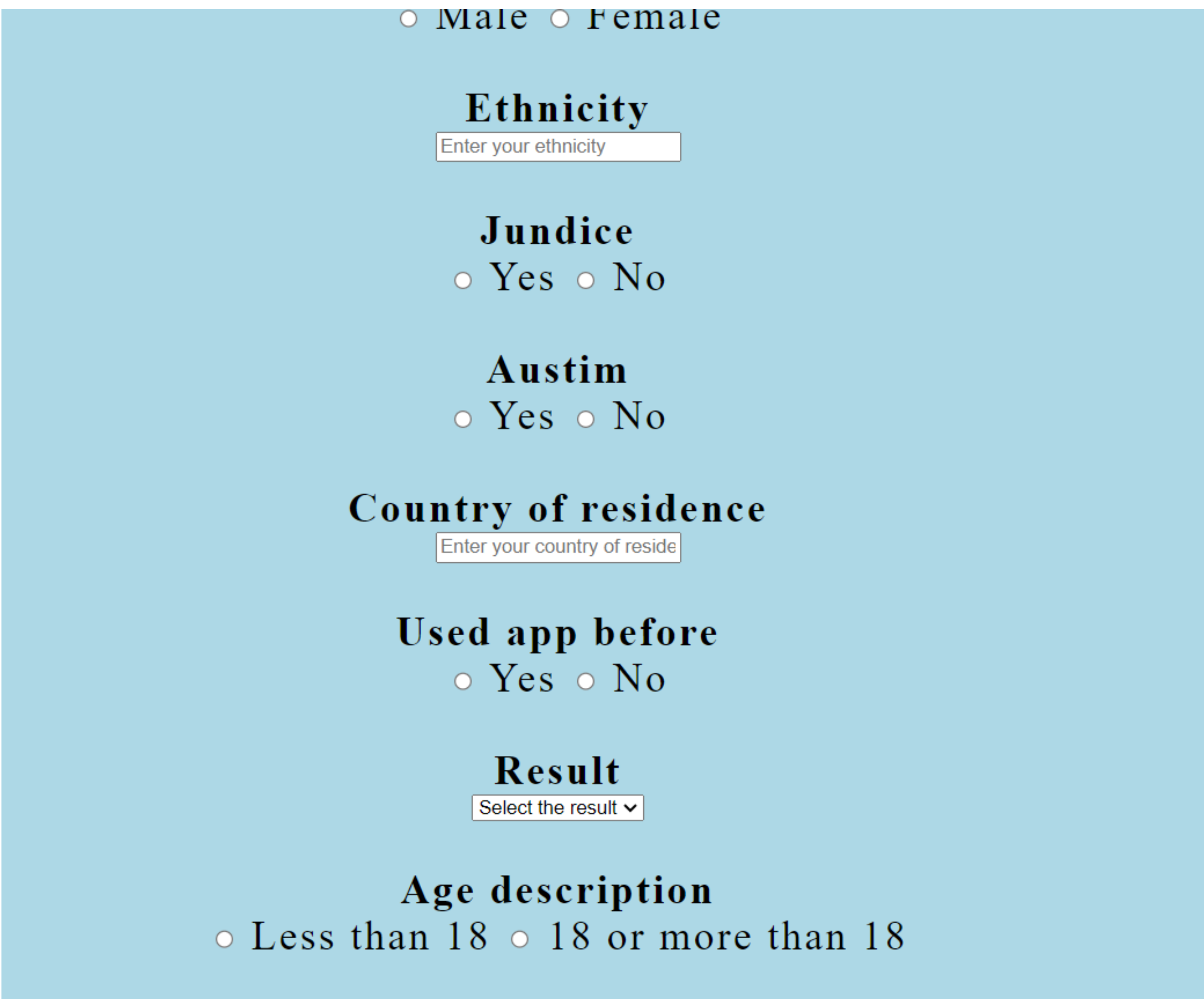
```
@app.route('/tasks', methods=['POST'])
def predict():
    A1_Score=request.form['A1_Score']
    A2_Score=request.form['A2_Score']
    A3_Score=request.form['A3_Score']
    A4_Score=request.form['A4_Score']
    A5_Score=request.form['A5_Score']
    A6_Score=request.form['A6_Score']
    A7_Score=request.form['A7_Score']
    A8_Score=request.form['A8_Score']
    A9_Score=request.form['A9_Score']
    A10_Score=request.form['A10_Score']
    age=request.form['age']
    result=request.form['result']
    Jundice=request.form['Jundice']
    Austim=request.form['Austim']
    Ethnicity=request.form['Ethnicity']

X=[[int(request.form['A1_score']),int(request.form['A2_score']),int(r
equest.form['A3_score']),int(request.form['A4_score']),int(request.fo
rm['A5_score']),int(request.form['A6_score']),int(request.form['A7_sc
ore']),int(request.form['A8_score']),int(request.form['A9_score']),in
t(request.form['A10_score']),]]
    prediction=model.predict(X)[0]
    return render_template('predict.html',prediction='Detected_YES
{}'.format(X))


@app.route('/tasks/<int:task_id>', methods=['GET'])
def get_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
```

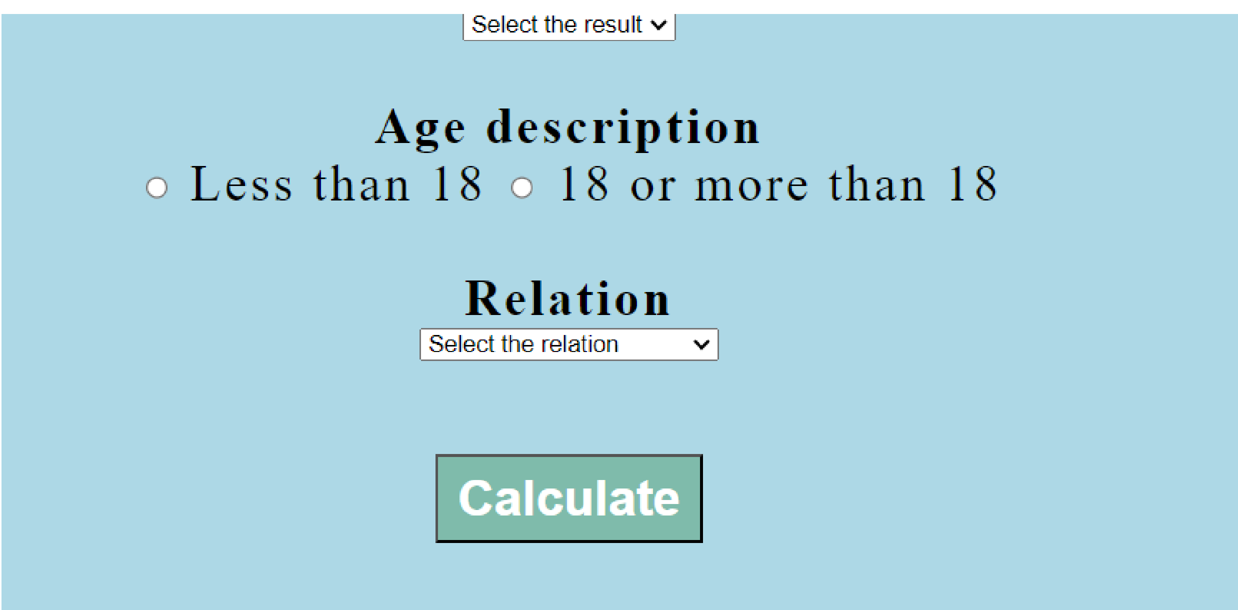
```
        abort(404)
    return jsonify({'task': task[0]})


@app.route('/tasks', methods=['POST'])
def create_task():
    if not request.json or not 'title' in request.json:
        abort(400)
    task = {
        'id': tasks[-1]['id'] + 1,
        'title': request.json['title'],
        'description': request.json.get('description', ""),
        'done': False
    }
    tasks.append(task)
    return jsonify({'task': task}), 201


@app.route('/tasks/<int:task_id>', methods=['PUT'])
def update_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    if not request.json:
        abort(400)
    if 'title' in request.json and type(request.json['title']) !=
unicode:
        abort(400)
    if 'description' in request.json and
type(request.json['description']) is not unicode:
        abort(400)
```

```python
    if 'done' in request.json and type(request.json['done']) is not bool:
        abort(400)
    task[0]['title'] = request.json.get('title', task[0]['title'])
    task[0]['description'] = request.json.get('description', task[0]['description'])
    task[0]['done'] = request.json.get('done', task[0]['done'])
    return jsonify({'task': task[0]})


@app.route('/tasks/<int:task_id>', methods=['DELETE'])
def delete_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    tasks.remove(task[0])
    return jsonify({'result': True})


@app.errorhandler(404)
def not_found(error):
    return make_response(jsonify({'error': 'Not found'}), 404)


if __name__ == '__main__':
    app.run(debug=True)
```

This Python code constitutes a Flask web application designed for autism detection. It integrates a machine learning model, loaded from "RF_uncompressed.joblib," to predict autism based on user-provided information. The application includes routes for the homepage ('/'), where users can likely find a form for inputting data relevant to autism diagnosis. The form submits this data to the '/tasks' route, where the machine learning model processes the input parameters, such as scores on various scales, age, and additional factors. The prediction results, indicating whether autism is detected or not, are then displayed on a 'predict.html' template. The code also handles common HTTP errors, such as 404 (Not Found) and 400 (Bad Request), and the application can be run in debug mode for local testing.




Male Female
Ethnicity
Enter your ethnicity
Jundice
Yes No
Austim
Yes No
Country of residence
Enter your country of reside
Used app before
Yes No
Result
Select the result
Age description
Less than 18 18 or more than 18


Select the result
Age description
Less than 18 18 or more than 18
Relation
Select the relation
Calculate