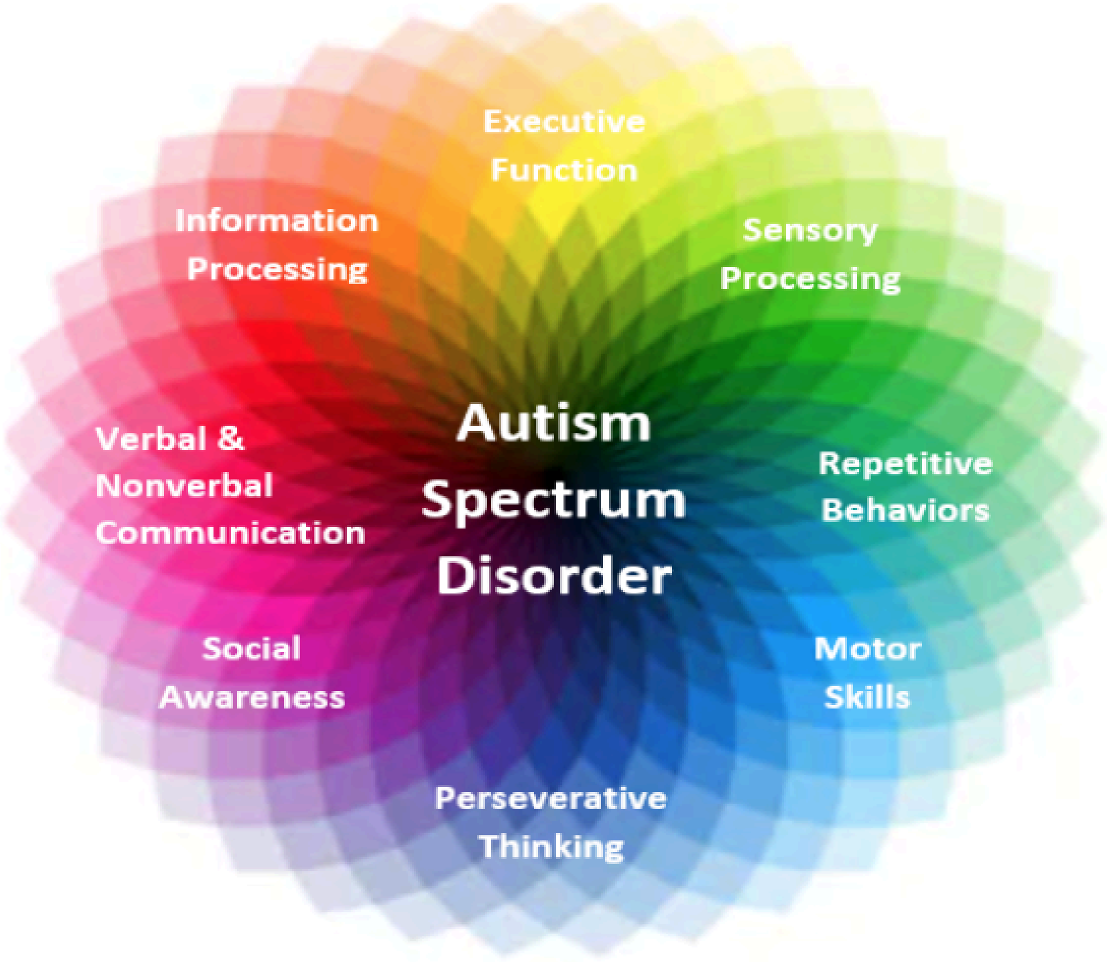
Case Study - Autism

Table of Contents

1. What is Autism ?

Autistic Spectrum Disorder (ASD) is a neurodevelopmental condition associated with significant healthcare costs, and early diagnosis can significantly reduce these. Unfortunately, waiting times for an ASD diagnosis are lengthy and procedures are not cost effective. The economic impact of autism and the increase in the number of ASD cases across the world reveals an urgent need for the development of easily implemented and effective screening methods. Therefore, a time-efficient and accessible ASD screening is imminent to help health professionals and inform individuals whether they should pursue formal clinical diagnosis. The rapid growth in the number of ASD cases worldwide necessitates datasets related to behavior traits. However, such datasets are rare making it difficult to perform thorough analyses to improve the efficiency, sensitivity, specificity and predictive accuracy of the ASD screening process. Presently, very limited autism datasets associated with clinical or screening are available and most of them are genetic in nature.



2. About the dataset

I have collected this dataset from kaggle and this dataset is related to autism screening of toddlers that contains influential features to be utilized for further analysis especially in determining autistic traits and improving the classification of ASD cases. In this dataset, they have recorded ten behavioral features plus other individuals characteristics that have proved to be effective in detecting the ASD cases from controls in behavior science.

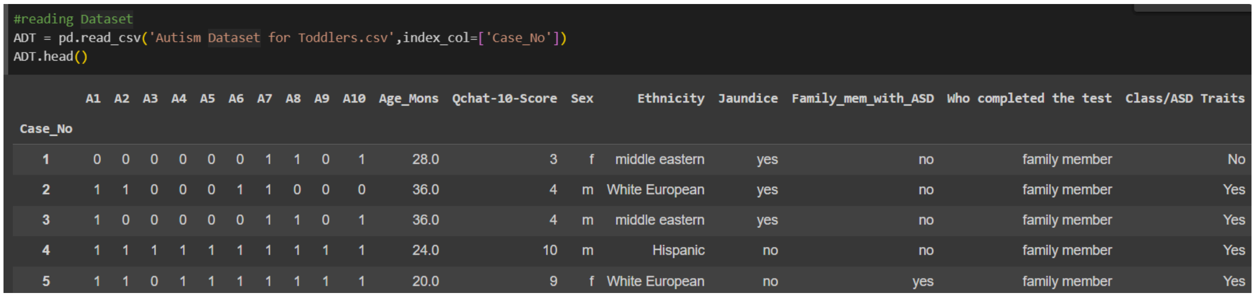
A1-A10: Items within Q-Chat-10 in which questions possible answers : “Always, Usually, Sometimes, Rarely & Never " items' values are mapped to"1 " or"0 " in the dataset. For questions 1-9 (A1-A9) in Q-chat-10, if the response was Sometimes / Rarely / Never “1” is assigned to the question (A1-A9). However, for question 10 (A10), if the response was Always / Usually / Sometimes then “1” is assigned to that question.

If the user obtained More than 3 Add points together for all ten questions. If your child scores more than 3 (Q-chat-10- score) then there is a potential ASD trait otherwise no ASD traits are observed. The other details are collected through an application. Here A1-A9 are described below :

A1 - Does your child look at you when you call his/her name?
A2 - How easy is it for you to get eye contact with your child?
A3 - Does your child point to indicate that s/he wants something? (e.g. a toy that is out of reach)
A4 - Does your child want to share interest with you? (e.g. pointing at an interesting sight)
A5 - Does your child pretend? (e.g. care for dolls, talk on a toy phone)
A6 - Does your child follow where you're looking?
A7 - If you or someone else in the family is visibly upset, does your child show signs of wanting to comfort them? (e.g. stroking hair, hugging)
A8 - Would you describe your child's first words as unusual?
A9 - Does your child use simple gestures? (e.g. wave goodbye)
A10 - Does your child stare at nothing with no apparent purpose?

3. Loading the dataset

Now for this case study I will use python as a programming language and apply Data Science techniques to get meaningful insights. So first I have downloaded the "Autism Dataset for Toddlers.csv" in the project directory where I am working from kaggle.

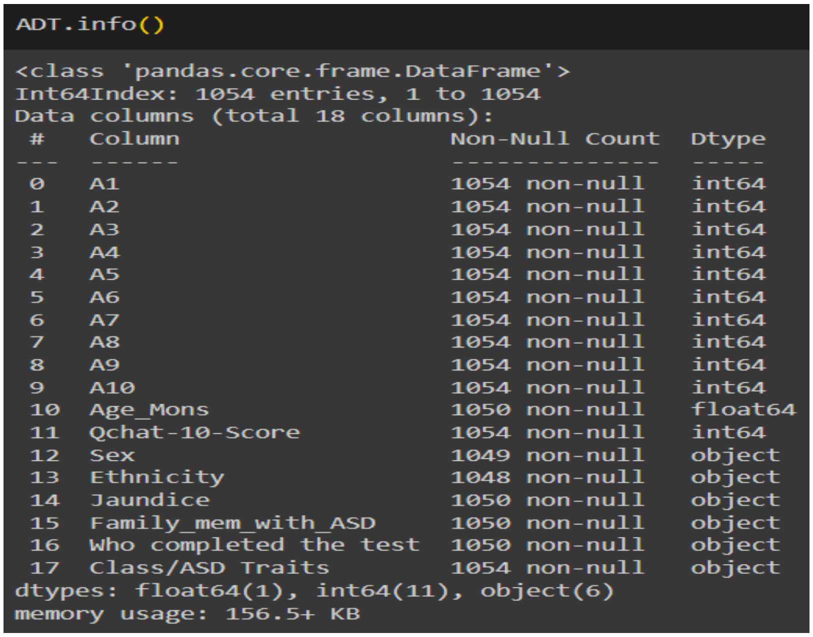


```
#reading Dataset
ADT = pd.read_csv('Autism Dataset for Toddlers.csv',index_col=['Case_No'])
ADT.head()
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	Age_Mons	Qchat-10-Score	Sex	Ethnicity	Jaundice	Family_mem_with_ASD	Who completed the test	Class/ASD Traits
Case_No																		
1	0	0	0	0	0	0	1	1	0	1	28.0	3	f	middle eastern	yes	no	family member	No
2	1	1	0	0	0	1	1	0	0	0	36.0	4	m	White European	yes	no	family member	Yes
3	1	0	0	0	0	0	1	1	0	1	36.0	4	m	middle eastern	yes	no	family member	Yes
4	1	1	1	1	1	1	1	1	1	1	24.0	10	m	Hispanic	no	no	family member	Yes
5	1	1	0	1	1	1	1	1	1	1	20.0	9	f	White European	no	yes	family member	Yes

4. Getting insights from the dataset

> Getting information about dimension and datatypes of each attribute.

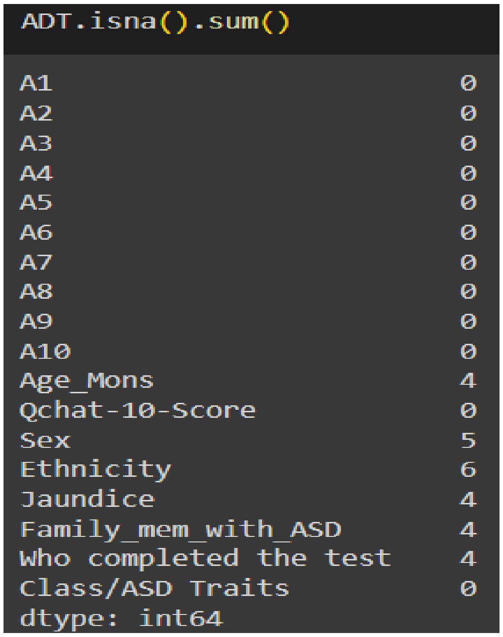


```
ADT.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1054 entries, 1 to 1054
Data columns (total 18 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   A1                      1054 non-null   int64  
 1   A2                      1054 non-null   int64  
 2   A3                      1054 non-null   int64  
 3   A4                      1054 non-null   int64  
 4   A5                      1054 non-null   int64  
 5   A6                      1054 non-null   int64  
 6   A7                      1054 non-null   int64  
 7   A8                      1054 non-null   int64  
 8   A9                      1054 non-null   int64  
 9   A10                     1054 non-null   int64  
 10  Age_Mons                1050 non-null   float64
 11  Qchat-10-Score          1054 non-null   int64  
 12  Sex                     1049 non-null   object 
 13  Ethnicity               1048 non-null   object 
 14  Jaundice                1050 non-null   object 
 15  Family_mem_with_ASD     1050 non-null   object 
 16  Who completed the test  1050 non-null   object 
 17  Class/ASD Traits        1054 non-null   object 
dtypes: float64(1), int64(11), object(6)
memory usage: 156.5+ KB
```

So from above we can see there are 1054 total entries and 18 attributes.

> Checking About missing values

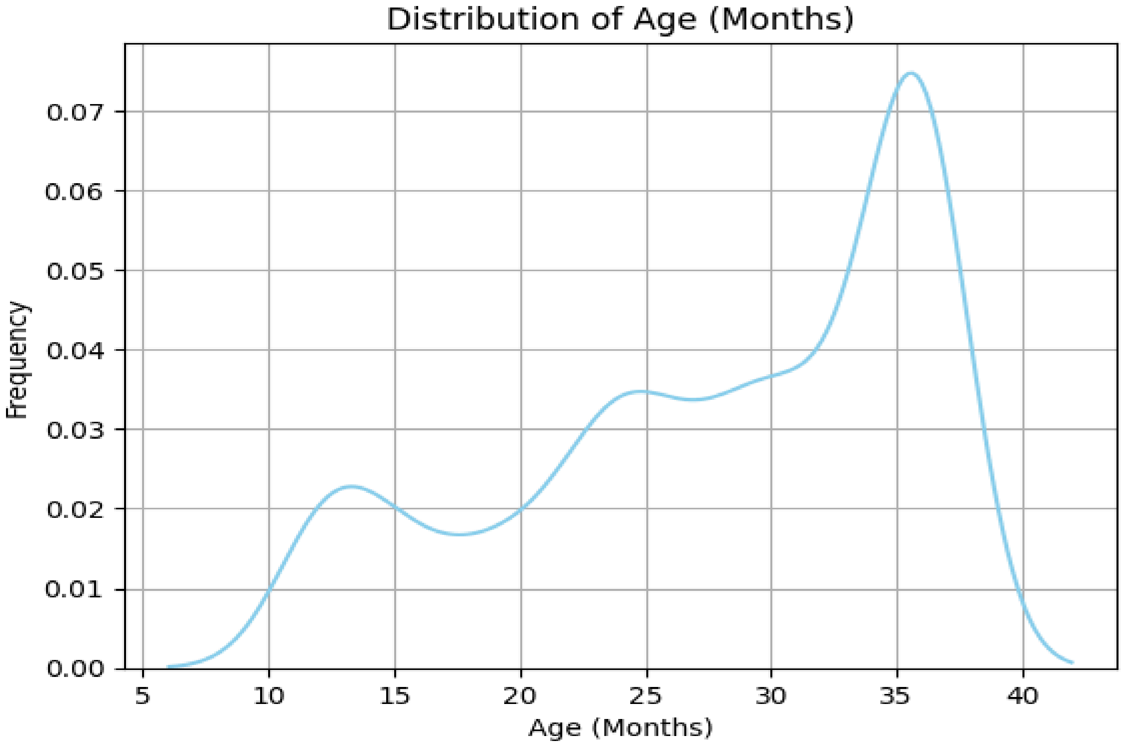


```
ADT.isna().sum()
```

```
A1                        0
A2                        0
A3                        0
A4                        0
A5                        0
A6                        0
A7                        0
A8                        0
A9                        0
A10                       0
Age_Mons                  4
Qchat-10-Score            0
Sex                       5
Ethnicity                 6
Jaundice                  4
Family_mem_with_ASD       4
Who completed the test    4
Class/ASD Traits          0
dtype: int64
```

From above we can see that the columns “Age_Mons”, “sex”, “Ethnicity”, “Jaundice”, “Family_mem_with_ASD”, “who completed the test” are the columns with missing values.

> So now to fill missing values of this columns we need to understand that attributes so the column except “Age_Mons” are categorical attributes so to fill them we can use forward_fill,backward_fill technique or else we can simple fill it with mode of that attribute and it would be more convenient to use it as no records are related to each other. So now if we talk about “Age_Mons”, first we need to visualize the distribution of data points in this attribute.



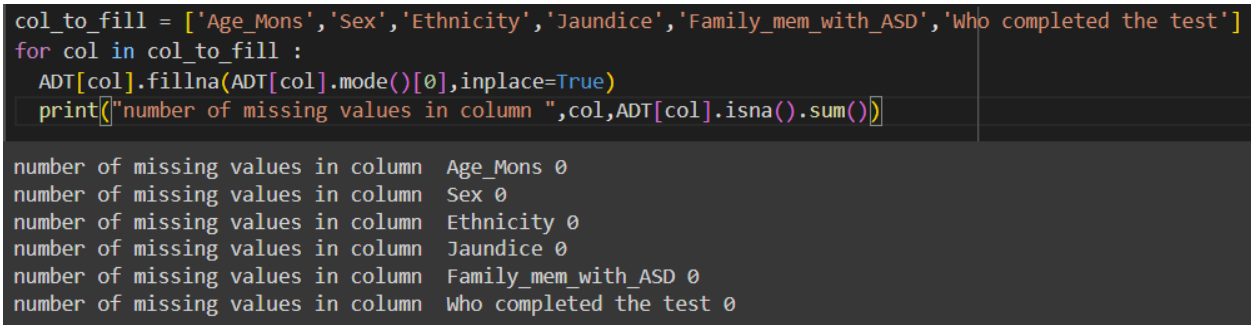
The observed data distribution indicates a negative skew, signifying a non-standard distribution. Given that this is a numerical variable, conventional methods such as mean or median may be appropriate for filling missing values. However, due to the observed skewness, utilizing the mode of the attribute for imputation is preferred.

5. Dataset Pre-Processing

It is an important step in the data science lifecycle, because due to noisy data we may not train models properly, we may miss out important information, data can be inconsistent so we need to handle them or else at last we would end up with a wrong result.

5.1 Handling Missing Values

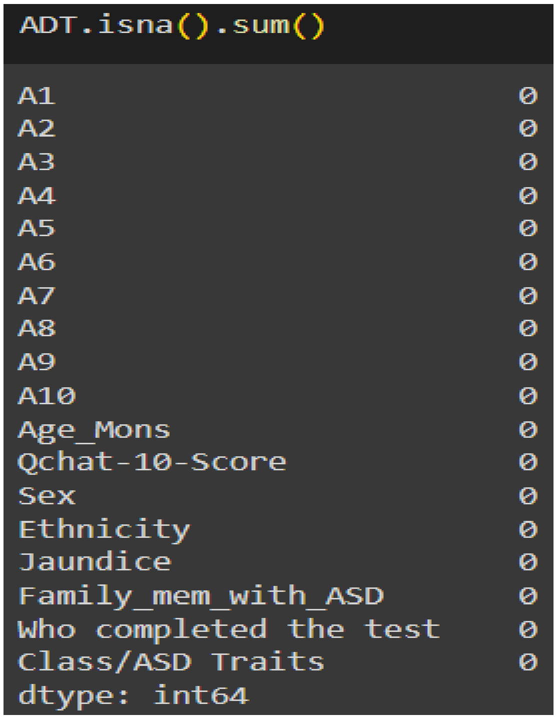
So as we discussed in the previous section of Gathering Insights to handle missing value by imputing it with that attribute mode so we will do it.



```
col_to_fill = ['Age_Mons','Sex','Ethnicity','Jaundice','Family_mem_with_ASD','Who completed the test']
for col in col_to_fill :
  ADT[col].fillna(ADT[col].mode()[0],inplace=True)
  print("number of missing values in column ",col,ADT[col].isna().sum())
```

```
number of missing values in column  Age_Mons 0
number of missing values in column  Sex 0
number of missing values in column  Ethnicity 0
number of missing values in column  Jaundice 0
number of missing values in column  Family_mem_with_ASD 0
number of missing values in column  Who completed the test 0
```

Verifying that now if there is any missing value present or not.



```
ADT.isna().sum()
```

```
A1                          0
A2                          0
A3                          0
A4                          0
A5                          0
A6                          0
A7                          0
A8                          0
A9                          0
A10                         0
Age_Mons                    0
Qchat-10-Score              0
Sex                         0
Ethnicity                   0
Jaundice                    0
Family_mem_with_ASD         0
Who completed the test      0
Class/ASD Traits            0
dtype: int64
```

So now there are no null values present.

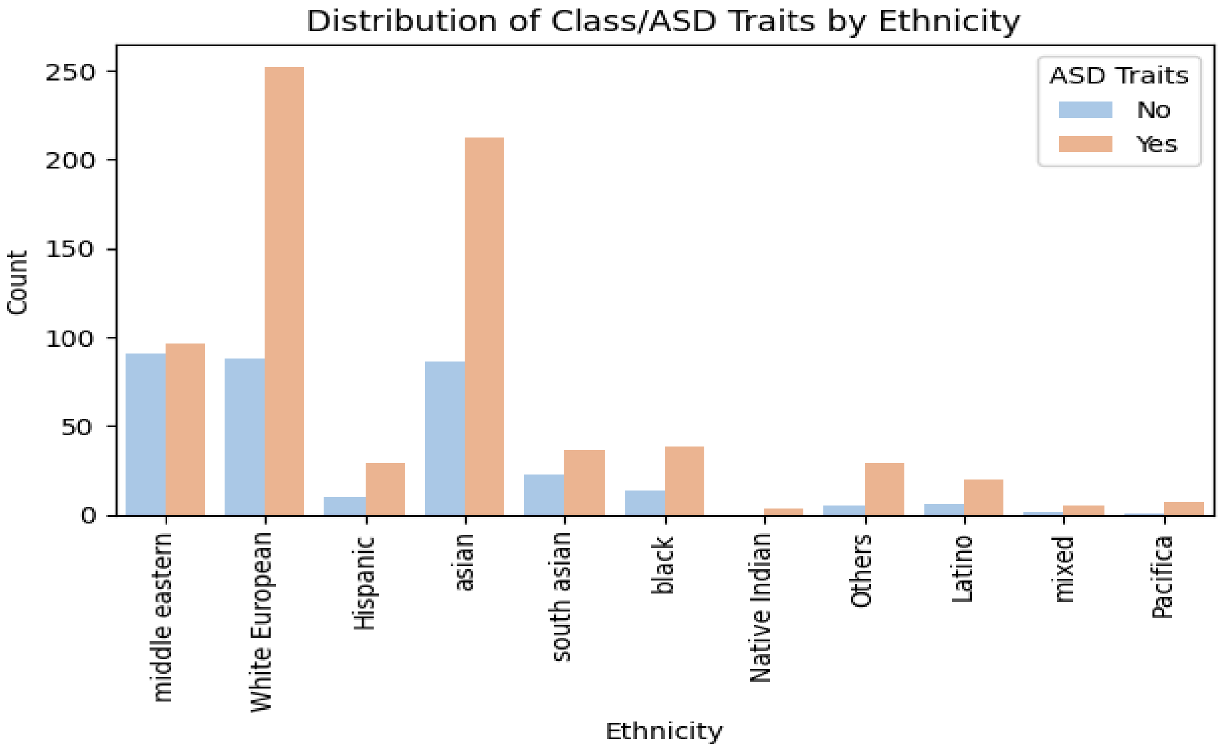
5.2 Dimensionality Reduction

It is an essential step of pre-processing as the dataset can contain redundant information, irrelevant information, there can be many attributes(here comparatively less in comparison to larger datasets). So we can now apply a bottom-up approach to reduce the dimension of the dataset.

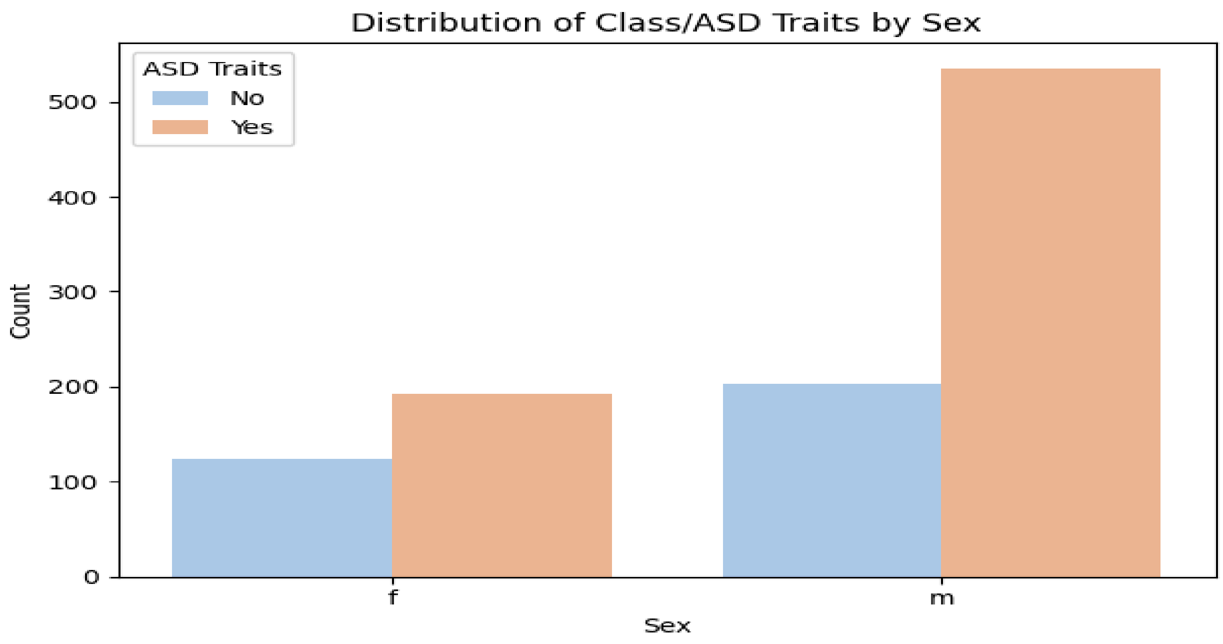
-> Now if we talk about columns A1 - A10 and Qchat-10-score then they all represent the same means the column Qchat-10-score is just a summation of values in A1 - A10. So it is better to drop all the A1 - A10 columns to reduce redundancy.

-> The columns "Age_Mons", "Qchat-10-score" are an important factor for early symptoms detection of Autism so we can't drop them.

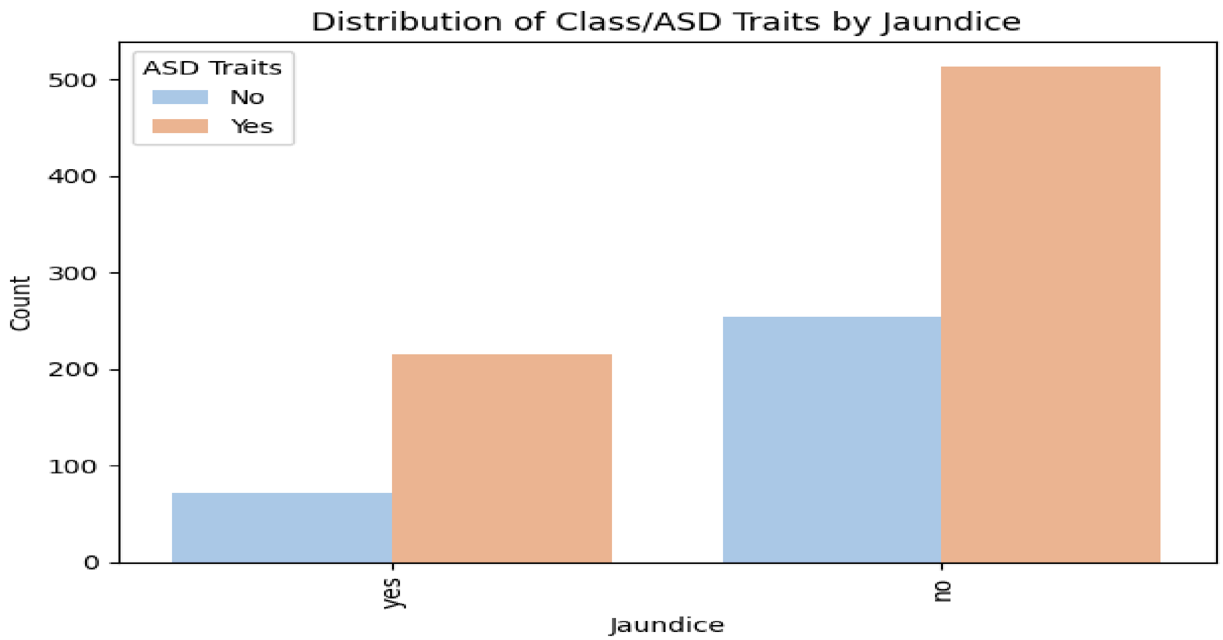
-> For rest columns we need to decide graphically whether the feature is important or not.



-> From the above Count Plot we can see that major number of cases of autism are from white European, Asian, Middle eastern Ethnicity so it is important factor for determining which child belongs to which Ethnicity.



-> From the Count Plot we can see that Autism is observed more in the boy child so again it is an important factor.

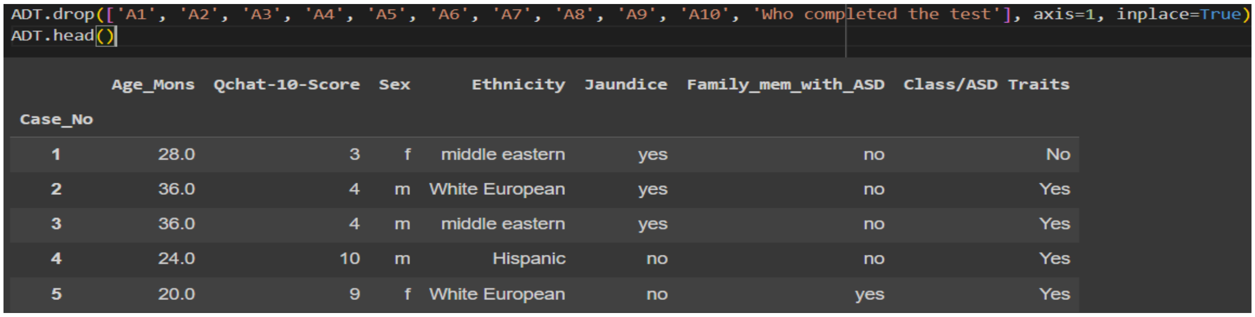


-> Above Count Plot says that the childrens whom are not suffered from Jaundice has suffered from Autism so we cant ignore this relationship.

-> Now if we talk about the column “Who completed the test” than for these we did not need to visualize as it doesn't matter the test result who completed the test.

-> And the last column “Class/ASD Traits” it is our target attribute so it has no reason to drop.

-> So finally from all above decisions we have decided to drop 11 columns.

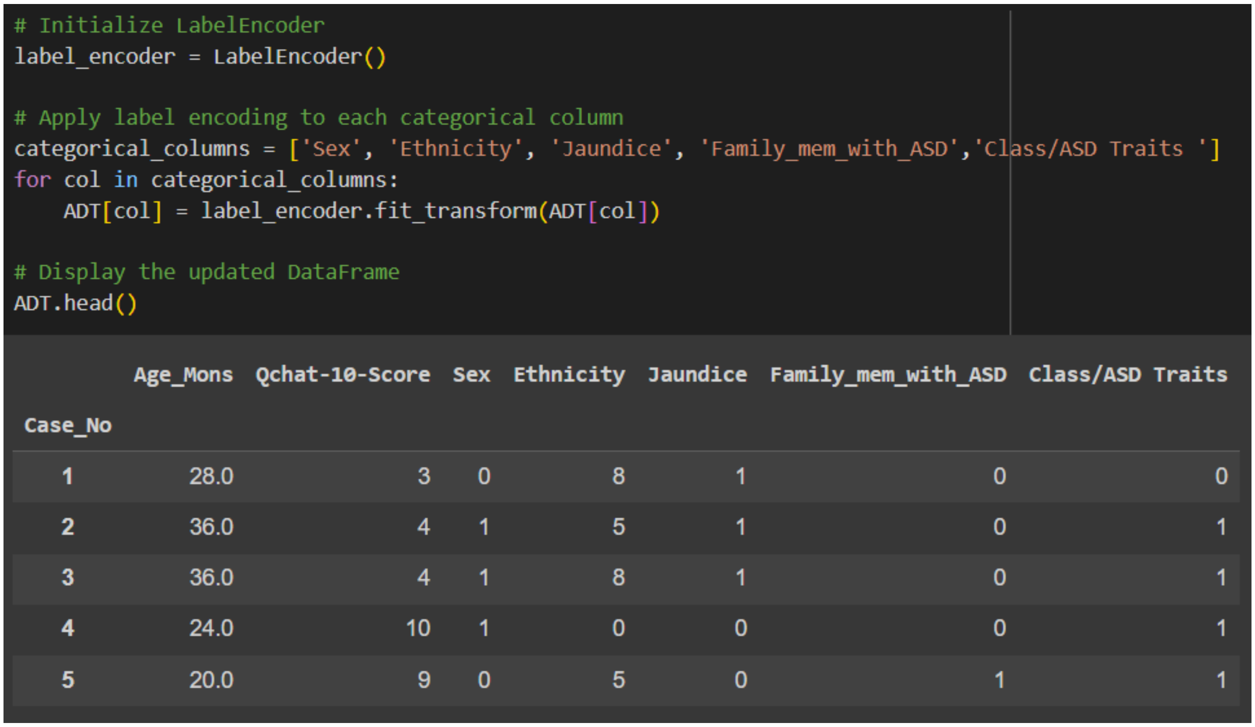


```
ADT.drop(['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'Who completed the test'], axis=1, inplace=True)
ADT.head()
```

	Age_Mons	Qchat-10-Score	Sex	Ethnicity	Jaundice	Family_mem_with_ASD	Class/ASD Traits
Case_No							
1	28.0	3	f	middle eastern	yes	no	No
2	36.0	4	m	White European	yes	no	Yes
3	36.0	4	m	middle eastern	yes	no	Yes
4	24.0	10	m	Hispanic	no	no	Yes
5	20.0	9	f	White European	no	yes	Yes

5.3 Encoding

From above we can see that many attributes are categorical and many machine learning models do not work with categorical attributes and I am going to use three models “Logistic Regression”, “SVM”, “XGBOOST” in which “XGBOOST” can't work with categorical data so we need to encode the data. There are many techniques available for encoding and I will use Label encoding techniques to encode data. So after all Pre-Processing step our dataset will look like :

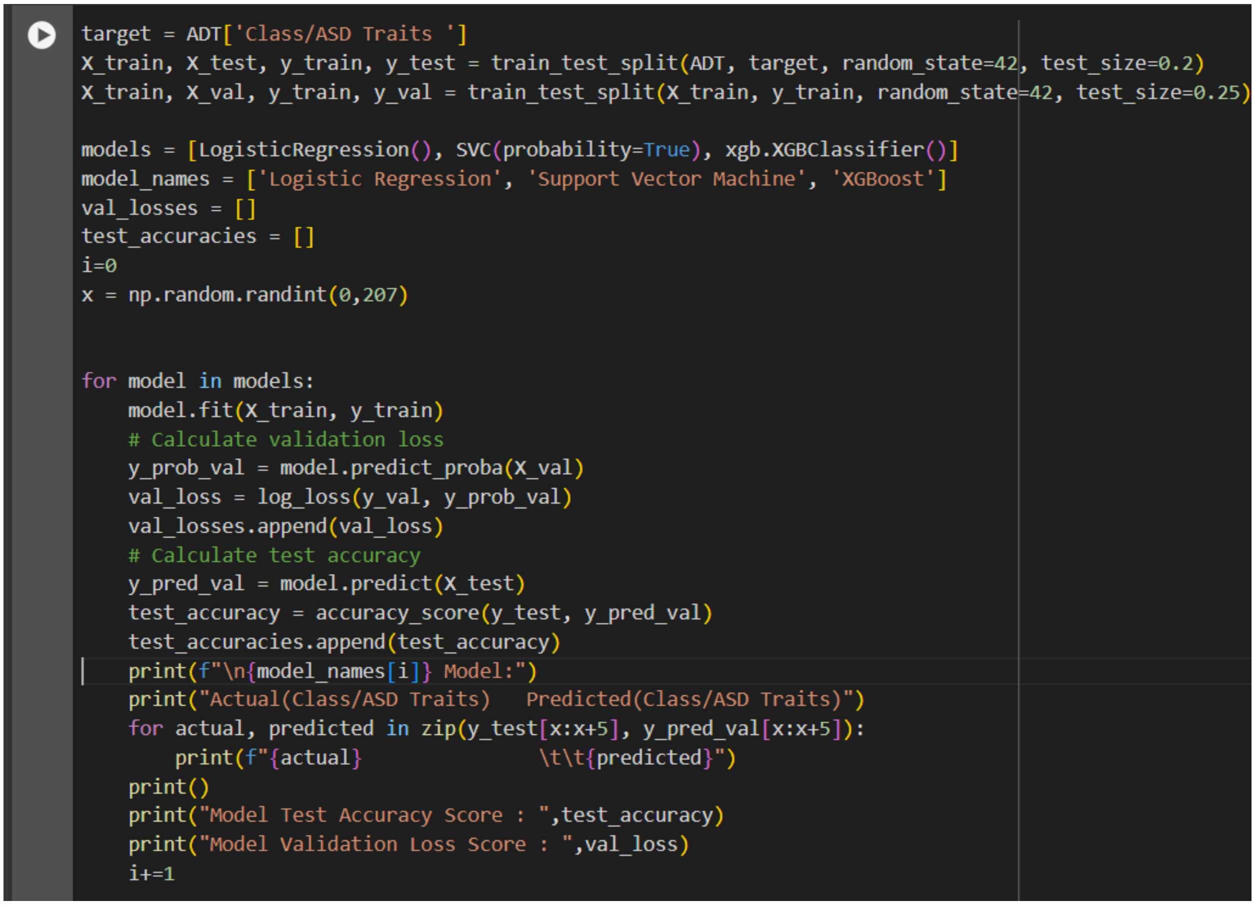


```
# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column
categorical_columns = ['Sex', 'Ethnicity', 'Jaundice', 'Family_mem_with_ASD','Class/ASD Traits ']
for col in categorical_columns:
    ADT[col] = label_encoder.fit_transform(ADT[col])

# Display the updated DataFrame
ADT.head()
```

	Age_Mons	Qchat-10-Score	Sex	Ethnicity	Jaundice	Family_mem_with_ASD	Class/ASD Traits
Case_No							
1	28.0	3	0	8	1	0	0
2	36.0	4	1	5	1	0	1
3	36.0	4	1	8	1	0	1
4	24.0	10	1	0	0	0	1
5	20.0	9	0	5	0	1	1

6. Training the model

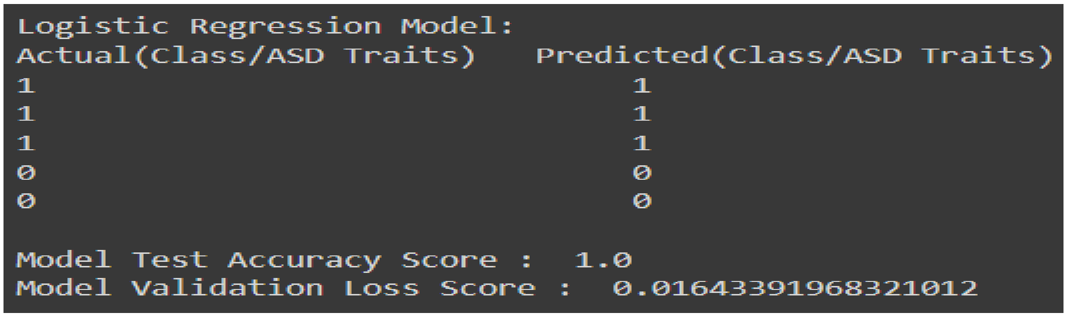
As discussed earlier I have used 3 models and these are the most popular classification models each with different working. For comparing performance of these models we have many parameters but I will use two validation losses and test accuracy. For training and performance measurement we need to split the dataset into 3 sets: training, validation and testing set; training set for fighting the model, validation set for measuring validation loss and testing set for predicting(evaluating) the model. Here we split the dataset into standard 80 : 20 ratio of train and test size and again from 80% of train set we have split 25% data into validation set.



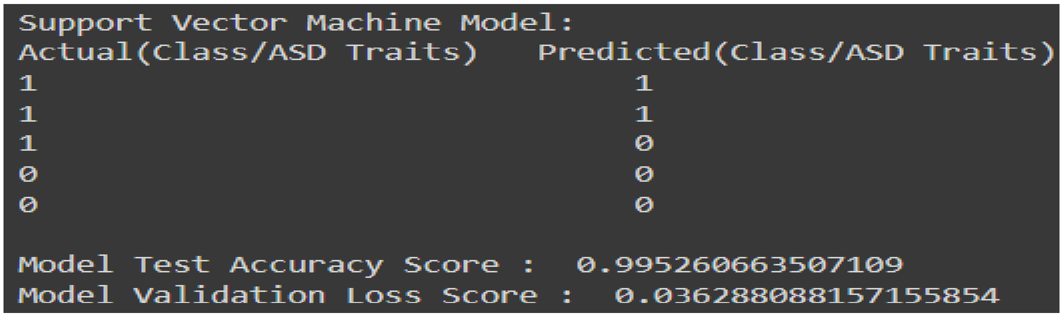
```
target = ADT['Class/ASD Traits ']
X_train, X_test, y_train, y_test = train_test_split(ADT, target, random_state=42, test_size=0.2)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, random_state=42, test_size=0.25)

models = [LogisticRegression(), SVC(probability=True), xgb.XGBClassifier()]
model_names = ['Logistic Regression', 'Support Vector Machine', 'XGBoost']
val_losses = []
test_accuracies = []
i=0
x = np.random.randint(0,207)


for model in models:
    model.fit(X_train, y_train)
    # Calculate validation loss
    y_prob_val = model.predict_proba(X_val)
    val_loss = log_loss(y_val, y_prob_val)
    val_losses.append(val_loss)
    # Calculate test accuracy
    y_pred_val = model.predict(X_test)
    test_accuracy = accuracy_score(y_test, y_pred_val)
    test_accuracies.append(test_accuracy)
    print(f"\n{model_names[i]} Model:")
    print("Actual(Class/ASD Traits)   Predicted(Class/ASD Traits)")
    for actual, predicted in zip(y_test[x:x+5], y_pred_val[x:x+5]):
        print(f"{actual}               \t\t{predicted}")
    print()
    print("Model Test Accuracy Score : ",test_accuracy)
    print("Model Validation Loss Score : ",val_loss)
    i+=1
```

7. Evaluating the Model

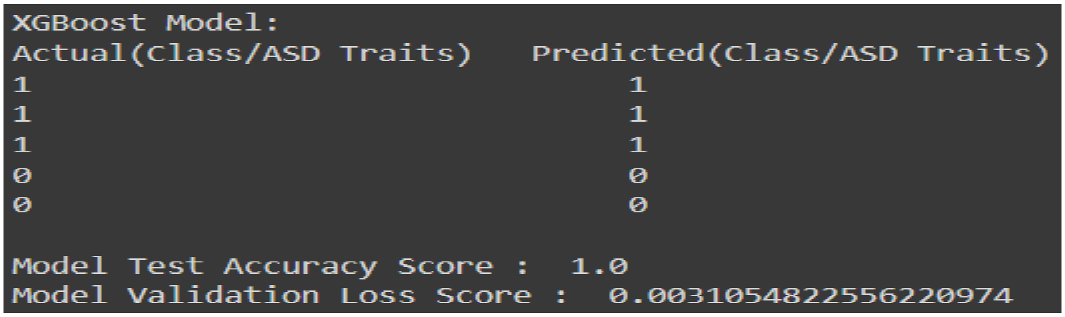
From the above section code snippet you can see that I have chosen the random number x according to the size of the test set and below I have shown Actual ASD traits and prediction result for the 5 records from that x. Here label “1” indicates that the child is suffering from ASD and “0” indicates there are no ASD traits present.



```
Logistic Regression Model:
Actual(Class/ASD Traits)   Predicted(Class/ASD Traits)
1                               1
1                               1
1                               1
0                               0
0                               0

Model Test Accuracy Score :  1.0
Model Validation Loss Score :  0.01643391968321012
```

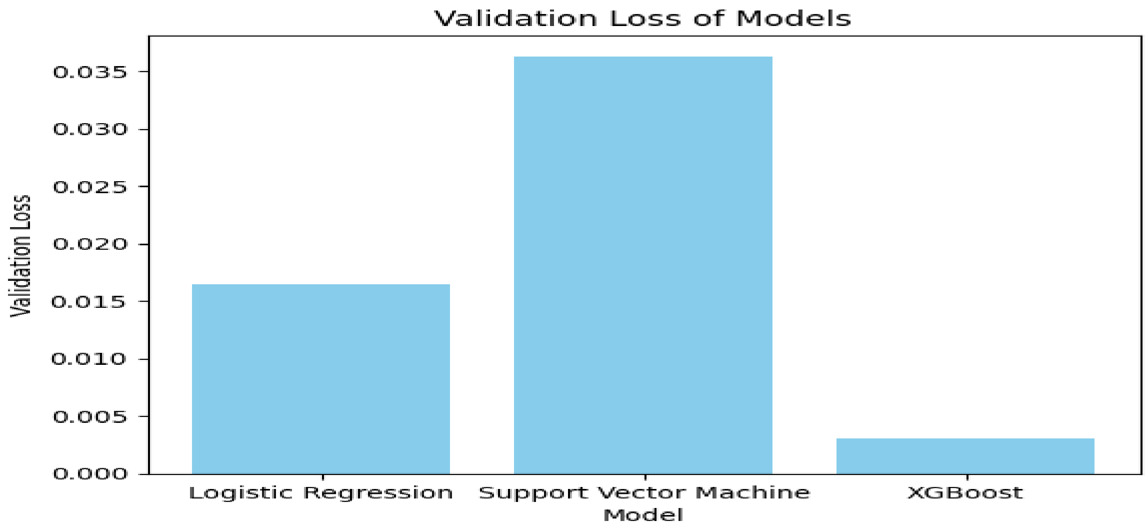


```
Support Vector Machine Model:
Actual(Class/ASD Traits)   Predicted(Class/ASD Traits)
1                               1
1                               1
1                               0
0                               0
0                               0

Model Test Accuracy Score :  0.995260663507109
Model Validation Loss Score :  0.036288088157155854
```

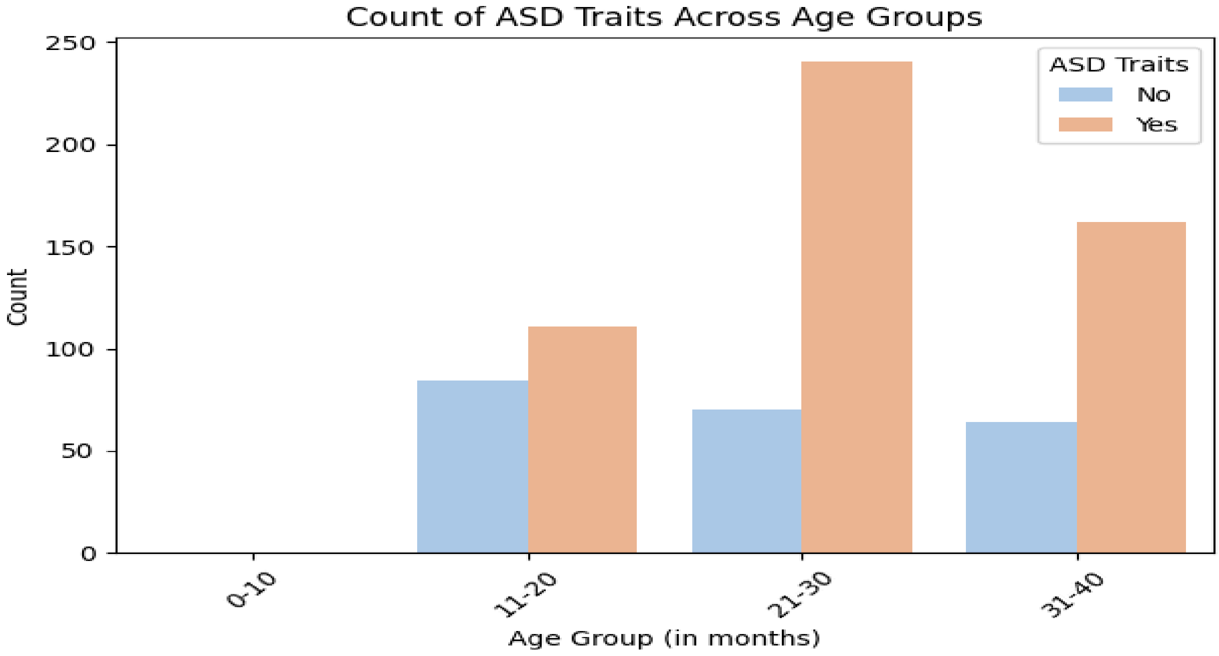


```
XGBoost Model:
Actual(Class/ASD Traits)   Predicted(Class/ASD Traits)
1                               1
1                               1
1                               1
0                               0
0                               0

Model Test Accuracy Score :  1.0
Model Validation Loss Score :  0.0031054822556220974
```

Evaluation of models based on accuracy than all models are accurate except SVM and it is wrong only for 1 or 2 test case and if we talk about validation loss than we can see than “XGBoost” model here wins the race it validation loss score is very less compared rest two models seeing numerical value may not differentiate the difference so see below bar graph to have clear vision.



8. Conclusion



From above study I am concluding that XGBoost model is robust model for this classification task of Autism Spectrum Disorder detection and Major characteristics for early detection and prevention of ASD are if “Qchat-10-Score” is greater than 2 and “Age_Mons” is in the 21 - 30 months and Ethnicity belongs to Middle Eastern/ White European/ Asian and Gender is male and Jaundice is not occurred to child than their is very much high chance of that child to suffer from ASD.