MACHINE LEARNING – CS 584

Leveraging Machine Learning for Autism Spectrum Disorder (ASD) Detection

PROJECT REPORT

Choladevi Gheereddy - A20544476
Manogna Vadlamudi - A20551908
Bollepalli Tejaswini - A20562616

Professor
Stephan Avsec

Leveraging Machine Learning for Autism Spectrum Disorder (ASD) Detection

1. Abstract

Autistic Spectrum Disorder (ASD) encompasses a range of developmental disorders affecting the nervous system, with symptoms spanning from mild to severe. These symptoms include challenges in language development, difficulties in social interaction, and repetitive behaviours, often accompanied by additional conditions such as anxiety, mood disorders, and ADHD. The healthcare sector grapples with substantial economic implications due to the growing prevalence of ASD cases and the extensive resources required for diagnosis. Early detection is paramount, as it facilitates timely intervention and mitigates the long-term financial burdens associated with delayed diagnosis. Consequently, healthcare professionals worldwide are in dire need of efficient and accessible ASD screening methods capable of accurately predicting the likelihood of ASD based on specific measured characteristics, thereby guiding individuals in their decision to pursue formal clinical diagnosis.

Despite the pressing demand for improved ASD screening methods, significant challenges persist. Research endeavours are hampered by the scarcity of comprehensive datasets containing detailed behavioural traits and demographic factors such as age, gender, and ethnicity. These datasets are essential for refining the efficiency, sensitivity, specificity, and predictive accuracy of ASD screening processes. Presently, available autism-related datasets primarily focus on genetic data, which are not only sensitive but also challenging to obtain due to privacy concerns and stringent regulatory frameworks. As a result, the shortage of suitable datasets complicates efforts to conduct thorough analyses and develop more effective screening tools for ASD.

2. Problem Statement

The problem statement involves using available ASD data to predict whether new patients can be classified into two categories: either "patient has ASD" or "patient does not have ASD." This is essentially a binary classification problem aimed at determining the likelihood of ASD based on individual characteristics. The approach relies on supervised machine learning techniques, where models learn from labelled data, with a set of data containing correct answers for the models to learn from. Additionally, a feature selection algorithm will be applied to identify which variables among the 20 are most significant in determining ASD presence.

3. Literature Review

The study explores various supervised machine learning classification techniques, including Decision Trees, Random Forests, Logistic Regression. The aim is to identify the most effective method or combination of classifiers (Ensemble Learning) to accurately predict ASD presence. Performance evaluation will be conducted using standard metrics, and the strengths and weaknesses of each model will be analysed. Ultimately, the goal is to construct a model that accurately predicts ASD presence based on individual characteristics, utilizing the mentioned methods for evaluation and comparison.

Technologies and Libraries

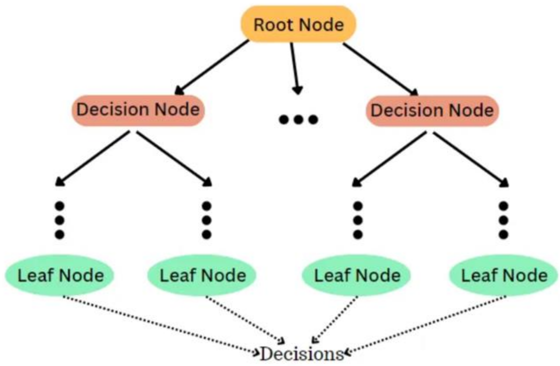
- Python: Programming language.
- pandas and NumPy: For data manipulation and numerical calculations.
- matplotlib and seaborn: For data visualization.

3.1. Algorithms and Techniques

Decision Tree

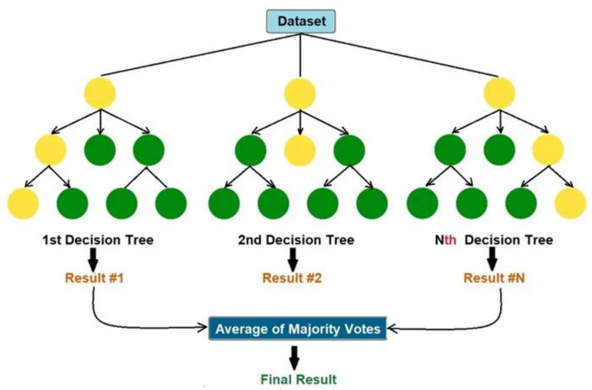
A decision tree is a supervised machine learning algorithm that partitions the feature space into a tree-like structure by recursively splitting the data based on feature attributes. Each internal node in the tree represents a decision based on a feature, while each leaf node represents the outcome or prediction. Decision trees are versatile and can be used for both classification and regression tasks, making them widely applicable across various domains.

- One advantage of decision trees is their simplicity and interpretability. The decision-making process of a decision tree is easy to visualize and understand, as it mimics human decision-making logic. Additionally, decision trees can handle both numerical and categorical data without requiring extensive data pre-processing, which can save time and effort in the data preparation phase.
- However, decision trees have some limitations. They are prone to overfitting, especially when the tree grows too deep or when dealing with noisy data. Overfitting occurs when the model captures noise or irrelevant patterns in the training data, leading to poor generalization performance on unseen data. To mitigate overfitting, techniques such as pruning, limiting the tree depth, and using ensemble methods like random forests can be employed.



Random Forest

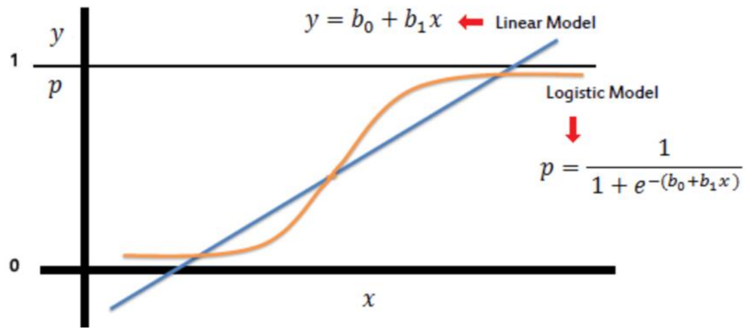
- Random Forest is an ensemble learning technique based on decision trees. It constructs a multitude of decision trees during training and outputs the mode (for classification) or average prediction (for regression) of the individual trees. The randomness comes from using bootstrap sampling of the training data and random feature selection at each split of the trees. This helps to decorrelate the trees and improve the model's performance.



- One significant advantage of Random Forest is its robustness to overfitting. By combining multiple decision trees, Random Forest reduces the risk of overfitting compared to individual decision trees. However, Random Forest also has some limitations. The model's interpretability may be reduced compared to individual decision trees since it involves multiple trees. Furthermore, Random Forest can be computationally expensive, especially when dealing with a large number of trees and features.

Logistic Regression

- Logistic regression is a statistical method used for binary classification tasks, where the target variable (y) is categorical and has only two possible outcomes, typically coded as 0 and 1. Despite its name, logistic regression is a classification algorithm rather than a regression algorithm. It models the relationship between the independent variables(x0,x1,x2,…xn) (features) and the probability of a particular outcome occurring using a logistic function.



- One advantage of logistic regression is its simplicity, interpretability and low computational cost. The coefficients in logistic regression provide insight into the relationship between the features and the outcome, allowing for easy interpretation of the model.

3.2 Model Evaluation Metrics:

To measure the effectiveness of each above-mentioned classification models we will study the accuracy score along with the precision, recall, F-Beta Score.

- **Accuracy**: Accuracy is a measure of the overall correctness of the model's predictions. It represents the ratio of correctly predicted instances (both positive and negative) to the total number of instances evaluated.

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

- **Precision**: Precision is the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives and false positives).

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

- **Recall (also known as Sensitivity or True Positive Rate):** Recall is the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives and false negatives).

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

- **F-1 Score:** The F-1 score is a weighted harmonic mean of precision and recall, F1 score, giving equal weight to precision and recall.

$$F1 = 2 \times \left(\frac{\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}\right)$$

3.3 Handling Overfitting and Under-fitting

- **Overfitting**: Overfitting occurs when a model learns the training data too well, capturing noise or random fluctuations in the data instead of the underlying pattern. As a result, the model performs well on the training data but poorly on unseen data. Signs of overfitting include excessively high accuracy on the training set, but lower accuracy on the validation or test set. The model may exhibit high variance.

- **Underfitting**: Underfitting occurs when a model is too simple to capture the underlying structure of the data. The model may fail to learn from the training data and generalize poorly to new, unseen data. Signs of underfitting include poor performance on both the training and validation or test sets, as well as overly simplistic decision boundaries or high bias.

3.4 Bias-Variance trade-off:

- A basic idea in machine learning is the bias-variance trade-off. It speaks about the harmony between variance and bias, which influences the performance of predictive models. Making the appropriate trade-off is essential to developing models that perform well when applied to fresh data.
- The inverse relationship between bias and variance is illustrated by the bias-variance trade-off. One tends to rise while the other falls, and vice versa. Achieving the ideal balance is essential. While an excessively complicated model with high variance will fit the noise in the data, an excessively basic model with high bias will fail to identify the underlying patterns.

High Bias, Low Variance: A model that has high bias and low variance is considered to be **underfitting**.
High Variance, Low Bias: A model that has high variance and low bias is considered to be **overfitting**.
High-Bias, High-Variance: A model with high bias and high variance cannot capture underlying patterns and is too sensitive to training data changes. On average, the model will generate unreliable and inconsistent predictions.
Low Bias, Low Variance: A model with low bias and low variance can capture data patterns and handle variations in training data. This is the **perfect scenario** for a machine learning model where it can **generalize well to unseen data** and make consistent, accurate predictions.

To address overfitting and underfitting, various techniques can be employed:

Cross-Validation: Cross-validation can help assess model performance on unseen data and detect overfitting or underfitting.

K-fold Cross Validation:

- K-fold cross-validation is a technique used to assess the performance of a machine learning model by dividing the dataset into K subsets or folds of approximately equal size. The model is trained K times, each time using K-1 folds for training and the remaining fold for validation.
- Cross-validation helps to estimate the model's performance on unseen data and detect issues such as overfitting or underfitting. It provides a more reliable estimate of the model's performance compared to a single train-test split by averaging the performance metrics across multiple iterations.
- K-fold cross-validation is particularly useful when the dataset is limited in size or when the data distribution is heterogeneous. It helps to ensure that the model's performance is

consistent across different subsets of the data and reduces the risk of biased performance estimates.

4. Data:

Dataset for autism disorder typically comprises structured information from individuals diagnosed with autism spectrum disorder (ASD) alongside a control group. It encompasses demographic details such as age, gender, and family history, coupled with clinical features like diagnostic scores from standardized tests such as the Autism Diagnostic Observation Schedule (ADOS) or behavioural traits. Additionally, medical history, genetic factors, environmental exposures, intervention records, and outcome measures contribute to a comprehensive understanding. Ethical collection practices and privacy safeguards are paramount, while addressing biases ensures the dataset's reliability for training ML models aimed at furthering our comprehension of autism spectrum disorder.

The dataset provided appears to contain information related to autism spectrum disorder (ASD). Here's a breakdown of the columns:

1. **ID**: Unique identifier for each individual in the dataset.
2. **A1_Score to A10_Score**: Scores representing responses to ten different questions or items. These scores may be related to specific behaviours, symptoms, or characteristics associated with ASD. Each score likely indicates the severity or frequency of a particular behaviour or trait.
3. **age**: Age of the individual.
4. **gender**: Gender of the individual.
5. **ethnicity**: Ethnicity of the individual.
6. **jaundice**: Binary variable indicating whether the individual had jaundice (a medical condition involving yellowing of the skin and eyes) or not.
7. **autism**: Binary variable indicating whether the individual has autism or not.
8. **country_of_res**: Country of residence of the individual.
9. **used_app_before**: Binary variable indicating whether the individual has used an app related to autism before or not.
10. **result**: Result or outcome of the assessment or evaluation conducted. This could be related to the presence or severity of ASD symptoms.
11. **age_desc**: Description of the age group the individual belongs to.
12. **relation**: Relationship of the respondent to the individual (e.g., parent, healthcare proffesionals).
13. **Class/ASD**: Binary variable indicating whether the individual has ASD or not.

    The dataset may require pre-processing steps such as handling missing values, encoding categorical variables, and scaling numerical features before being used for machine learning algorithms. Additionally, thorough analysis and understanding of the data's context would be necessary for accurate model development and interpretation.

Data Collection: We took the autism dataset (ASD) from Kaggle.

Importing Dataset

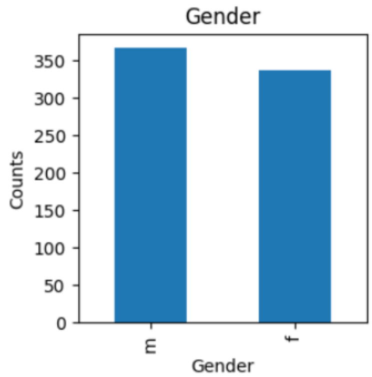
```
In [1]: #importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
data = pd.read_csv('Autism-Child-Data.csv')  #importing data
data.head(n=10)         #viewing first 10 rows
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	...	gender	ethnicity	jundice	austim	contry_of_res	used_app_before	result	age_desc	relation	Class
0	1	1	1	1	0	0	1	1	0	0	...	f	White-European	no	no	United States	no	6.0	18 and more	Self	
1	1	1	0	1	0	0	0	1	0	1	...	m	Latino	no	yes	Brazil	no	5.0	18 and more	Self	
2	1	1	0	1	1	0	1	1	1	1	...	m	Latino	yes	yes	Spain	no	8.0	18 and more	Parent	
3	1	1	0	1	0	0	1	1	0	1	...	f	White-European	no	yes	United States	no	6.0	18 and more	Self	
4	1	0	0	0	0	0	0	1	0	0	...	f	?	no	no	Egypt	no	2.0	18 and more	?	
5	1	1	1	1	1	0	1	1	1	1	...	m	Others	yes	no	United States	no	9.0	18 and more	Self	
6	0	1	0	0	0	0	0	1	0	0	...	f	Black	no	no	United States	no	2.0	18 and more	Self	
7	1	1	1	1	0	0	0	0	1	0	...	m	White-European	no	no	New Zealand	no	5.0	18 and more	Parent	
8	1	1	0	0	1	0	0	1	1	1	...	m	White-European	no	no	United States	no	6.0	18 and more	Self	
9	1	1	1	1	0	1	1	1	1	0	...	m	Asian	yes	yes	Bahamas	no	8.0	18 and more	Health care professional	

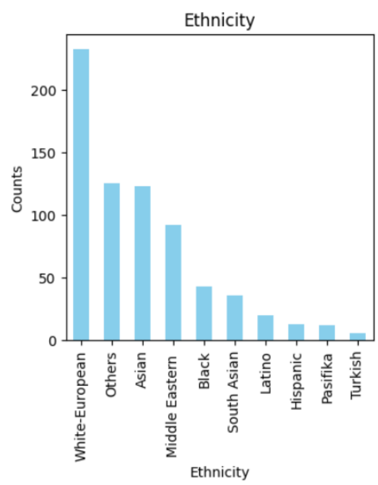
10 rows × 21 columns

Data Analysis:

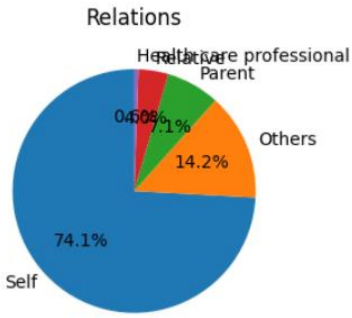
Gender: Here we observe that count of male is comparatively high than count of female.



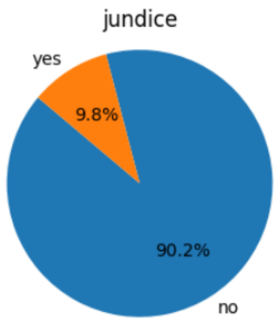
Ethnicity: Here we can observe that White-European is relatively high when compared to rest all i.e., Others, Middle Eastern, Black, South Asian, Latino, Hispanic, Pasifika and Turkish. Turkish is comparatively lower that the rest.



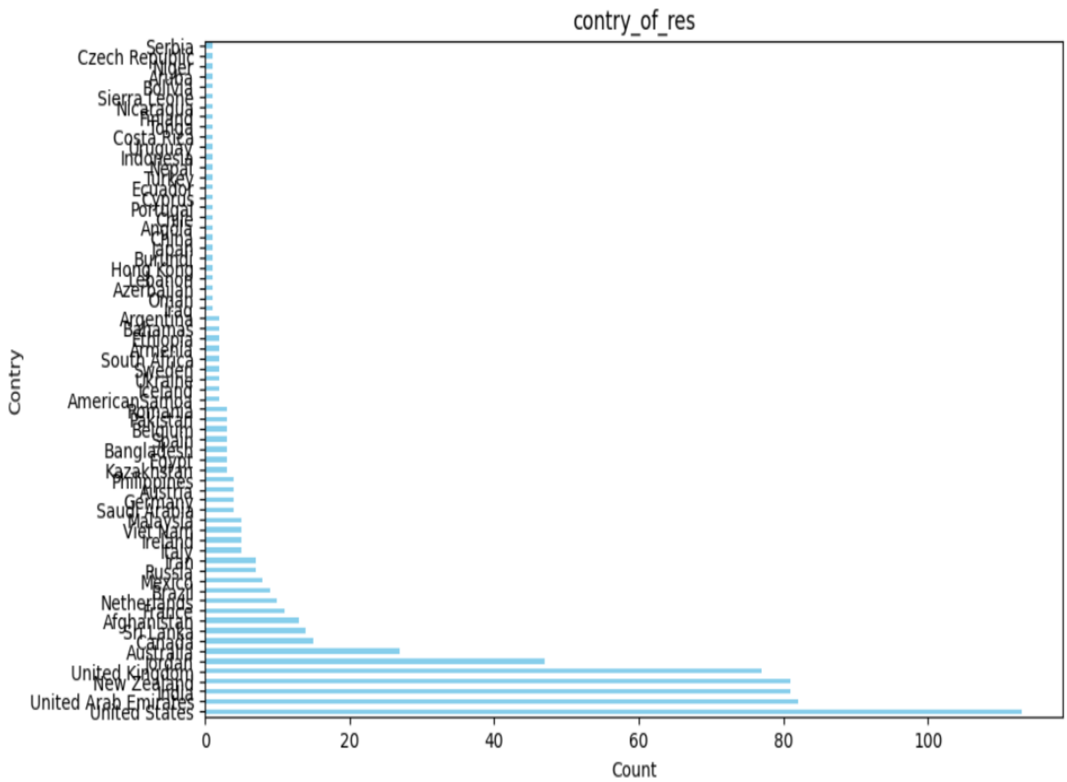
Relations: Here the relation to self is more dominant than rest other features i.e., relative, parent, other and healthcare professionals. In which relation to healthcare professionals are the least in count.



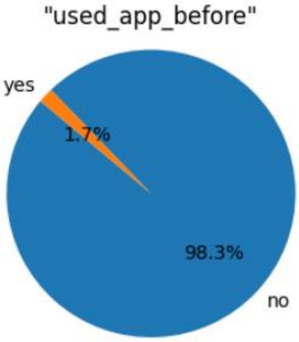
Jundice: We observe that the jundice is highly ‘no’ whereas yes in some cases i.e., 9.8%.



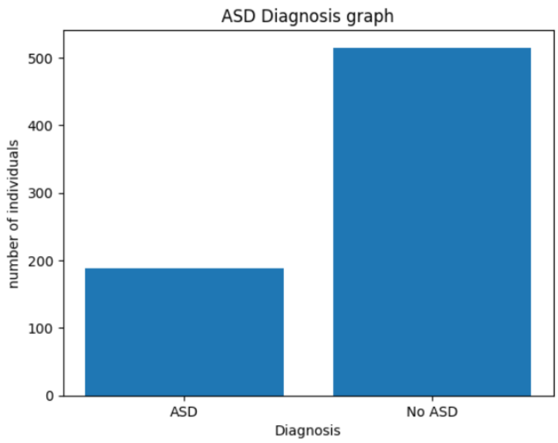
Country of residence: We observe that country of residence has high count for United states and low count for countries like Serbia, Indonesia and few others.



Used screening app: We see that participant screening app usage is highly no and yes in about 1.7%.



ASD: The classification of ASD is highly nowhere as only few in our data set have been diagnosed with ASD.



Data Cleaning:

- Handled missing values and corrected erroneous entries to ensure data quality.

s step involves handling missing values, outliers, and inconsistencies in the data. Common techniques include imputati

Cleaning Numerical Features are:

In [3]:
```
data['age'].value_counts()  # counts the occurances of each data (age) values.
```

Out[3]:
```
21.0     49
20.0     46
23.0     37
22.0     37
19.0     35
24.0     34
27.0     31
18.0     31
30.0     30
26.0     28
```

Outlier Identification: Using the below we have identified an outlier that is between 350-400.

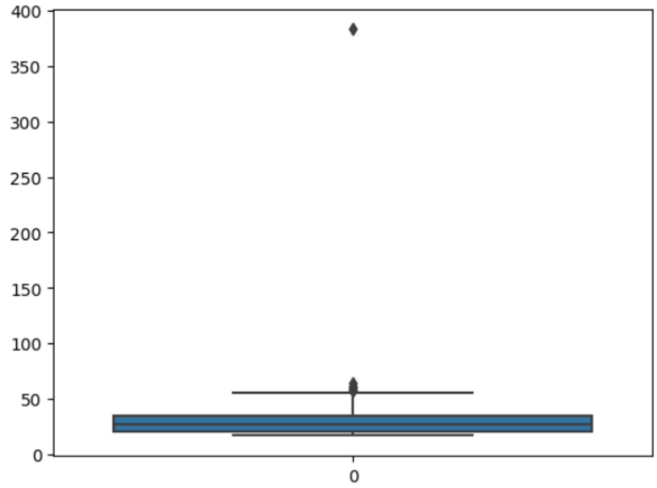
Maximum and Minimum Value Calculation: In order to find the outlier, we used min and max values using which we can get to know that the highest values is our outlier. We found that the maximum age is 383 which is unrealistic as the highest age is around 115-120.So this is an incorrect data. We have replaced the 383 data value with 33.

In [4]:
```
#calculating the maximum and minimum value of age
max_value = data['age'].max()
min_value = data['age'].min()
print("Maximum age found:", max_value)
print("Minimum age found:", min_value)
```

```
Maximum age found: 383.0
Minimum age found: 17.0
```

In [5]:
```
sns.boxplot(data['age']) #plotting age
```

Out[5]: <AxesSubplot:>



we found some usual age for one of the record which is 383, either it might be wrong entry(incorrect value). we can either remove it or change the value to 38

We are also found some null values for the age, so we have filled the null values with the mean.

In [7]:
```
data['age'].isnull().sum() #analysing null values in age column
```

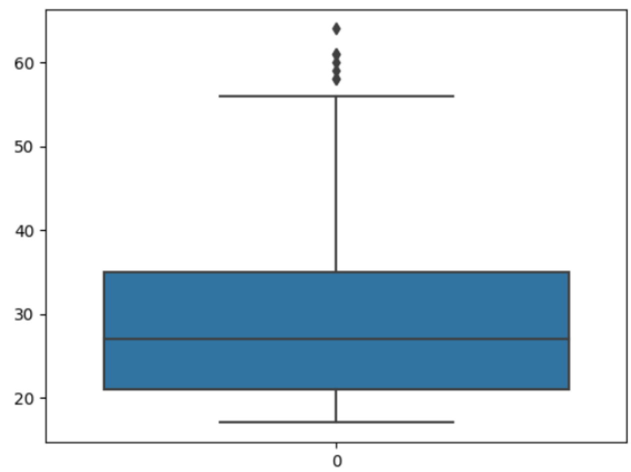
Out[7]: 2

In [8]:
```
data['age'] = data['age'].fillna(round(data['age'].mean())) #filling the null values with mean of the age column data
data['age'].value_counts()
```

Out[8]:
```
21.0    49
20.0    46
23.0    37
```

In [9]:
```
sns.boxplot(data['age']) #plotting age
```

Out[9]: <AxesSubplot:>



In [10]:
```
#calculating the maximum and minimum value of age
max_value = data['age'].max()
min_value = data['age'].min()
print("Maximum age found:", max_value)
print("Minimum age found:", min_value)
```

```
Maximum age found: 64.0
Minimum age found: 17.0
```

Cleaning non-numeric data:

In this we have cleaned non-numeric data(ethnicity (we have replaced the ? with others and merged others with Others), relations (we have replaced ? with others).

Cleaning ethnicity:

```
# calculating ethinicity count
data['ethnicity'].value_counts()
```

```
White-European     233
Asian              123
?                   95
Middle Eastern      92
Black               43
South Asian         36
Others              30
Latino              20
Hispanic            13
Pasifika            12
Turkish              6
others               1
Name: ethnicity, dtype: int64
```

```
# replacing unknown with others
data['ethnicity'] = data['ethnicity'].replace('?', 'others')
#replacing all others with Others as both are same
data['ethnicity'] = data['ethnicity'].replace('others', 'Others')
data['ethnicity'].value_counts()
```

```
White-European     233
Others             126
Asian              123
Middle Eastern      92
Black               43
South Asian         36
Latino              20
Hispanic            13
Pasifika            12
Turkish              6
Name: ethnicity, dtype: int64
```

Cleaning relation :

```
# calculating no.of relations existing
data['relation'].value_counts()
```

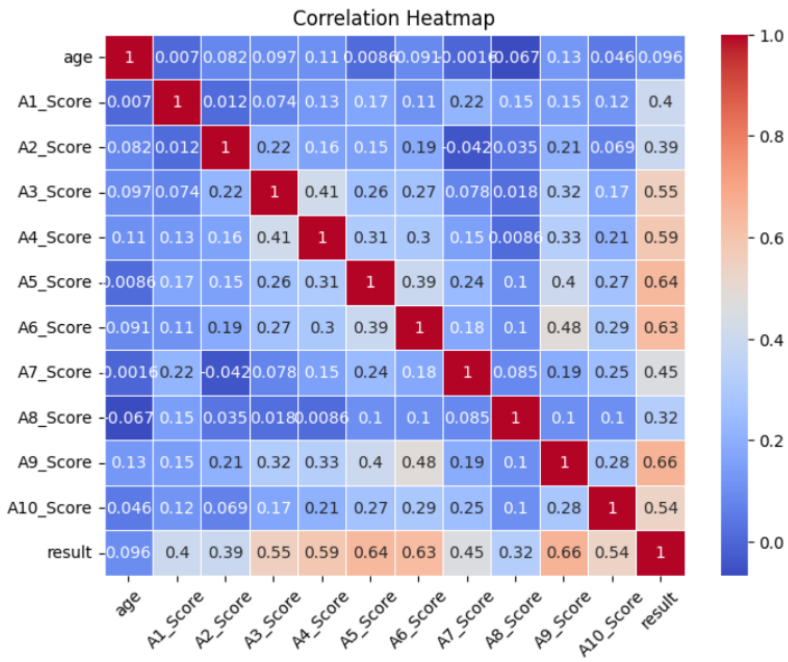
```
Self                        522
?                            95
Parent                       50
Relative                     28
Others                        5
Health care professional      4
Name: relation, dtype: int64
```

```
# replacing unknown relation values with others.
data['relation'] = data['relation'].replace('?', 'Others')
data['relation'].value_counts()
```

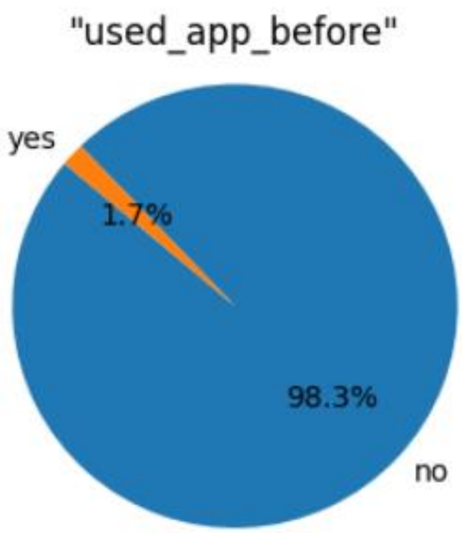
```
Self                        522
Others                      100
Parent                       50
Relative                     28
Health care professional      4
Name: relation, dtype: int64
```

Feature Analysis: This involves techniques such as summary statistics, data visualization (e.g., histograms, scatter plots, box plots), and correlation analysis of features.

We have used heatmap to plot the corelation between the features.

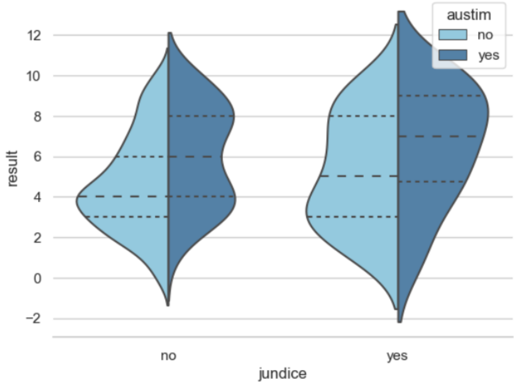


We can see that A1 to A10 scores are closely co-related to the result so we are using A1 to A10 scores and eliminating the result from features.



We can see that the values of the used_app_before feature are highly biased towards no and negligibly yes so we can see that this feature is not highly contributing for our classification.

```
#visualizing results scores corresponding to jundice and autism.
sns.set(style="whitegrid", color_codes=True)
sns.violinplot(x="jundice", y="result", hue="austim", data=c_data, split=True,
               inner="quart", palette={'yes': "steelblue", 'no': "skyblue"})
sns.despine(left=True)
```




```
#visualizing results scores corresponding to gender and Class/ASD
sns.violinplot(x="gender", y="result", hue="Class/ASD", data=c_data, split=True,
               inner="quart", palette={'YES': "steelblue", 'NO': "skyblue"})
sns.despine(left=True)
```


```
#visualizing results scores corresponding to relation and Class/ASD
sns.violinplot(x="relation", y="result", hue="Class/ASD", data=c_data, split=True,
               inner="quart", palette={'YES': "steelblue", 'NO': "skyblue"})
sns.despine(left=True)
```

Feature Engineering:

This involves encoding categorical variables, scaling numerical features, or creating new features.

Feature Engineering

```python
#identifying relevent features based on visualizations and data analysis.
asd_raw = c_data['Class/ASD']
features_raw = c_data[['age', 'gender', 'jundice', 'contry_of_res', 'relation','austim',
                      'A1_Score','A2_Score','A3_Score','A4_Score','A5_Score','A6_Score','A7_Score','A8_Score',
                      'A9_Score','A10_Score']]
```

One-Hot-Coding for non-numeric features

```python
#handling and scaling non numeric features.
def min_max_scaling(series):
    min_val = series.min()
    max_val = series.max()
    scaled_series = (series - min_val) / (max_val - min_val)
    return scaled_series
features_minmax_transform = features_raw.copy()
features_minmax_transform['age'] = min_max_scaling(features_raw['age'])
print(features_minmax_transform.head(5))
```

```
        age gender jundice  contry_of_res relation austim  A1_Score  A2_Score  \
0  0.191489      f      no  United States     Self     no         1         1
1  0.148936      m      no         Brazil     Self    yes         1         1
2  0.212766      m     yes          Spain   Parent    yes         1         1
3  0.382979      f      no  United States     Self    yes         1         1
4  0.489362      f      no          Egypt   Others     no         1         0

   A3_Score  A4_Score  A5_Score  A6_Score  A7_Score  A8_Score  A9_Score  \
0         1         1         0         0         1         1         0
1         0         1         0         0         0         1         0
2         0         1         1         0         1         1         1
3         0         1         0         0         1         1         0
4         0         0         0         0         0         1         0

   A10_Score
0          0
1          1
2          1
3          1
4          0
```

```python
# performing one-hot encoding.
features_final = pd.get_dummies(features_minmax_transform, dtype='int')
display(features_final.head(100))
asd_classes = asd_raw.apply(lambda x: 1 if x == 'YES' else 0)
encoded = list(features_final.columns)
print ("total features after one-hot encoding.".format(len(encoded)))
print (encoded)

### Finding co relation between the features so we can elimonate the features which are highly co related and does not contribute anything to the predictions
c_data_correction= features_final[['age','A1_Score','A2_Score','A3_Score','A4_Score','A5_Score','A6_Score','A7_Score','A8_Score',
                     'A9_Score','A10_Score', 'gender_f', 'gender_m', 'jundice_yes', 'jundice_no']]
correlation_matrix = c_data_correction.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
plt.show()
```

	age	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	...	contry_of_res_United States	contry_of_res_Uruguay	contry_of_res_Viet Nam	relation_Health care professional	relation_Others	relation_Parent	relation_Relative	relation_Self	austim_no	austim_yes
0	0.191489	1	1	1	1	0	0	1	1	0	...	1	0	0	0	0	0	0	1	1	0
1	0.148936	1	1	0	1	0	0	0	1	0	...	0	0	0	0	0	0	0	1	0	1
2	0.212766	1	1	0	1	1	0	1	1	1	...	0	0	0	0	0	1	0	0	0	1
3	0.382979	1	1	0	1	0	0	1	1	0	...	1	0	0	0	0	0	0	1	0	1
4	0.489362	1	0	0	0	0	0	0	1	0	...	0	0	0	0	1	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
95	0.425532	1	1	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	1	0
96	0.234043	1	0	0	0	1	0	1	1	0	...	0	0	0	0	0	0	0	1	1	0
97	0.106383	1	0	0	1	1	0	1	0	0	...	0	0	0	0	0	0	0	1	1	0
98	0.191489	1	1	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	1	1	0
99	0.085106	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	1	1	0

100 rows × 89 columns

```
total features after one-hot encoding.
['age', 'A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score', 'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'gender_f', 'gender_m', 'jundice_no', 'jundice_yes', 'contry_of_res_Afghanistan', 'contry_of_res_AmericanSamoa', 'contry_of_res_Angola', 'contry_of_res_Argentina',
'contry_of_res_Armenia', 'contry_of_res_Aruba', 'contry_of_res_Australia', 'contry_of_res_Austria', 'contry_of_res_Azerbaijan', 'contry_of_res_Bahamas', 'contry_of_res_Bangladesh', 'contry_of_res_Belgium', 'contry_of_res_Bolivia', 'contry_of_res_Brazil', 'contry_of_res_Burundi', 'contry_of_re
s_Canada', 'contry_of_res_Chile', 'contry_of_res_China', 'contry_of_res_Costa Rica', 'contry_of_res_Cyprus', 'contry_of_res_Czech Republic', 'contry_of_res_Ecuador', 'contry_of_res_Egypt', 'contry_of_res_Ethiopia', 'contry_of_res_Finland', 'contry_of_res_France', 'contry_of_res_Germany', 'con
try_of_res_Hong Kong', 'contry_of_res_Iceland', 'contry_of_res_India', 'contry_of_res_Indonesia', 'contry_of_res_Iran', 'contry_of_res_Iraq', 'contry_of_res_Ireland', 'contry_of_res_Italy', 'contry_of_res_Japan', 'contry_of_res_Jordan', 'contry_of_res_Kazakhstan', 'contry_of_res_Lebanon', 'co
ntry_of_res_Malaysia', 'contry_of_res_Mexico', 'contry_of_res_Nepal', 'contry_of_res_Netherlands', 'contry_of_res_New Zealand', 'contry_of_res_Nicaragua', 'contry_of_res_Niger', 'contry_of_res_Oman', 'contry_of_res_Pakistan', 'contry_of_res_Philippines', 'contry_of_res_Portugal', 'contry_of_r
es_Romania', 'contry_of_res_Russia', 'contry_of_res_Saudi Arabia', 'contry_of_res_Serbia', 'contry_of_res_Sierra Leone', 'contry_of_res_South Africa', 'contry_of_res_Spain', 'contry_of_res_Sri Lanka', 'contry_of_res_Sweden', 'contry_of_res_Tonga', 'contry_of_res_Turkey', 'contry_of_res_Ukrain
e', 'contry_of_res_United Arab Emirates', 'contry_of_res_United Kingdom', 'contry_of_res_United States', 'contry_of_res_Uruguay', 'contry_of_res_Viet Nam', 'relation_Health care professional', 'relation_Others', 'relation_Parent', 'relation_Relative', 'relation_Self', 'austim_no', 'austim_ye
s']
```

We have only selected the important features based on the above analysis on the features.

Spliting the Data

```python
#splitting data into train and test sets.
def data_spliting( X , y) :
    if isinstance(X, pd.Series):
        X = X.to_frame()  # Convert Series to DataFrame
    if isinstance(y, pd.DataFrame) and y.shape[1] == 1:
        y = y.iloc[:, 0]  # Convert DataFrame to Series

    X = (X - X.mean()) / X.std()
    split_index = int(0.50 * len(X))
    X_train, X_test = X.iloc[:split_index], X.iloc[split_index:]
    y_train, y_test = y.iloc[:split_index], y.iloc[split_index:]
    return X_train, X_test, y_train, y_test
```

Models:

Logistic Regression, Decision Trees, and Random Forests each offer unique advantages for ASD detection.

Logistic regression is a statistical model that estimates the probability of a binary outcome based on one or more predictor variables. It is particularly useful for cases where the outcome is dichotomous. It is well-suited for binary classification problems, like determining whether an individual has ASD or not.

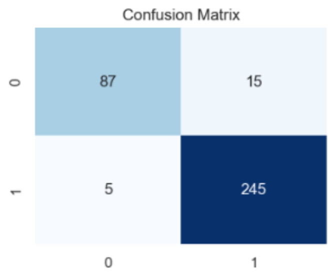
Decision trees are a non-parametric supervised learning method used for classification and regression. Decision trees can handle complex, non-linear relationships between features.

Random Forest is an ensemble learning method for classification, regression, and other tasks that operate by constructing a multitude of decision trees at training time. Random Forest reduces the risk of overfitting by averaging multiple trees, which typically leads to improved accuracy.

- ### Logistic Regression
  Without cross validation

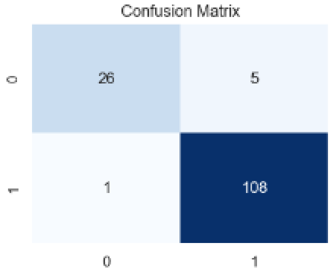
```
Logistic Regression without cross validation:
Accuracy: 0.9431818181818182
F1 Score: 0.8969072164948452
Confusion Matrix:
 [[ 87  15]
 [  5 245]]
```

```
#confusion matrix for logistic regression visualization
plt.figure(figsize=(4, 3))
plt.title('Confusion Matrix')
sns.heatmap(confusion_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.show()
```

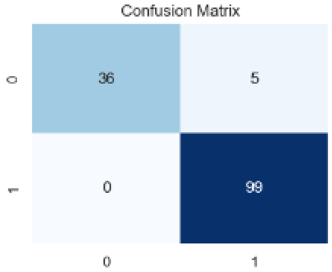


With cross validation(k-fold)

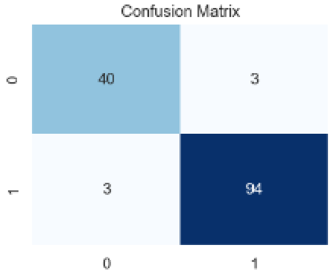
```
Result of Logistic Regression with 5-Fold Cross-Validation: {'log_cv_mean_accuracy': 0.9571428571428572, 'f1_score': 0.896551724137931, 'confusion_matrix': array([[ 26,   5],
       [  1, 108]], dtype=int64)}
```



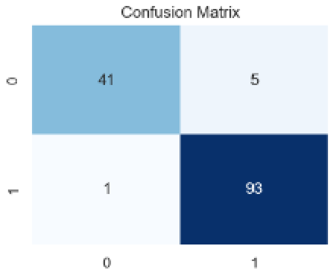

```
Result of Logistic Regression with 5-Fold Cross-Validation: {'log_cv_mean_accuracy': 0.9642857142857143, 'f1_score': 0.9350649350649352, 'confusion_matrix': array([[36,  5],
       [ 0, 99]], dtype=int64)}
```




```
Result of Logistic Regression with 5-Fold Cross-Validation: {'log_cv_mean_accuracy': 0.9571428571428572, 'f1_score': 0.9302325581395349, 'confusion_matrix': array([[40,  3],
       [ 3, 94]], dtype=int64)}
```




```
Result of Logistic Regression with 5-Fold Cross-Validation: {'log_cv_mean_accuracy': 0.9571428571428572, 'f1_score': 0.9318181818181818, 'confusion_matrix': array([[41,  5],
       [ 1, 93]], dtype=int64)}
```

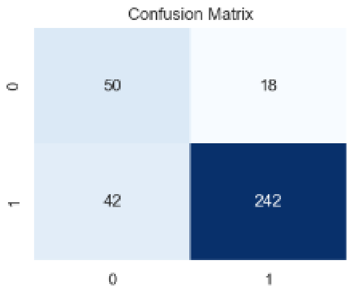



```
Mean Accuracy of Logistic Regression with K fold cross validation: 0.96
```

- # Decision tree

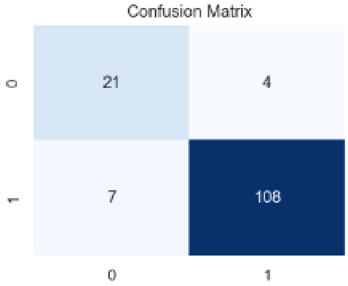
Without cross validation

```
Result of Decision tree without cross validation:
Accuracy: 0.8295454545454546
F1 Score: 0.625
```

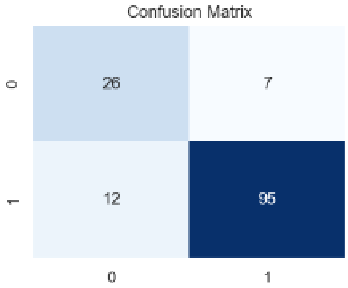


With cross validation(k-fold)

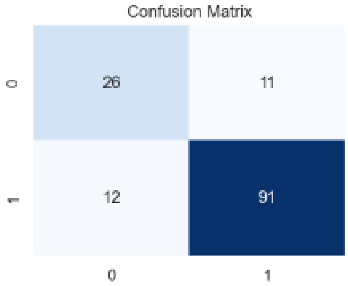
```
confusion_matrix: [[ 21   4]
 [  7 108]]
```



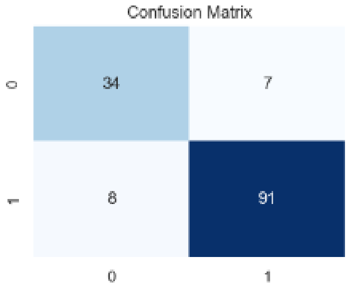

```
confusion_matrix: [[26  7]
 [12 95]]
```




```
confusion_matrix: [[26 11]
 [12 91]]
```



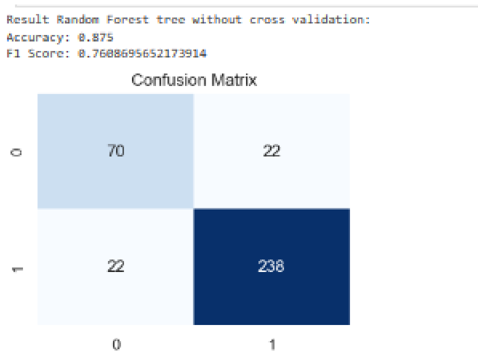

```
confusion_matrix: [[34  7]
 [ 8 91]]
```



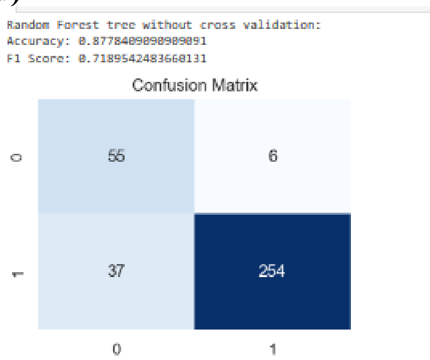

```
Result of Decision tree with cross validation:
F1 Score: 0.8192771084337348
Cross-Validation Accuracy Scores: [0.8214285714285714, 0.9214285714285714, 0.8642857142857143, 0.8357142857142857, 0.8928571428571429]
Mean Accuracy of Decision tress with k-fold cross validation: 0.8671428571428572
```

- ## Random Forest

  Without cross validation

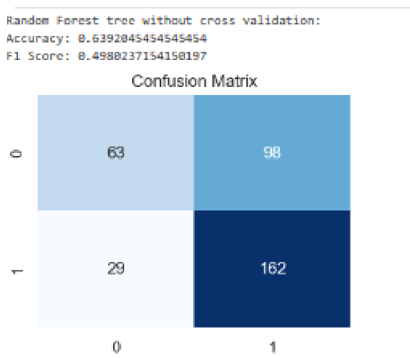


  With cross validation(k-fold)

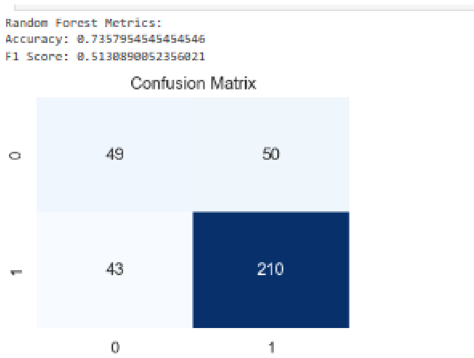


Hyperparameter Tuning:

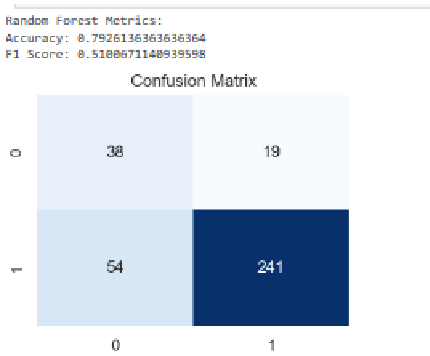
Fine-tune the hyperparameters of the model(s) to optimize their performance further.
Parameter tuning for random forest:
For,
rf = RandomForest(n_trees=8, max_depth=5, sample_size=90, n_features=int(np.sqrt(features_final.shape[1])), random_state=1)



For,
rf = RandomForest(n_trees=10, max_depth=3, sample_size=100, n_features=int(np.sqrt(features_final.shape[1])), random_state=1



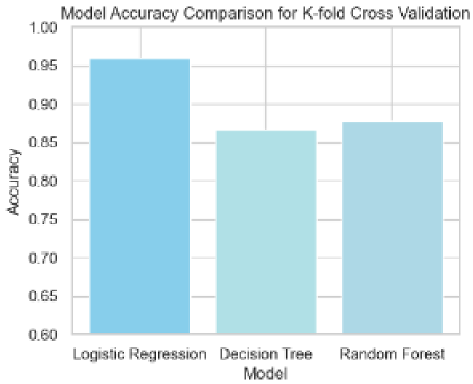
For,
rf = RandomForest(n_trees=20, max_depth=5, sample_size=100, n_features=int(np.sqrt(features_final.shape[1])), random_state=1)



MODEL ACCURACY:

Without cross validation

With cross validation



From the above report, we observed that logistic regression has high accuracy, compared with decision trees and random forest. Logistic regression accuracy is 0.94, followed by random forest with an accuracy of 0.87, and decision tree with an accuracy of 0.82.

After performing k-fold (cross-validation) the highest accuracy value is 0.96 which is for the logistic regression, and decision tree with an accuracy 0.86, and random forest with an accuracy of 0.87.

We have seen that there is no drastic difference in accuracies for with and without cross validation technique this could be due to the size of our dataset which is relatively small. Models trained on small dataset are less likely to capture the irrelevant patterns in the data, leading to better generalization performance and higher accuracy on test data.

6. Conclusion:

We analysed the effectiveness of different machine learning models for the task of autism detection. Our analysis focused on logistic regression, decision tree and random forest models, evaluating their performance using accuracy as the primary evaluation metric.

Our outcomes indicate that logistic regression outperforms the decision tree and random forest models, achieving a higher accuracy score in detecting autism. This suggest that logistic regression maybe a suitable and effective approach for autism detection, specifically when accuracy is prioritized as the evaluation metric.

7. Future Scope:

The future scope by leveraging advancements in ML algorithms, such as deep learning and ensemble techniques, the project can refine its predictive models for autism spectrum disorder (ASD) diagnosis and prognosis, enhancing accuracy and reliability.

Integration of multimodal data sources, including genetic, imaging, behavioural, and environmental factors, can enable more comprehensive and holistic assessments, facilitating early detection and individualized treatment planning.

8. References:

1. https://ieeexplore.ieee.org/document/8545113
2. https://www.kaggle.com/datasets/afarinbargrizan/asd-final