Detecting Autism

Sonia[1] Priyanka[2] Aakanksha Kumari[3] Kanika Kashyap[4] Nidhi Gangwar[5] and Himanshi Gupta[6]

Indira Gandhi Delhi Technical University for Women, Delhi, India
{aakanksha17319@gmail.com,himanshi.july@gmail.com,
soniaprasad1997@gmail.com,subramanipriyanka3@gmail.com,kanikakashyap205@gmail.com,
nidhigang2707@gmail.com,}

Abstract. In present day Autism Spectrum Disorder (ASD) is gaining its momentum faster than ever. Detecting autism traits through screening tests is very expensive and time consuming. With the advancement of artificial intelligence and machine learning (ML), autism can be predicted at quite early stage. Though number of studies have been carried out using different techniques, these studies didn't provide any definitive conclusion about predicting autism traits in adults.Therefore this paper aims to propose an effective prediction model based on ML technique. The proposed model was evaluated with AQ-10 dataset and 704 real dataset collected from people with and without autistic traits. The evaluation results showed that the proposed prediction model provide better results in terms of accuracy, specificity, sensitivity, precision for both kinds of data sets.

1 Introduction

Autism spectrum disorder is a neural developmental disorder that affects a person's interaction, communication and learning skills. Although diagnosis of autism can be done at any age, its symptoms generally appear in the first two years of life and develops through time. Autism patients face different types of challenges such as difficulties with concentration, learning disabilities, mental health problems such as anxiety, depression etc, motor difficulties, sensory problems and many others. Current explosion rate of autism around the world is numerous and it is increasing at a very high rate. According to WHO, about 1 out of every 160 children has ASD. Some people with this disorder can live independently, while others require life-long care and support. Diagnosis of autism requires significant amount of time and cost. Earlier detection of autism can come to a great help by prescribing patients with proper medication at an early stage. It can prevent the patient's condition from deteriorating further and would help to reduce long term costs associated with delayed diagnosis. Thus a time efficient, accurate and easy screening test tool is very much required which would predict autism traits in an individual and identify whether or not they require comprehensive autism assessment.

Problem Statement. *Autism spectrum disorder impacts the nervous system and affects the overall cognitive, emotional, social and physical health of the affected individual. Number of questions are asked from a person and on the basis of score he gained we predict if the person has autism.*

2 Methodology

2.1 Data Collection

To develop an effective predictive model, AQ-10 dataset was used which consists of three different datasets based on AQ-10 screening tool questions [16]. The dataset contain data of age group 18 or more (adult). AQ-10 or Autism Spectrum Quotient tool is used to identify whether an individual should be referred for a comprehensive autism assessment.AQ-10 screening questions focus on different domains such as- attention to detail, attention switching, communication,imagination and social interaction. Scoring method of the questions is that only 1 point can be scored for each of the 10 questions. User may score 0 or 1 point on each question based on their answer [17]. Datasets of child, adolescent and adult contain 292, 104 and 704 instances respectively. The dataset contains twenty-one attributes which are a mix of numerical and categorical data, that includes: Age, Gender,Ethnicity, If born with Jaundice, Family member with PDD, Who is completing the test, Country of Residence, Used the screening app before, Screening method type, Question 1-10,Result and Class.

2.2 Data Synthesization

The collected data were synthesized to remove irrelevant features. For example, the ID column was irreverent to develop a prediction model, thus it was removed. To handle null values,list wise deletion technique was applied where a particular observation was deleted if it had one or more missing values.Then to extract unnecessary features from the dataset, decision tree algorithm was used. Results showed dropping 'relation','age desc', 'used app before' and 'age' columns would result in more accurate classification and so those columns were dropped. Summary of the synthesized datasets are shown in Table I.

2.3 Developing the Prediction Model

To generate prediction of autism traits, algorithms had been developed and their accuracy were tested. After attaining results from various types of supervised learning like Linear Regression, SVM, Naive Bayes; Random Forest was found to be highly feasible with higher accuracy than the other algorithms. So, Random Forest (CART) was proposed for implementing the ASD predictive system. Further modifications were made to the algorithm to attain even better results.

2.4 Evaluating the Prediction Model

The proposed predictive model was tested with the AQ-10 dataset and data collected from real-world in terms of the accuracy, specificity, precision, sensitivity and false positive rate. For the AQ-10 dataset, leave-one-out technique was also applied to check effectiveness of the proposed model. Again,to validate the proposed model almost 100 data of ASD cases were collected from an institute of special education for the people with special needs and 150 data of Non-ASD cases were collected through field visit to different schools and shopping malls, using both printed forms and online forms. In later case, the online questionnaires were distributed through social media and email to different administrative and teaching communities.

2.5 Dataset Description

Details	Count
Number of instances	704
Number of attributes	21

Table 1. Details of the dataset

Data Attributes	Brief Explanation
A1 Score	Question 1 Answer Binary (0, 1)
A2 Score	Question 2 Answer Binary (0, 1)
A3 Score	Question 3 Answer Binary (0, 1)
A4 Score	Question 4 Answer Binary (0, 1)
A5 Score	Question 5 Answer Binary (0, 1)
A6 Score	Question 6 Answer Binary (0, 1)
A7 Score	Question 7 Answer Binary (0, 1)
A8 Score	Question 8 Answer Binary (0, 1)
A9 Score	Question 9 Answer Binary (0, 1)
A10 Score	Question 10 Answer Binary (0, 1)
Age	Age of the person
Gender	Gender of the Person
Ethnicity	String List of common ethnicities
Jundice	(yes or no) Whether the case was born with jaundice
Austim	Whether autism patient or not
country of res	Name of the country
used app before	Whether application was used before or not
result	The final score obtained based on the screening
age desc	Belongs to which group of age
relation	Who is completing the test String Parent, self, caregiver, medical staff, clinician ,etc
Class/ASD	Belongs to which class YES or NO

Table 2. Details of Data Attributes.

3 Data Exploration

3.1 Metrics

Attribute	Mean	Median	Mode
A1_Score	0.721591	1.0	0
A2_Score	0.453125	0.0	1
A3_Score	0.457386	0.0	0
A4_Score	0.495739	0.0	0
A5_Score	0.498580	0.0	0
A6_Score	0.284091	0.0	0
A7_Score	0.417614	0.0	0
A8_Score	0.649148	1.0	1
A9_Score	0.323864	0.0	0
A10_Score	0.573864	1.0	1
result	4.875000	4.0	4

Fig. 1. Metrics

3.2 Visualizations

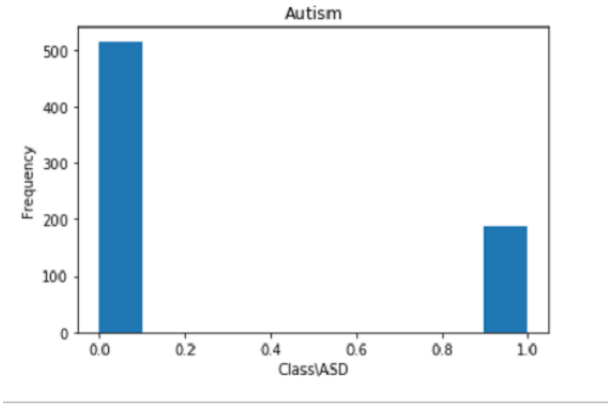

Fig. 2. Number Of Patients
Graph shows the line plot between the Class/ASD on X-Axis and Frequency on Y-axis. On x-axis 0 represent who does not have autism and 1 represent who have autism

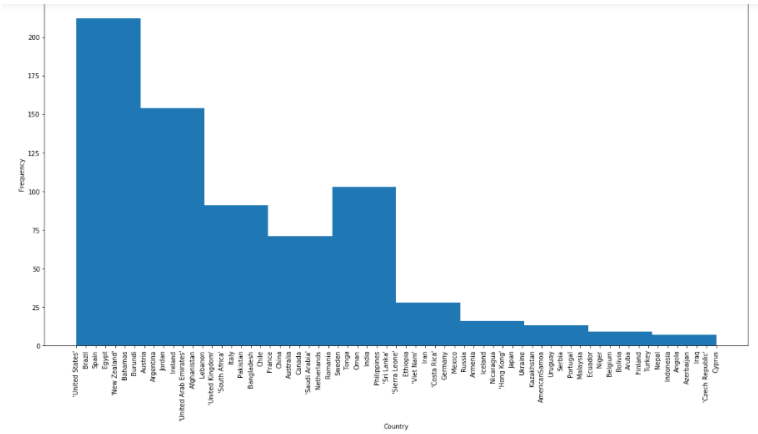

Fig. 3. Number Of Country
Graph shows the line plot between the Country on X-Axis and Frequency on Y-axis. On x-axis Various countries who is autism patient.

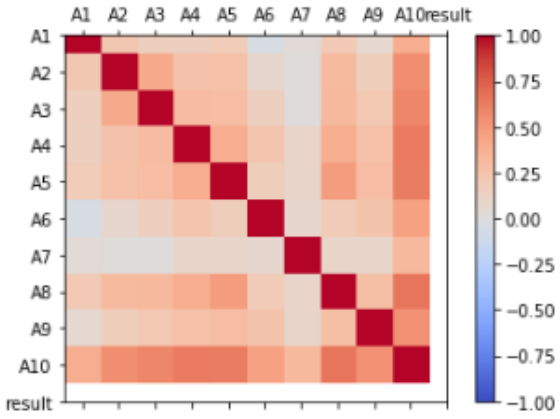

Fig. 4. Correlation Matrix
Graph shows correlation between questions and result.

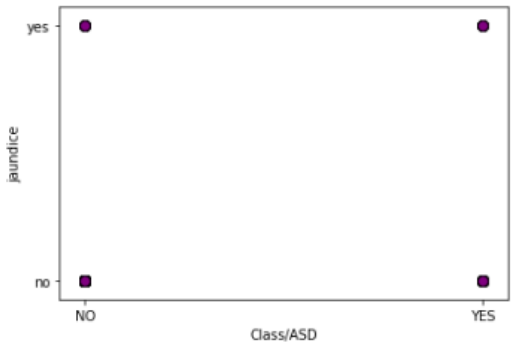

Fig. 5. Jaundice Vs Autism
Graph shows the scatter plot between jaundice patients on Y-axis and Class/ASD on X-axis.

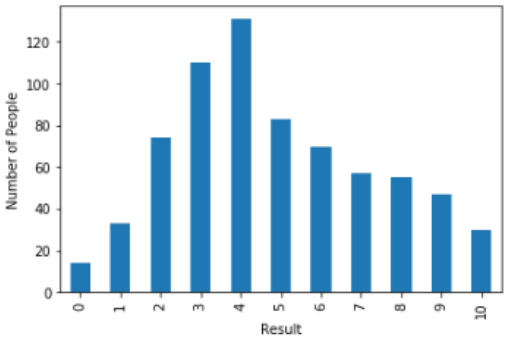

Fig. 6. Score Of Patient To Detect Autism
Graph shows bar graph between number of people on Y-axis and result i.e. score on the basis of questions on X -axis.

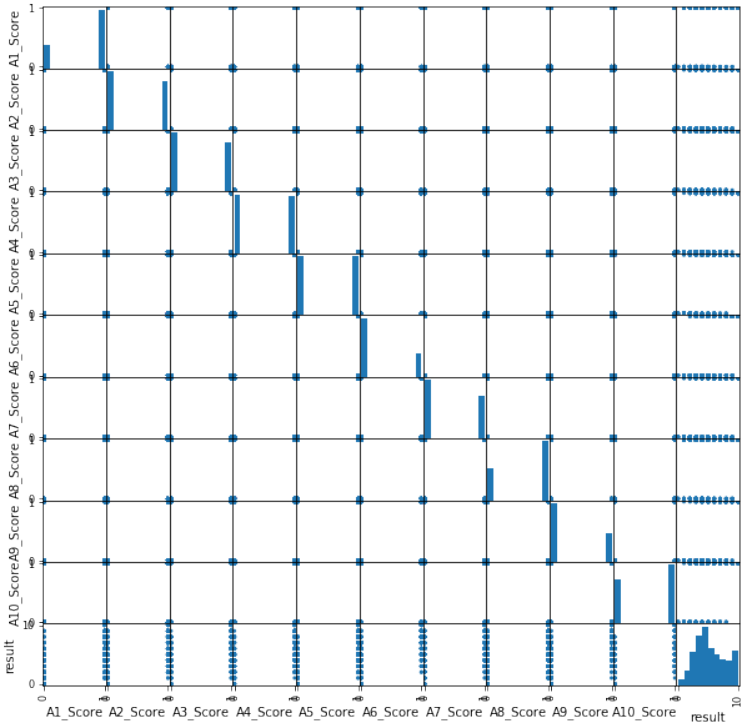

Fig. 7. Scatter Matrix
Graph shows scatter matrix between colmuns of dataset and Diagonals represents yes and no of each attribute from A1score to A2 score and last diagonal which is result shows the overall score of A1 to A10.

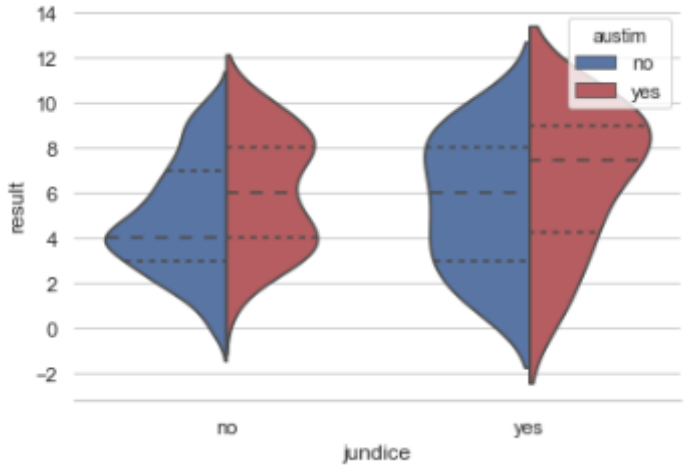

Fig. 8. Violin Plot
In the specified Quartile, the graph represents the patients from "Autism" stating yes and no who lie in the quartile NO and YES of "jaundice".

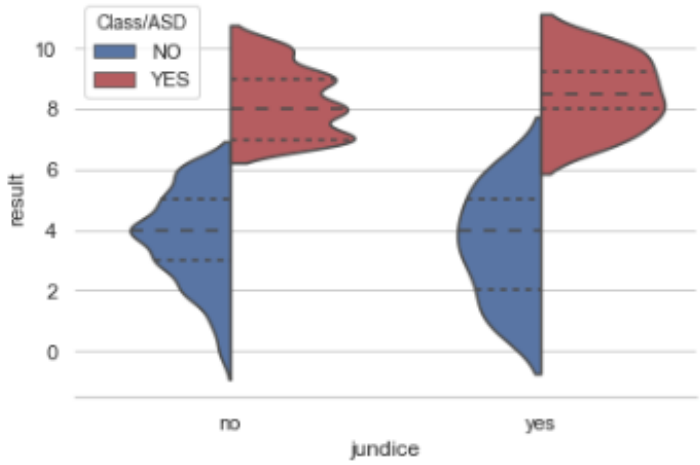

Fig. 9. Violin Plot
In the specified Quartile, the graph represents the patients from "Class/ASD" stating yes and no who lie in the quartile NO and YES of "jaundice".

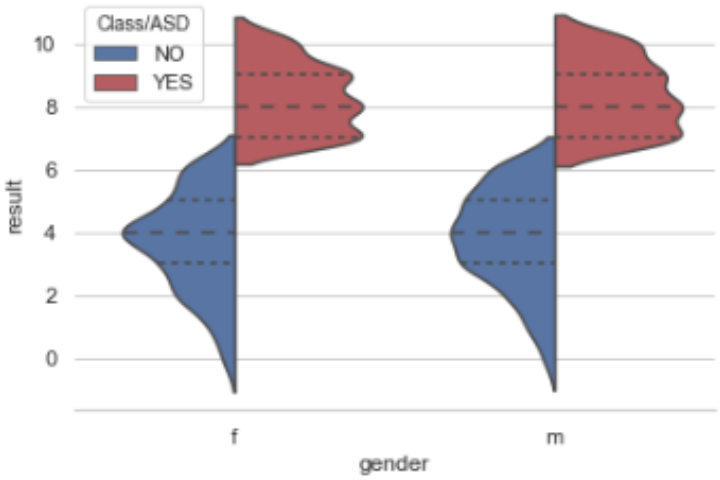

Fig. 10. Violin Plot
In the specified Quartile, the graph represents the patients from "Class/ASD" stating yes and no who lie in the quartile F and M of "Gender".

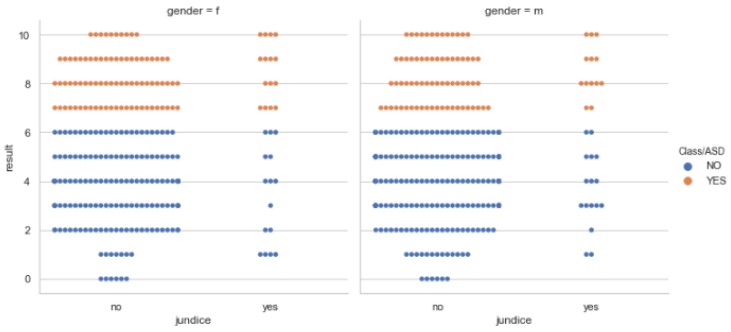

Fig. 11. Factor Plot
In this case, we can see when 'jaundice' is present at birth, an individual with a higher 'result' score will have autism irrespective of their gender.

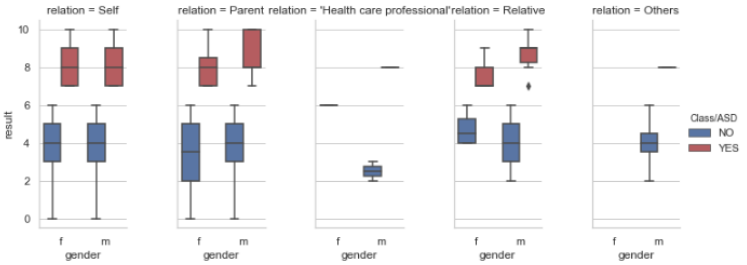

Fig. 12. Factor Plot
The boxplots in red show the distributions of the data which belongs to the 'ASD class' whereas the blue one showing the distributions of the data which are non-ausitic. This gives us our first impression of the internal connections of some of the above mentioned features that are present in our data set.

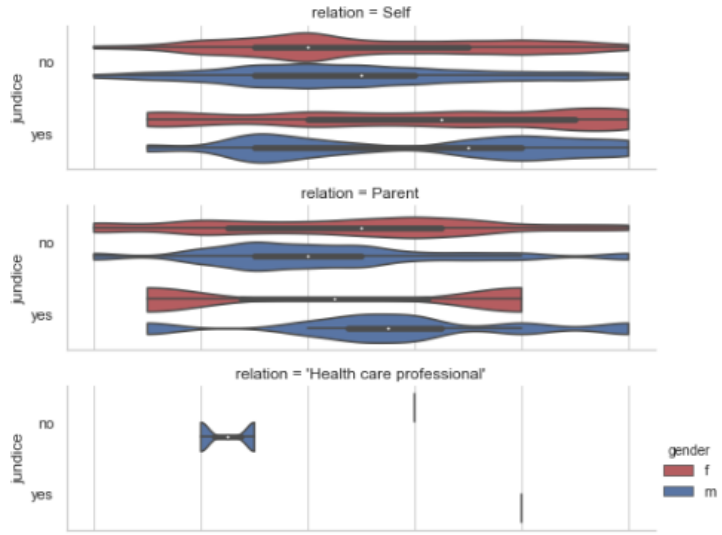

Fig. 13.

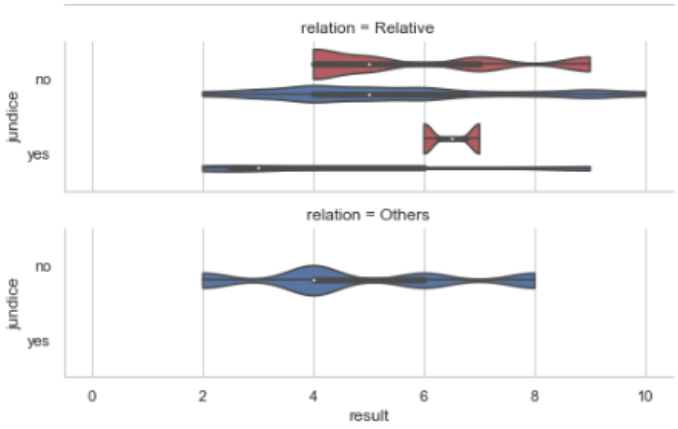

Fig. 14. A violine plot depicting multi-feature relationship.
These dictiction excersice is a great way to understand the data anatomy before we decide to apply algorithms which will be most suitable for our goal.

4 Proposed Approach

At first Decision Tree-CART algorithm was implemented to predict autism traits in an individual. For further improvement Random Forest-CART was implemented and better results were obtained. The four algorithms consecutively used to implement the system are discussed below:

4.1 Prediction model based on Decision Tree-CART

Initially Decision Tree-CART classifier was selected to create the prediction model.

At the beginning, the tree root consists of whole dataset. Then data would be

$$Gini(data) = 1 - \sum_{i \in unique\_classes} P(i)^2 \quad (1)$$

$$InfoGain(data, featureX) = Gini(data) - \sum_{i \in featureX} AvgGini(i)| \quad (2)$$

Fig. 15.

split using the best feature. The splitting process will continue recursively until a node consists of data of a unique label class. Sequential attribute selection method is resolved by Gini Impurity and Information Gain (IG) .

Attribute with maximum IG will be chosen first to split data. Algorithm used here [Algorithm 1] can be split into two phases:

Building decision tree [line number 3-13] and classifying test data using tree [line number 15-20].
The followed steps are given below:
•Initially best features were selected to construct the decision tree [Line 1] and the class labels were segregated[Line 2].

•To construct a decision tree, training data is called from'BUILD TREE' function [Line 3]. Then each feature from data is iterated and the feature with max IG is identified [Line 4-6]. If max IG equals zero then that means the class labels of that portion of data is pure and will return as leaf nodes [Line 7-9].

•If max IG is not equal to zero then the data will be split into two portions(TrueRows and FalseRows) with respect to the feature with max information gain [Line 10].

•'BUILD TREE' function will run recursively on both portions of the data [Line 11-12] and the two branches will form a decision node or rule [Line 13].

•Finally after the decision tree is constructed, test data is classified using it. The tree is iterated using the feature values . When tree reaches a leaf node then it will classify the test data with the leaf's prediction [Line 15-20].

Algorithm 1 Decision Tree CART Classifier

$features \leftarrow$ AQ10 questions,gender,inheritance
$classes \leftarrow$ yes($autistictraits$),no($noautistictraits$)
BUILD TREE($rows$)
for each possible features do **do**
    calculate max gain
end for
if **then**$maxgain$ =0
    return $leaf$
end if
$TrueRows$, $FalseRows$←Partition($rows$)TrueBranch← $BuildTree$(TrueRows)

$FalseBranch \leftarrow BuildTree$(FalseRows)
return $DecisionNode$(TrueBranch,FalseBranch)
procedure CLASSIFY(row,$node$)
    if $node = leaf$ **then**
        return node.predictions
        $Iterate_Tree$
    end if

4.2 Prediction model based on Random Forest-CART

In a random forest, each node is split using the best among a subset of predictors randomly chosen. This somewhat counter intuitive strategy turns out to perform very well compared to many other classifiers, including discriminant analysis,support vector machines and neural networks, and is robust against over-fitting [18].

To make the predictive model more accurate, Random Forest-CART classifier [Algorithm 2] was implemented. Here also the algorithm can be split into two phases: generating random forest [line number 1-10] and classifying test data [line number 12-28].

Classification using the random forest has been done following the steps below:
•At first, an array named 'tree array' is initialized as null to store the decision trees [Line 3].

•Then to generate 'p' number of decision trees of the forest, 'BUILD TREE' function is called 'p' times and the generated trees are stored in 'tree array' [Line 4-9].

•Each decision tree is generated for 'n' number of random attributes. Construction of decision tree procedure is same as described in Line 1-13 of Algorithm 1.

•Finally to classify a test data, votes are taken from each decision tree of the random forest. If majority of votes are "Yes" then we'll classify test data as "Yes"(Probable autistic traits) or else we'll classify test data as "No"(No autistic traits) [Line 12-28].

Algorithm 2 Random Forest CART Classifier

```
procedure BUILD FOREST(rows,p,train_ratio)
    Same as Line 1-13 of algorithm 1
    tress_array = []
While p != = 0
    train ← random(train_ratio * len(rows))
    tree ← BUILDTree(train)
    tree_array.append(tree)
    p = p-1
end While
return tree_array
    procedure CLASSIFY(row,tree_array[],p)
        i ← 0, vote_yes ← 0, vote_no ← 0
        while i do != p do
            tree = tree_array(i)
            node = root(tree)
            if node = leaf then
                if leaf.prediction = "Yes" then
                    vote_yes ← vote_yes + 1
                    if leaf.prediction = "No" then
                        vote_no ← vote_no + 1
                    end if
                    Iterate_tree


                end if
                i = i+1
end While
return vote_yes > vote_no
```

4.3 Prediction model based on KNN

•The k Nearest Neighbor (kNN) algorithm is based on mainly two ideas: the notion of a distance metric and that points that are close to one another are similar.
•Let x be the new data point that we wish to predict a label for. The k Nearest Neighbor algorithm works by finding the k training data points x1,x2,⋯xk closest to x using a Euclidean distance metric. kNN algorithm then performs majority voting to determine the label for the new data point x. In the case of binary classification it is customary to choose k as odd.
•In the situation where we encounter a tie as a result of majority voting there are couple things we can do. First of the all we could randomly choose the winner among the labels that are tied. Secondly, we could weigh the votes by distance and choose the weighted winner and last but not least, we could lower the value of k until we find a unique winner.

Algorithm 3 KNN Algorithm

$X\_train, X\_test, y\_train, y\_test \leftarrow features\_final, asd\_classes, train\_size = 0.60, random\_state = 1$
Print"Training set has (X_trian.shapre[0]) samples."
Print"Testing set has (X_test.shpe[0]) samples."
$knn \leftarrow neighbors.KNeighborsClassifier(n\_neighbors = 10)$
$y\_pred\_class \leftarrow knn.predict(X\_test)$
Print ("True:", $y\_test.values[0:25]$)
Print ("False:", $y\_pred_class[0:25]$)
$confusion \leftarrow metrics.confusion_matrix(y\_test, y\_pred\_class)$
Print (confusion)
$TP \leftarrow confusion[1,1]$
$TN \leftarrow confusion[0,0]$
$FP \leftarrow confusion[0,1]$
$FN \leftarrow confusion[1,0]$
Print ((TP + TN) / float(TP + TN + FP + FN))
$classification\_error = (FP + FN)/float(TP + TN + FP + FN)$
Print (classification_error)
$sensitivity = TP/float(FN + TP)$
Print(*sensitivity*)
print($metrics.recall\_score(y\_test, y\_pred\_class)$)
$specificity = TN/(TN + FP)$
Print(*specificity*)
$false\_positive\_rate = FP/float(TN + FP)$
Print ($false\_positive\_rate$)
$precision = TP/float(TP + FP)$
Print ($metrics.precision_score(y\_test, y\_pred_class)$) =0

4.4 Prediction model based on Naive Bayes Algorithm

•We proceed the study of supervised machine learning algorithms by applying Naive Bayes (NB), which is based around conditional probability (Bayes theorem) and counting. The name "naive" comes from its core assumption of conditional independence i.e. all input features are independent from one and another.
•If the NB conditional independence assumption actually holds, a NB classifier will converge quicker than discriminative models like logistic regression, so one needs less training data. And even if the NB assumption doesn't hold, a NB classifier still often does a great job in practice.
•It's main disadvantage is that it can't learn interactions between features. It only works well with limited number of features. In addition, there is a high bias when there is a small amount of data.

Algorithm 4 Naive Bayes Algorithm

$data = pd.read\_csv("AD.csv")$
$token \leftarrow RegexpTokenizer(r'[a-zA-Z0-9]+')$
$cv \leftarrow CountVectorizer(stop\_words = "english", tokenizer \leftarrow token.tokenize)$
$text_counts_1 \leftarrow cv.fit_transform(data['Class/ASD'])$
$model = MultinomialNB()$
$X\_train\_1, X\_test\_1, Y\_train\_1, Y\_test\_1 = train\_test\_split(text\_counts\_1, data['Class/ASD'], test\_size = 0.80, random\_state = 1)$
$Y\_predict\_1 = model.predict(X\_test\_1)$
$model.fit(X\_train\_1, Y\_train\_1)$
$TN, FP, FN, TP = confusion_matrix(Y_test_1, Y_predict_1).ravel()$
Print(TN, TP, FN, FP)
Print("accuracy : ",(TP + TN) / float(TP + TN + FP + FN))
$classification_error = (FP + FN)/float(TP + TN + FP + FN)$
Print("classification error ",classification$_error$)sensitivity = TP / float(FN + TP)
Print("sensitivity " ,sensitivity)
$specificity = TN/(TN + FP)$
Print(" specificity ",specificity)
$false\_positive\_rate = FP/float(TN + FP)$
Print("false positive rate ",false_positive_rate)

5 Results

5.1 Useful Terms

D.T. = Decision Tree
R.F. = Random Forest
KNN = K- Nearest Neighbor

Table 3. Results of implemented prediction models on AQ-10 DATASET

Performance	KNN	D.T CART	R.F CART	Naive Bayce
Accuracy	0.93	1.0	0.97	0.74
Specificity	0.97	1.0	0.96	1.0
Sensitivity	0.85	1.0	0.96	0.0
Precision	0.99	1.0	0.90	0.25
FPR	0.024	0.0	0.40	0.0

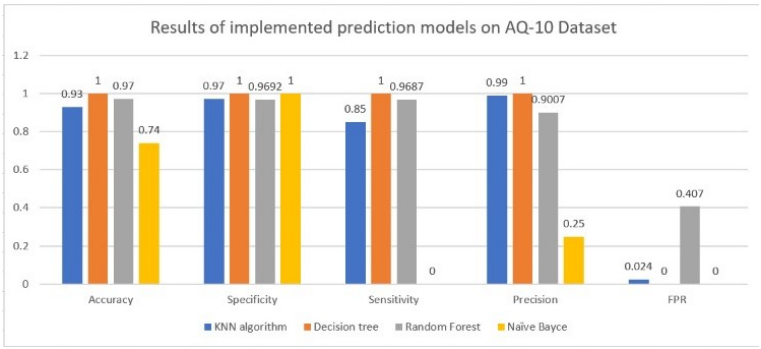

Fig. 16. AQ-10 DATASET

6 Future Work and Conclusion

The proposed Random Forest algorithm will be integrated in a screening android application with the help of Amazon Web Service (AWS). Using AWS, an API will be created to call from the android app. The application will be divided for three different age groups. Different questions will be used for different age groups based on the three AQ-10 screening tool versions. Based on the answer of all the questions the application will show whether or not the user has autism traits. Our future work will focus to collect more data from various sources and to improve the proposed machine learning classifier to enhance its accuracy. A user study will also be conducted to evaluate the usability and user experience (UX) of the mobile application.

6.1 Research Contributions

•This research provides threefold outcome: firstly, a prediction model was developed to predict autism traits. Using the AQ-10 dataset, the proposed model can predict autism with 0.97.
•Decision Tree-Cart algorithm shows accuracy 1.0 which leads to over-fitting and this is controlled in Random Forest- Cart Algorithm.
•Performance of Naive Bayce can degrade if the data contains highly correlated features. This is because the highly correlated features are voted for twice in the model, over inflating their importance.
•The main disadvantage of the KNN algorithm is that it is a lazy learner, i.e. it does not learn anything from the training data and simply uses the training data itself for classification.
•KNN Does not work well with high dimensions: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension.
•This research provides a comparative view among different ML approach in terms of their performance. The results showed that Random Forest-CART showed better performance than the Decision Tree-CART algorithm, KNN, Naive Bayce.

6.2 Limitations and Future Work

The primary limitation of the study is lack of sufficiently large data to train the prediction model. Another limitation is that, the screening application is not designed for age group below 3 years as open source data was not available that age group and for age group less than 18 . Our future work will focus to collect more data from various sources and to improve the proposed machine learning classifier to enhance its accuracy. A user study will also be conducted to evaluate the usability and user experience (UX) of the mobile application where we can make use of different measures like retina scanner,voice record and expression reader.

References

https://www.kaggle.com/faizunnabi/autism-screening