1.Importing the dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split, cross_val_score,RandomizedSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import pickle
```

2. Data Loading & Understading

```
df=pd.read_csv("/content/train.csv")
```

Initial Inspection

```
df.shape
```

```
(800, 22)
```

```
df.head()
```

	ID	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity
0	1	1	0	1	0	1	0	1	0	1	1	38.172746	f	?
1	2	0	0	0	0	0	0	0	0	0	0	47.750517	m	?
2	3	1	1	1	1	1	1	1	1	1	1	7.380373	m	White-European
3	4	0	0	0	0	0	0	0	0	0	0	23.561927	f	?
4	5	0	0	0	0	0	0	0	0	0	0	43.205790	m	?

```
df.tail(2)
```

	ID	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnici
798	799	0	0	0	0	0	0	0	0	0	0	16.414305	f	
799	800	0	1	0	0	0	0	0	0	0	0	46.966113	f	

```
# to display all the columns of a dataframe
pd.set_option('display.max_columns',None)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               800 non-null    int64
 1   A1_Score         800 non-null    int64
```

```
 2   A2_Score         800 non-null    int64
 3   A3_Score         800 non-null    int64
 4   A4_Score         800 non-null    int64
 5   A5_Score         800 non-null    int64
 6   A6_Score         800 non-null    int64
 7   A7_Score         800 non-null    int64
 8   A8_Score         800 non-null    int64
 9   A9_Score         800 non-null    int64
 10  A10_Score        800 non-null    int64
 11  age              800 non-null    float64
 12  gender           800 non-null    object
 13  ethnicity        800 non-null    object
 14  jaundice         800 non-null    object
 15  austim           800 non-null    object
 16  contry_of_res    800 non-null    object
 17  used_app_before  800 non-null    object
 18  result           800 non-null    float64
 19  age_desc         800 non-null    object
 20  relation         800 non-null    object
 21  Class/ASD        800 non-null    int64
dtypes: float64(2), int64(12), object(8)
memory usage: 137.6+ KB
```

```
#rename the column name from contry_of_res to country_of_res and austim to autism
df.rename(columns={"contry_of_res":"country_of_res"},inplace=True)
df.rename(columns={"austim":"autism"},inplace=True)
```

```
#convert age column datatype to integer
df["age"]=df["age"].astype(int)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 800 entries, 0 to 799
Data columns (total 22 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               800 non-null    int64
 1   A1_Score         800 non-null    int64
 2   A2_Score         800 non-null    int64
 3   A3_Score         800 non-null    int64
 4   A4_Score         800 non-null    int64
 5   A5_Score         800 non-null    int64
 6   A6_Score         800 non-null    int64
 7   A7_Score         800 non-null    int64
 8   A8_Score         800 non-null    int64
 9   A9_Score         800 non-null    int64
 10  A10_Score        800 non-null    int64
 11  age              800 non-null    int64
 12  gender           800 non-null    object
 13  ethnicity        800 non-null    object
 14  jaundice         800 non-null    object
 15  autism           800 non-null    object
 16  country_of_res   800 non-null    object
 17  used_app_before  800 non-null    object
 18  result           800 non-null    float64
 19  age_desc         800 non-null    object
 20  relation         800 non-null    object
 21  Class/ASD        800 non-null    int64
dtypes: float64(1), int64(13), object(8)
memory usage: 137.6+ KB
```

```
df.head(2)
```

	ID	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jaundi
0	1	1	0	1	0	1	0	1	0	1	1	38	f	?	
1	2	0	0	0	0	0	0	0	0	0	0	47	m	?	

```
for col in df.columns :
  numerical_features =["ID","age","result"]
  if col not in numerical_features:
    print(col,df[col].unique())
    print("-"*60)
```

```
A1_Score [1 0]
------------------------------------------------------------
A2_Score [0 1]
------------------------------------------------------------
A3_Score [1 0]
------------------------------------------------------------
A4_Score [0 1]
------------------------------------------------------------
A5_Score [1 0]
------------------------------------------------------------
A6_Score [0 1]
------------------------------------------------------------
A7_Score [1 0]
------------------------------------------------------------
A8_Score [0 1]
------------------------------------------------------------
A9_Score [1 0]
------------------------------------------------------------
A10_Score [1 0]
------------------------------------------------------------
gender ['f' 'm']
------------------------------------------------------------
ethnicity ['?' 'White-European' 'Middle Eastern ' 'Pasifika' 'Black' 'Others'
 'Hispanic' 'Asian' 'Turkish' 'South Asian' 'Latino' 'others']
------------------------------------------------------------
jaundice ['no' 'yes']
------------------------------------------------------------
autism ['no' 'yes']
------------------------------------------------------------
country_of_res ['Austria' 'India' 'United States' 'South Africa' 'Jordan'
 'United Kingdom' 'Brazil' 'New Zealand' 'Canada' 'Kazakhstan'
 'United Arab Emirates' 'Australia' 'Ukraine' 'Iraq' 'France' 'Malaysia'
 'Viet Nam' 'Egypt' 'Netherlands' 'Afghanistan' 'Oman' 'Italy'
 'AmericanSamoa' 'Bahamas' 'Saudi Arabia' 'Ireland' 'Aruba' 'Sri Lanka'
 'Russia' 'Bolivia' 'Azerbaijan' 'Armenia' 'Serbia' 'Ethiopia' 'Sweden'
 'Iceland' 'Hong Kong' 'Angola' 'China' 'Germany' 'Spain' 'Tonga'
 'Pakistan' 'Iran' 'Argentina' 'Japan' 'Mexico' 'Nicaragua' 'Sierra Leone'
 'Czech Republic' 'Niger' 'Romania' 'Cyprus' 'Belgium' 'Burundi'
 'Bangladesh']
------------------------------------------------------------
used_app_before ['no' 'yes']
------------------------------------------------------------
age_desc ['18 and more']
------------------------------------------------------------
relation ['Self' 'Relative' 'Parent' '?' 'Others' 'Health care professional']
------------------------------------------------------------
Class/ASD [0 1]
------------------------------------------------------------
```

```
#dropping the ID and age_desc column
df = df.drop(columns=["ID","age_desc"])
```

```
df.shape
```

```
(800, 20)
```

```
df.head(2)
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jaundice
0	1	0	1	0	1	0	1	0	1	1	38	f	?	no
1	0	0	0	0	0	0	0	0	0	0	47	m	?	no

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.columns
```

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jaundice', 'autism', 'country_of_res', 'used_app_before',
       'result', 'relation', 'Class/ASD'],
      dtype='object')
```

```
df["country_of_res"].unique()
```

```
array(['Austria', 'India', 'United States', 'South Africa', 'Jordan',
       'United Kingdom', 'Brazil', 'New Zealand', 'Canada', 'Kazakhstan',
       'United Arab Emirates', 'Australia', 'Ukraine', 'Iraq', 'France',
       'Malaysia', 'Viet Nam', 'Egypt', 'Netherlands', 'Afghanistan',
       'Oman', 'Italy', 'AmericanSamoa', 'Bahamas', 'Saudi Arabia',
       'Ireland', 'Aruba', 'Sri Lanka', 'Russia', 'Bolivia', 'Azerbaijan',
       'Armenia', 'Serbia', 'Ethiopia', 'Sweden', 'Iceland', 'Hong Kong',
       'Angola', 'China', 'Germany', 'Spain', 'Tonga', 'Pakistan', 'Iran',
       'Argentina', 'Japan', 'Mexico', 'Nicaragua', 'Sierra Leone',
       'Czech Republic', 'Niger', 'Romania', 'Cyprus', 'Belgium',
       'Burundi', 'Bangladesh'], dtype=object)
```

```
from collections.abc import Mapping
#define the mapping dictionary for country names
mapping ={
    "Viet Nam":"vietnam",
    "AmericanSamoa":"United States",
    "Hong Kong":"China"
}

#replace value in the country column

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


df["country_of_res"].unique()
```

```
array(['Austria', 'India', 'United States', 'South Africa', 'Jordan',
       'United Kingdom', 'Brazil', 'New Zealand', 'Canada', 'Kazakhstan',
       'United Arab Emirates', 'Australia', 'Ukraine', 'Iraq', 'France',
       'Malaysia', 'vietnam', 'Egypt', 'Netherlands', 'Afghanistan',
       'Oman', 'Italy', 'Bahamas', 'Saudi Arabia', 'Ireland', 'Aruba',
       'Sri Lanka', 'Russia', 'Bolivia', 'Azerbaijan', 'Armenia',
       'Serbia', 'Ethiopia', 'Sweden', 'Iceland', 'China', 'Angola',
       'Germany', 'Spain', 'Tonga', 'Pakistan', 'Iran', 'Argentina',
       'Japan', 'Mexico', 'Nicaragua', 'Sierra Leone', 'Czech Republic',
       'Niger', 'Romania', 'Cyprus', 'Belgium', 'Burundi', 'Bangladesh'],
      dtype=object)
```

```
#target class distribution

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

Class/ASD	count
0	639
1	161

Insight get from step 2

1.missing values in ethnicity & relation

2.age_desc column has only 1 unique value.so it removed as it is not important for prediction

3.fixed country name

4.identified class imbalance in the target column

3.Exploratory Data Analysis(EDA)

```
df.describe()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	
count	800.000000	800.000000	800.000000	800.00000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800.000000	800
mean	0.560000	0.530000	0.450000	0.41500	0.395000	0.303750	0.397500	0.508750	0.495000	0.617500	27.963750	8
std	0.496697	0.499411	0.497805	0.49303	0.489157	0.460164	0.489687	0.500236	0.500288	0.486302	16.329827	4
min	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.000000	-6
25%	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	17.000000	5
50%	1.000000	1.000000	0.000000	0.00000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	24.000000	9
75%	1.000000	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	35.250000	12
max	1.000000	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	89.000000	15

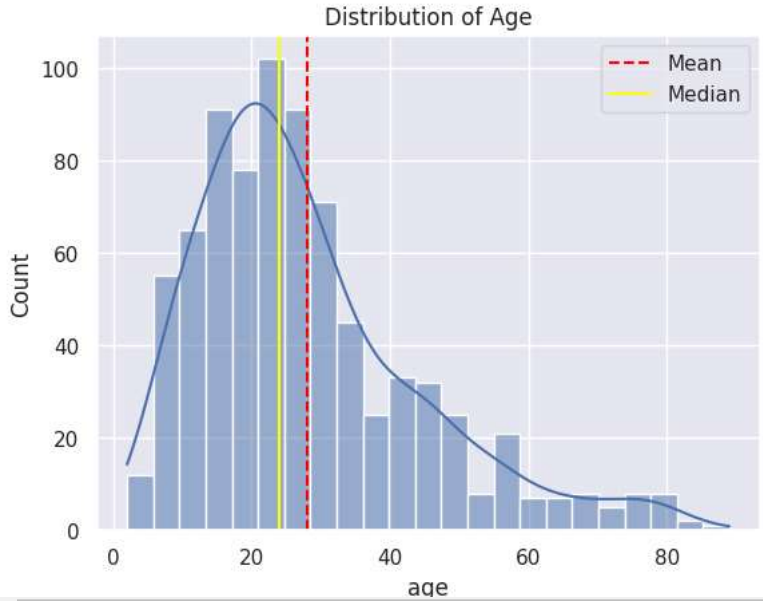
Univariate Analysis

Numerical Columns:

- age
- result

```
#set the desired theme using seaborn
sns.set_theme(style="darkgrid")


#Histogram for age
sns.histplot(df["age"],kde=True)
plt.title("Distribution of Age")

#calculate mean and median
age_mean=df["age"].mean()
age_median=df["age"].median()

print("Mean :",age_mean)
print("Median :",age_median)

#add vertical lines for mean and median
plt.axvline(age_mean,color="red",linestyle="--",label="Mean")
plt.axvline(age_median,color="yellow",linestyle="-",label="Median")
plt.legend()
plt.show()
```

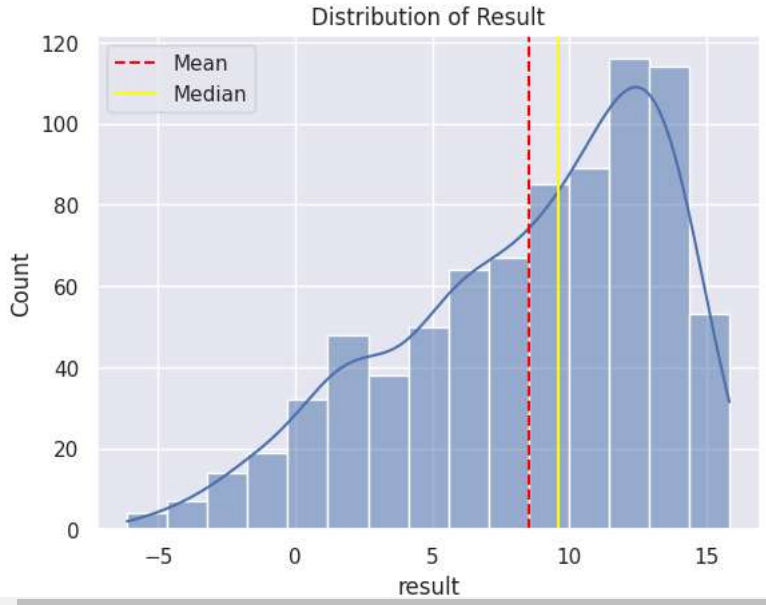
```
Mean : 27.96375
Median : 24.0
```




```
#Histogram for result
sns.histplot(df["result"],kde=True)
plt.title("Distribution of Result")
```

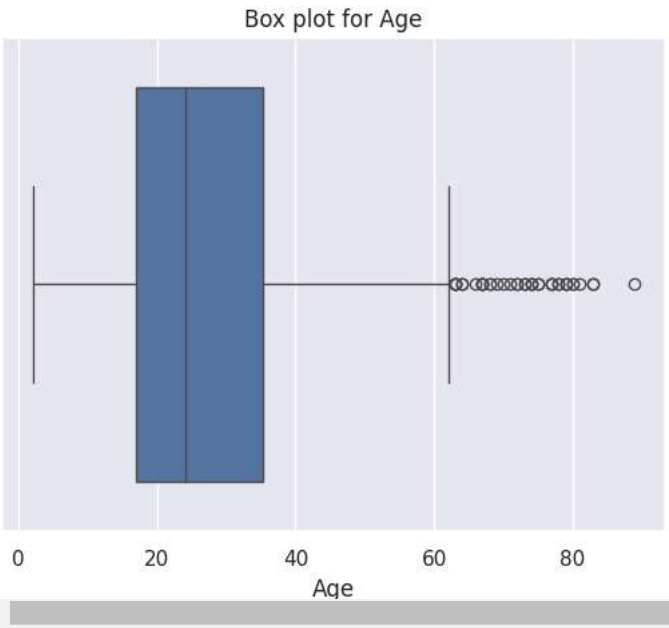
```
#calculate mean and median
result_mean=df["result"].mean()
result_median=df["result"].median()

print("Mean :",result_mean)
print("Median :",result_median)

#add vertical lines for mean and median
plt.axvline(result_mean,color="red",linestyle="--",label="Mean")
plt.axvline(result_median,color="yellow",linestyle="-",label="Median")
plt.legend()
plt.show()
```

```
Mean : 8.537303106501248
Median : 9.605299308
```

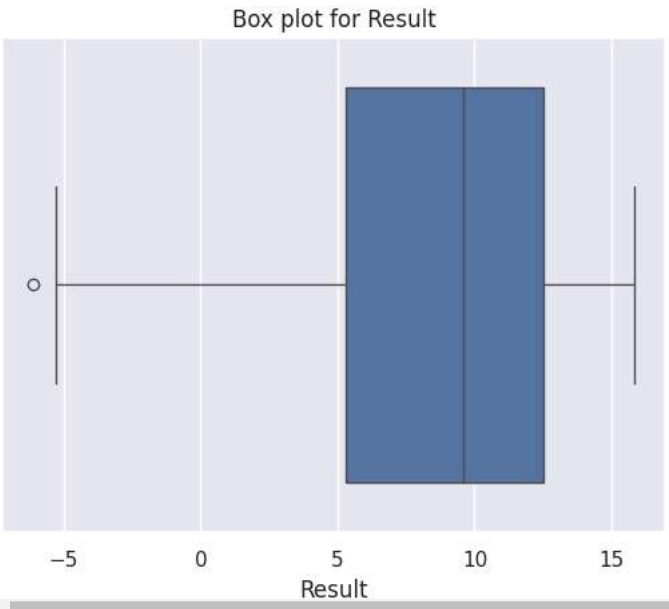


Box plots for identifying outliers in the numerical columns

```
#Box plot
sns.boxplot(x=df["age"])
plt.title("Box plot for Age")
plt.xlabel("Age")

plt.show()
```




```
#Box plot
sns.boxplot(x=df["result"])
plt.title("Box plot for Result")
plt.xlabel("Result")
plt.show()
```




```
#count the outliers using the IQR method
Q1=df["age"].quantile(0.25)
Q3=df["age"].quantile(0.75)
IQR=Q3-Q1
lower_bound=Q1-(1.5*IQR)
upper_bound=Q3+(1.5*IQR)

age_outlier=df[(df["age"]<lower_bound) | (df["age"]>upper_bound)]
len(age_outlier)
```

```
39
```

```
#count the outliers using the IQR method
Q1=df["result"].quantile(0.25)
Q3=df["result"].quantile(0.75)
IQR=Q3-Q1
lower_bound=Q1-(1.5*IQR)
upper_bound=Q3+(1.5*IQR)
```

```
result_outlier=df[(df["result"]<lower_bound) | (df["result"]>upper_bound)]
len(result_outlier)
```

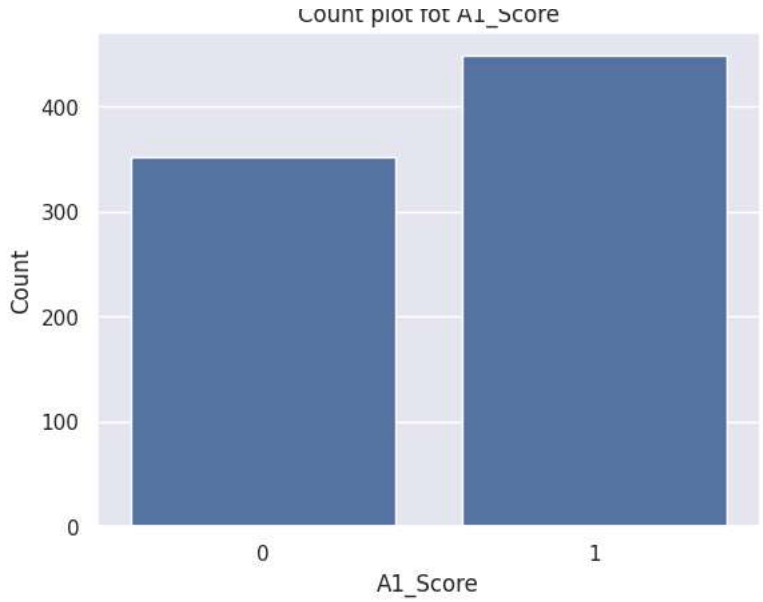
```
1
```

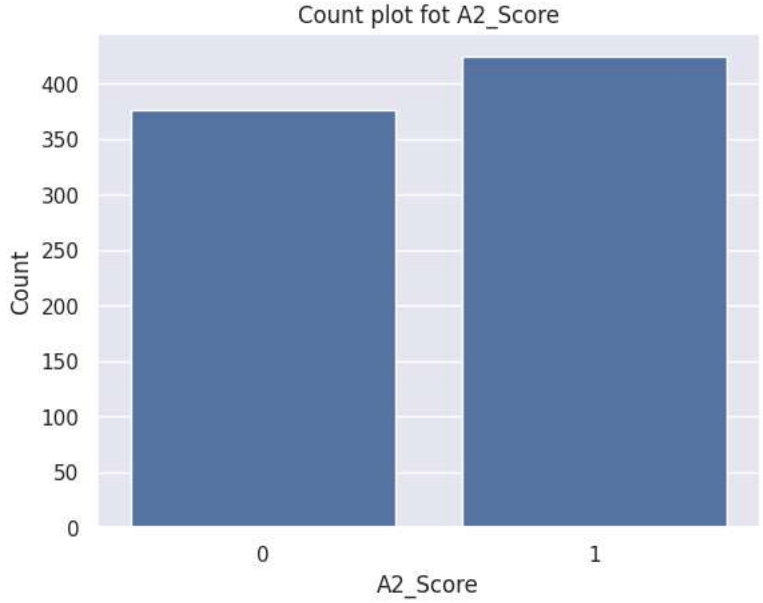
Univariate Analysis of Categorical columns

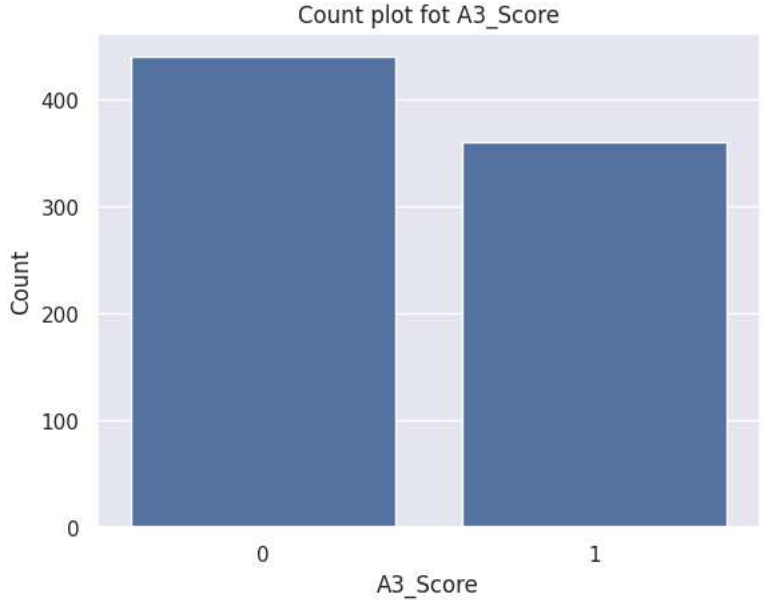
```
df.columns
```

```
Index(['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'age', 'gender',
       'ethnicity', 'jaundice', 'autism', 'country_of_res', 'used_app_before',
       'result', 'relation', 'Class/ASD'],
      dtype='object')
```

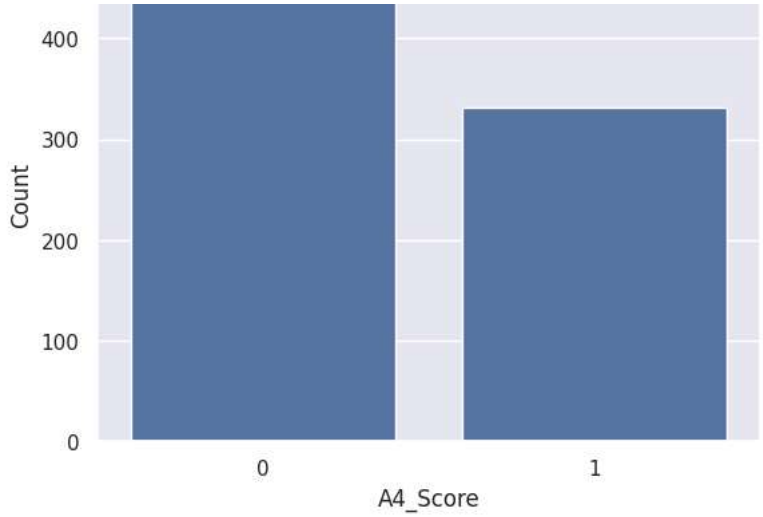
```
categorical_columns=['A1_Score', 'A2_Score', 'A3_Score', 'A4_Score', 'A5_Score', 'A6_Score',
       'A7_Score', 'A8_Score', 'A9_Score', 'A10_Score', 'gender',
       'ethnicity', 'jaundice', 'autism', 'country_of_res', 'used_app_before',
        'relation']

for col in categorical_columns:
  # plt.figure(figsize=(20,5))
  sns.countplot(x=df[col])
  plt.title(f"Count plot fot {col}")
  plt.xlabel(col)
  plt.ylabel("Count")
  plt.show()
```


Count plot fot A1_Score
Count
A1_Score
0
1
0
100
200
300
400

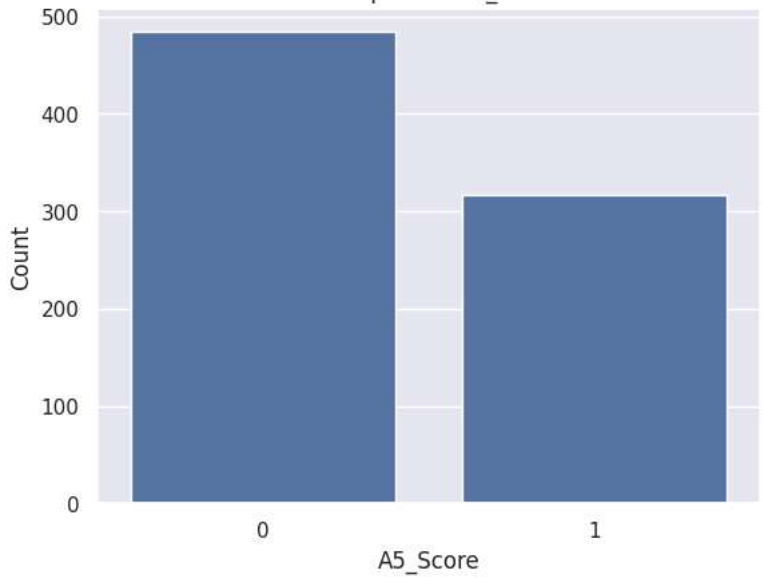

Count plot fot A2_Score
Count
A2_Score
0
1
0
50
100
150
200
250
300
350
400


Count plot fot A3_Score
Count
A3_Score
0
1
0
100
200
300
400

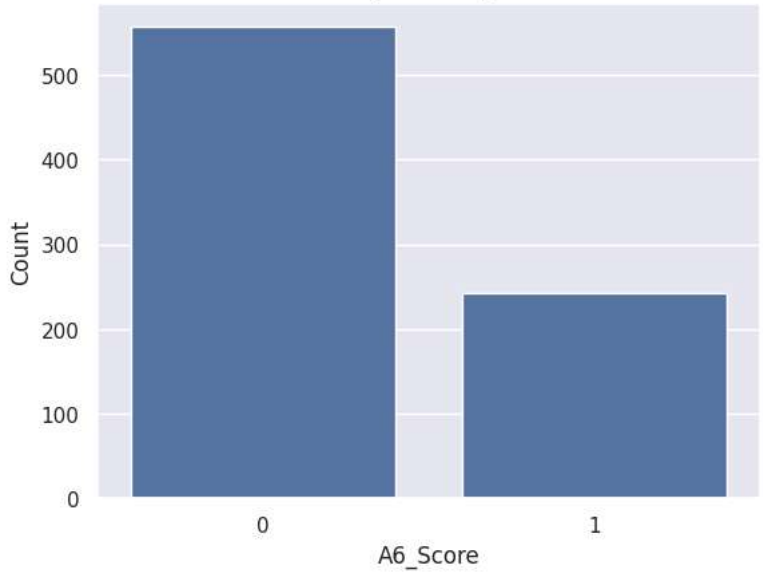

Count plot fot A4_Score



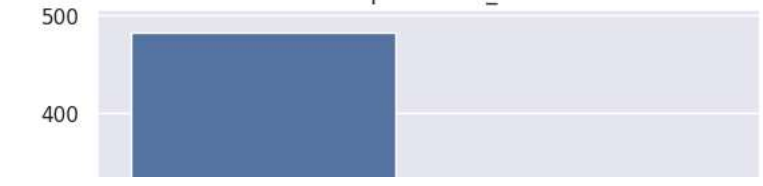
Count plot fot A5_Score

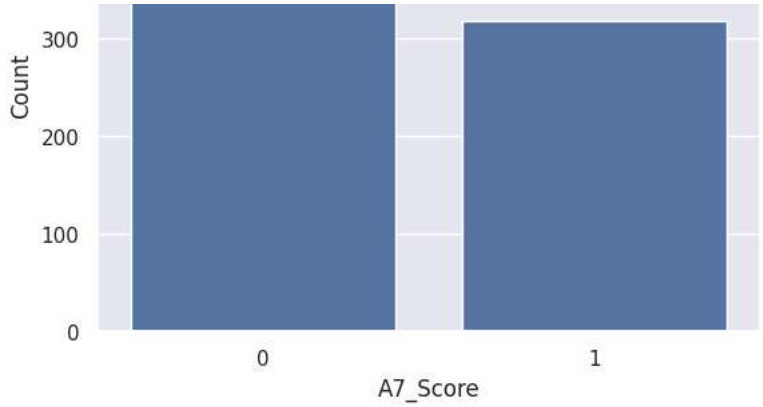


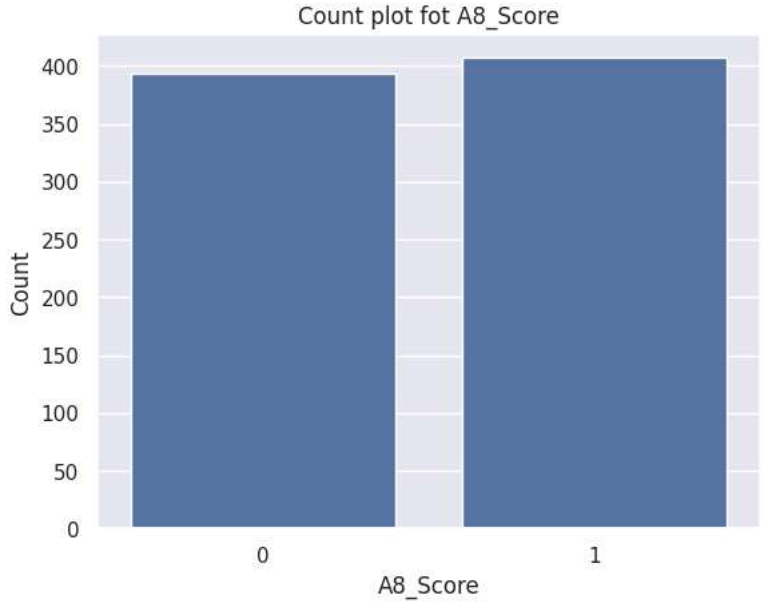
Count plot fot A6_Score

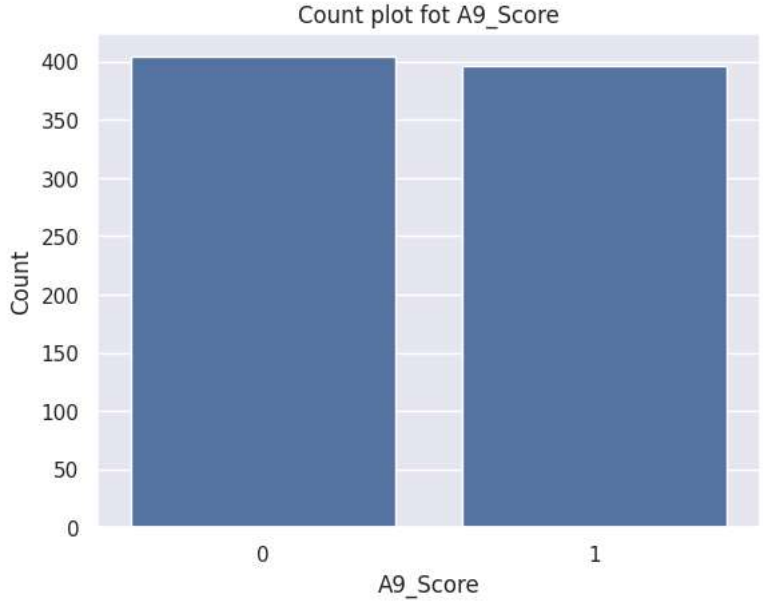


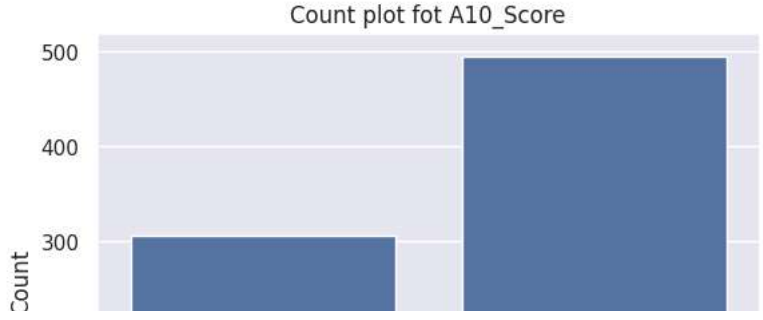
Count plot fot A7_Score

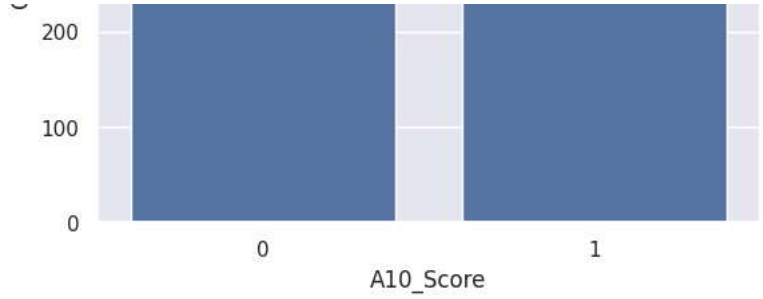


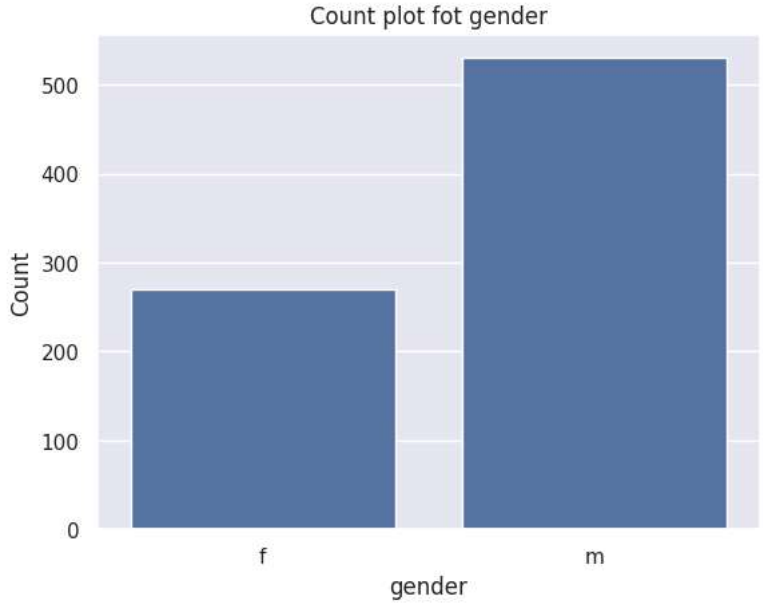

Count
300
200
100
0
0
1
A7_Score

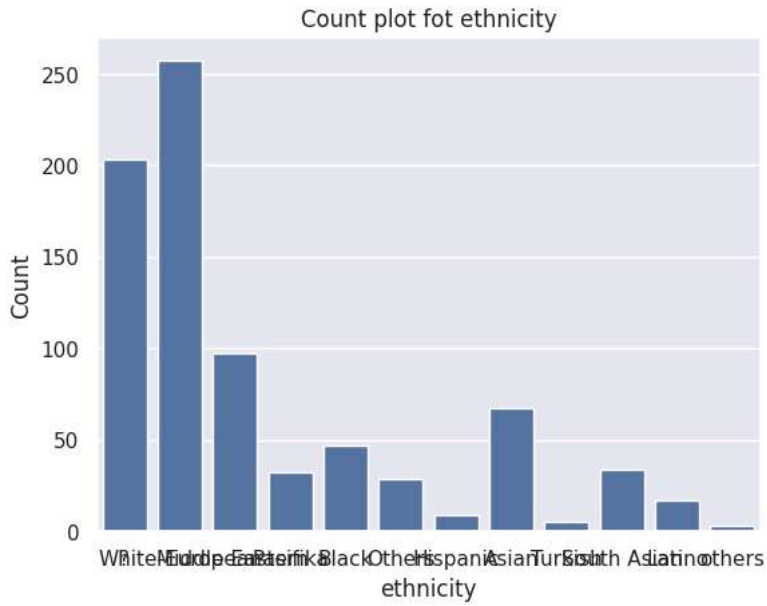

Count plot fot A8_Score
Count
400
350
300
250
200
150
100
50
0
0
1
A8_Score

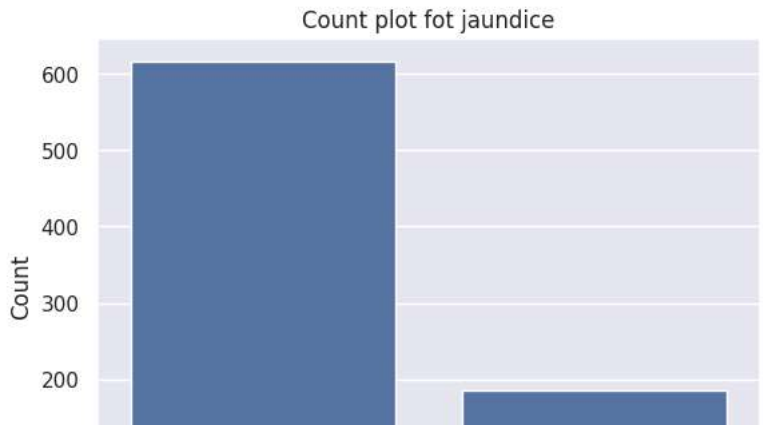

Count plot fot A9_Score
Count
400
350
300
250
200
150
100
50
0
0
1
A9_Score

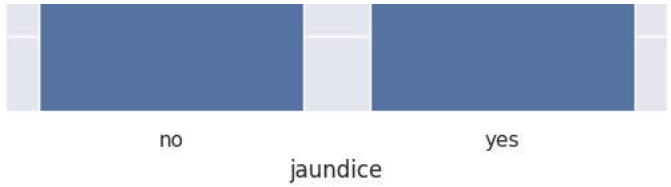

Count plot fot A10_Score
500
400
300
Count

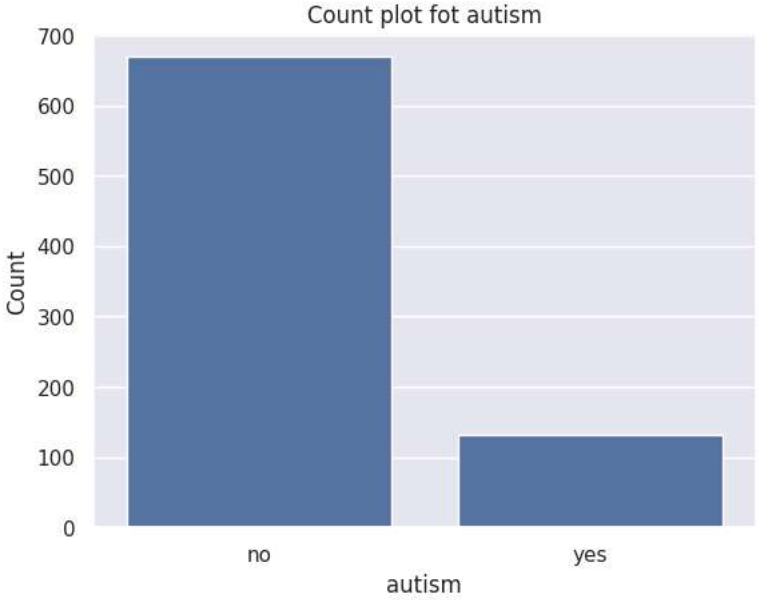

200
100
0
0
1
A10_Score

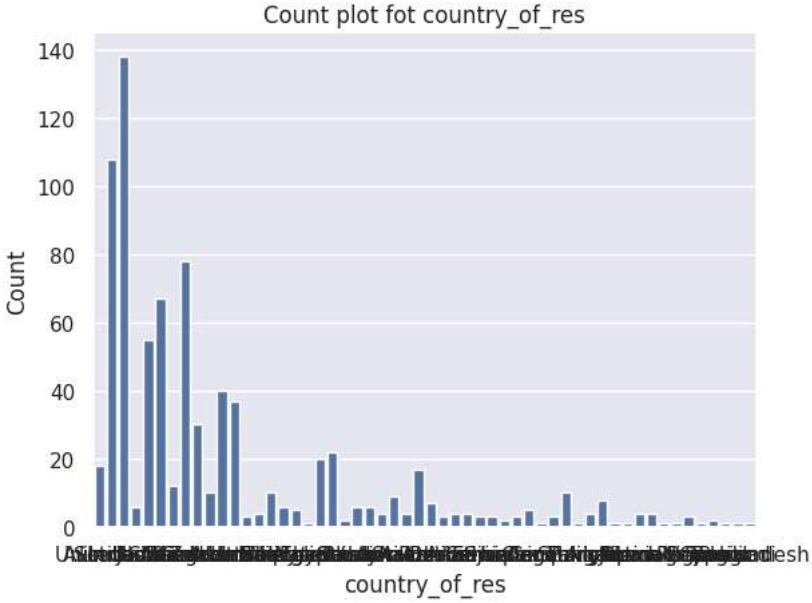

Count plot fot gender
500
400
300
200
100
0
Count
f
m
gender

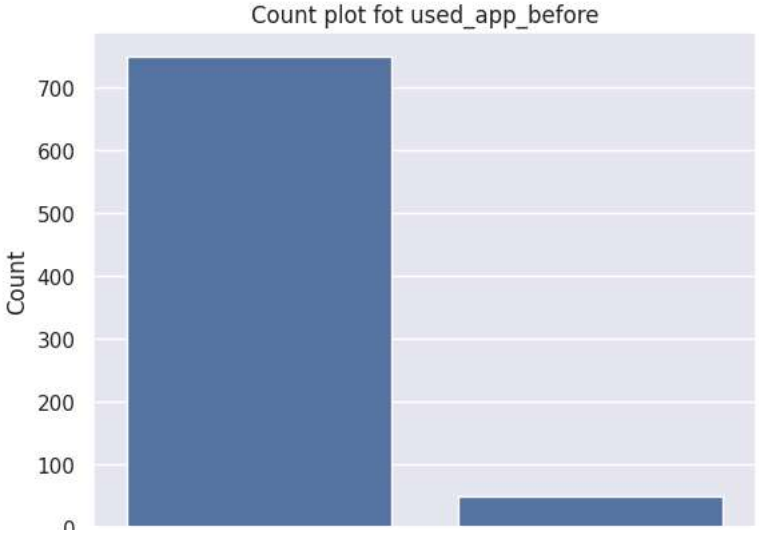

Count plot fot ethnicity
250
200
150
100
50
0
Count
White
Black
Others
Hispanic
Asian
Turkish
South Asian
Latino
others
ethnicity

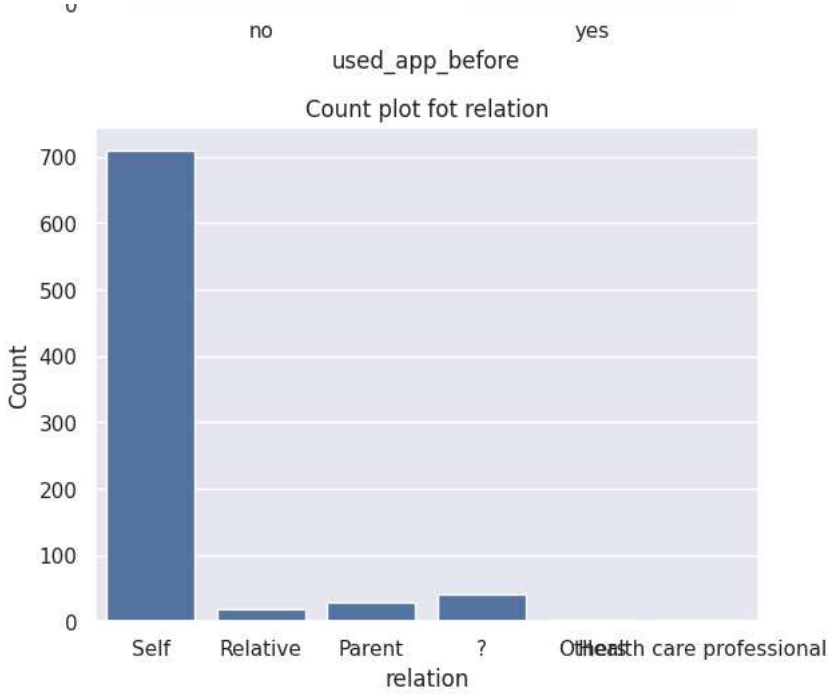

Count plot fot jaundice
600
500
400
300
200
Count


100
0
no
yes
jaundice

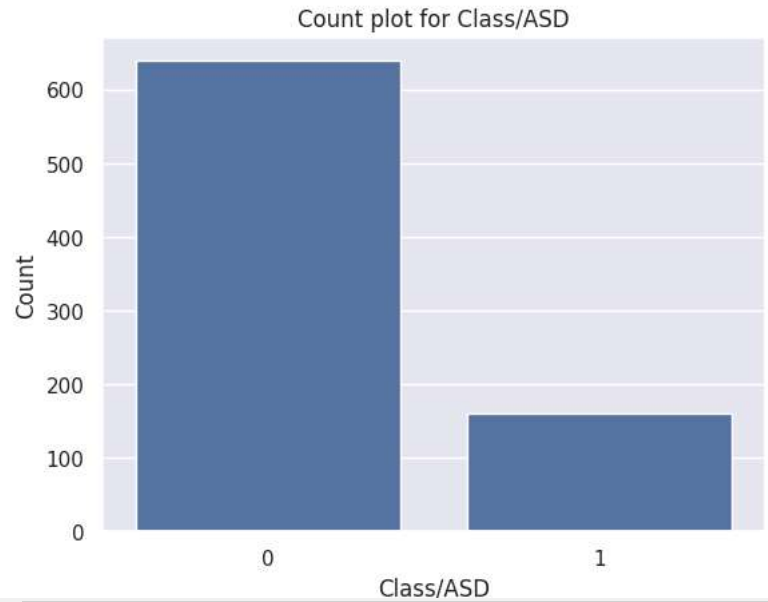

Count plot fot autism
700
600
500
400
300
200
100
0
Count
no
yes
autism


Count plot fot country_of_res
140
120
100
80
60
40
20
0
Count
country_of_res


Count plot fot used_app_before
700
600
500
400
300
200
100
0
Count


0
no
yes
used_app_before
Count plot fot relation
700
600
500
400
300
200
100
0
Count
Self
Relative
Parent
?
Health care professional
relation

```
#count plot for target column(Class/ASD)
sns.countplot(x=df["Class/ASD"])
plt.title("Count plot for Class/ASD")
plt.xlabel("Class/ASD")
plt.ylabel("Count")
plt.show()
```




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

Class/ASD	count
0	639
1	161

handle missing values in ethnicity and relation column

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

```
df["ethnicity"].unique()
```

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

```
df["relation"].unique()
```

```
array(['Self', 'Relative', 'Parent', '?', 'Others',
       'Health care professional'], dtype=object)
```

```
df["relation"]=df["relation"].replace({"?":"Others",
                                       "Relative":"Others",
                                       "Parent":"Others",
                                       "Health care professional":"Others"})
```

```
df["relation"].unique()
```

```
array(['Self', 'Others'], dtype=object)
```

```
df.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jaundice
0	1	0	1	0	1	0	1	0	1	1	38	f	Others	no
1	0	0	0	0	0	0	0	0	0	0	47	m	Others	no
2	1	1	1	1	1	1	1	1	1	1	7	m	White-European	no
3	0	0	0	0	0	0	0	0	0	0	23	f	Others	no
4	0	0	0	0	0	0	0	0	0	0	43	m	Others	no

Next steps: Generate code with df View recommended plots New interactive sheet

Label Encoding

```
#identify columns with object data type
object_columns=df.select_dtypes(include="object").columns
```

```
print(object_columns)
```

```
Index(['gender', 'ethnicity', 'jaundice', 'autism', 'country_of_res',
       'used_app_before', 'relation'],
      dtype='object')
```

```
#initialize a dictionary to store the encoders
encoders={}
#apply label encoding and store the encoders
for column in object_columns:
  label_encoder=LabelEncoder()
  df[column]=label_encoder.fit_transform(df[column])
  encoders[column]=label_encoder #saving the encoder for this column

#save the encoders as a pickle file
with open("encoders.pkl","wb") as f:
  pickle.dump(encoders,f)
```

```
encoders
```

```
{'gender': LabelEncoder(),
 'ethnicity': LabelEncoder(),
 'jaundice': LabelEncoder(),
 'autism': LabelEncoder(),
 'country_of_res': LabelEncoder(),
 'used_app_before': LabelEncoder(),
 'relation': LabelEncoder()}
```

```
df.head()
```

	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	age	gender	ethnicity	jaundice
0	1	0	1	0	1	0	1	0	1	1	38	0	5	0
1	0	0	0	0	0	0	0	0	0	0	47	1	5	0
2	1	1	1	1	1	1	1	1	1	1	7	1	9	0
3	0	0	0	0	0	0	0	0	0	0	23	0	5	0
4	0	0	0	0	0	0	0	0	0	0	43	1	5	0

Next steps: Generate code with df View recommended plots New interactive sheet

Bivariate Analysis

```
#correlaltion matrix
plt.figure(figsize=(15,6))
sns.heatmap(df.corr(),annot=True,cmap="coolwarm",fmt=".2f")
plt.title("Correlation heatmap")
```