PREDICTION OF AUTISTIC SPECTRUM DISORDER

GROUP:
SRIKRISHNA IYER(15BEC0619)
SHASHANK MOHANKUMAR (15BEC0351)

OBJECTIVE

- a time-efficient and accessible ASD prediction is required to help health professionals and inform individuals whether they should pursue formal clinical diagnosis.

- Hence, Algorithms implemented and compared are:

1. Random forests
2. Logistic regression
3. K nearest neighbours
4. Perceptron model of ANN

- Algorithms compared based on :

1. Mean squared error
2. Accuracy
3. R squared value
4. f score

DATASET

The dataset was formed using responses from an app based questionnaire of people exhibiting autistic traits and individuals who act as control.

Features of dataset are:

- Attribute Type: Categorical, continuous and binary
- Format Type: Non-Matrix
- Dataset contains missing values
- Number of Instances (records in data set): 996
- Number of Attributes (fields within each record): 21
- Relevant Information: Next Slide

DATASET

Table 1: Features and their descriptions

Attribute	Type	Description
Age	Number	years
Gender	String	Male or Female
Ethnicity	String	List of common ethnicities in text format
Born with jaundice	Boolean (yes or no)	Whether the case was born with jaundice
Family member with PDD	Boolean (yes or no)	Whether any immediate family member has a PDD
Who is completing the test	String	Parent, self, caregiver, medical staff, clinician ,etc.
Country of residence	String	List of countries in text format
Used the screening app before	Boolean (yes or no)	Whether the user has used a screening app
Screening Method Type	Integer (0,1,2,3)	The type of screening methods chosen based on age category (0=toddler, 1=child, 2= adolescent, 3= adult)
Question 1 Answer	Binary (0, 1)	I often notice small sounds when others do not
Question 2 Answer	Binary (0, 1)	I usually concentrate more on the whole picture, rather than the small details
Question 3 Answer	Binary (0, 1)	I find it easy to do more than one thing at once
Question 4 Answer	Binary (0, 1)	If there is an interruption, I can switch back to what I was doing very quickly
Question 5 Answer	Binary (0, 1)	I find it easy to read between the lines when someone is talking to me
Question 6 Answer	Binary (0, 1)	I know how to tell if someone listening to me is getting bored
Question 7 Answer	Binary (0, 1)	When I'm reading a story I find it difficult to work out the character's intentions
Question 8 Answer	Binary (0, 1)	I like to collect info about categories of things(eg : types of cars, types of birds, types of train, types of plant etc.)
Question 9 Answer	Binary (0, 1)	I find it easy to work out what someone is thinking or feeling just by looking at their face
Question 10 Answer	Binary (0, 1)	I find it difficult to work out people's intentions
Screening Score	Integer	The final score obtained based on the scoring algorithm of the screening method used. This was computed in an automated manner

Data acquisition

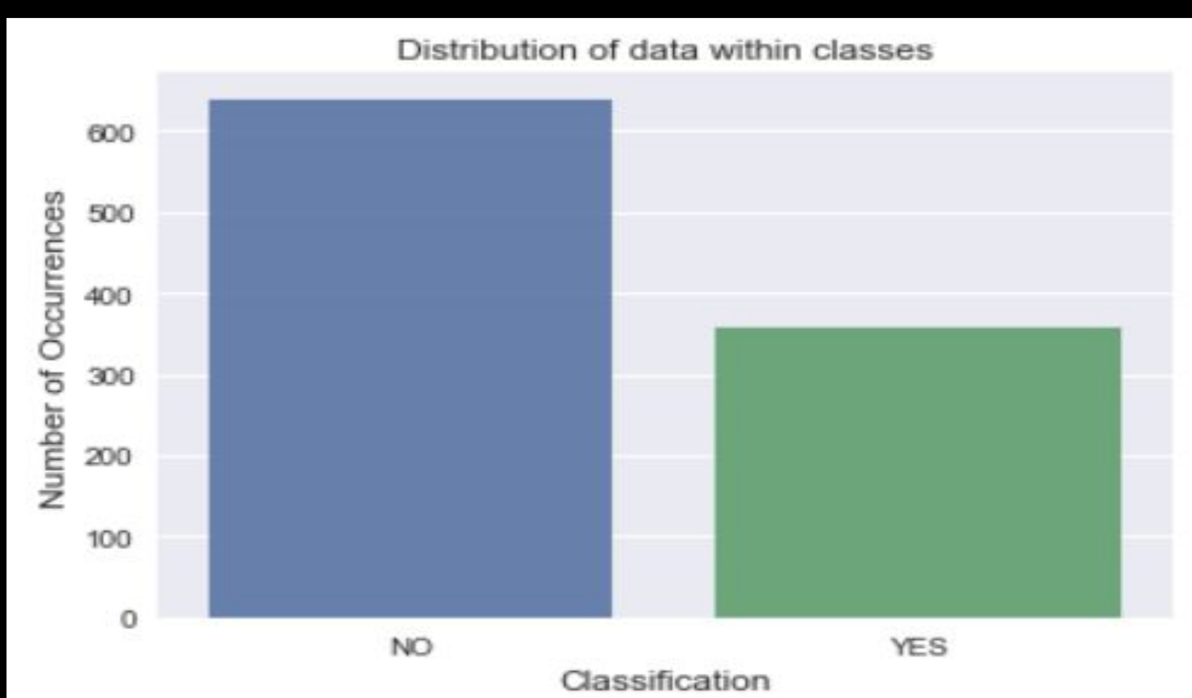
	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	...	gender	ethnicity	jundice	austim	contry_of_res	used_app_before	result	age_desc	relation	Class/ASD
0	1	1	1	1	0	0	1	1	0	0	...	f	White-European	no	no	'United States'	no	6	'18 and more'	Self	YES
1	1	1	0	1	0	0	0	1	0	1	...	m	Latino	no	yes	Brazil	no	5	'18 and more'	Self	YES
2	1	1	0	1	1	0	1	1	1	1	...	m	Latino	yes	yes	Spain	no	8	'18 and more'	Parent	YES
3	1	1	0	1	0	0	1	1	0	1	...	f	White-European	no	yes	'United States'	no	6	'18 and more'	Self	NO
4	1	0	0	0	0	0	0	1	0	0	...	f	?	no	no	Egypt	no	2	'18 and more'	?	YES
5	1	1	1	1	1	0	1	1	1	1	...	m	Others	yes	no	'United States'	no	9	'18 and more'	Self	NO
6	0	1	0	0	0	0	0	1	0	0	...	f	Black	no	no	'United States'	no	2	'18 and more'	Self	YES

The dataset contained missing values and non-categorical values that needed rectifying.

Data preparation

- Handling missing data using forward fill
- Encoding training data into categorical labels

	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	age_desc	relation	Class/ASD
0	1	1	1	1	0	0	1	1	0	0	26	0	9	0	0	13	0	0	4	1
1	1	1	0	1	0	0	0	1	0	1	24	1	5	0	1	30	0	0	4	1
2	1	1	0	1	1	0	1	1	1	1	27	1	5	1	1	76	0	0	2	1
3	1	1	0	1	0	0	1	1	0	1	35	0	9	0	1	13	0	0	4	0
4	1	0	0	0	0	0	0	1	0	0	40	0	9	0	0	38	0	0	4	1
5	1	1	1	1	1	0	1	1	1	1	36	1	6	1	0	13	0	0	4	0
6	0	1	0	0	0	0	0	1	0	0	17	0	3	0	0	13	0	0	4	1
7	1	1	1	1	0	0	0	0	1	0	64	1	9	0	0	4	0	0	2	0
8	1	1	0	0	1	0	0	1	1	1	29	1	9	0	0	13	0	0	4	1
9	1	1	1	1	0	1	1	1	1	0	17	1	2	1	1	24	0	0	0	1



Perceptron implementation

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 32)                608
_________________________________________________________________
dense_2 (Dense)              (None, 64)                2112
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 65
=================================================================
Total params: 2,785
Trainable params: 2,785
Non-trainable params: 0
_________________________________________________________________
```

Model summary of perceptron implementation using sequential model of keras

1. The training model consists of 1 input layer whose input dimension is 18, a hidden layer and an output layer.
2. method of weights initialization: uniform normal distribution
3. input activation function: relu function (Rectified linear unit) which is defined as:
4. Output activation function: sigmoid function
5. Error function: mean squared error
6. optimizer: gradient descent to minimize weights
7. validation set split: 9% of training data
8. number of epochs=300
9. Batch size=100

$$f(x)= \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x => 0 \end{cases}$$

```
906/906 [==============================] - 0s 15us/step - loss: 0.0404 - acc: 0.9558 - val_loss: 0.0722 - val_acc: 0.9222
Epoch 292/300
906/906 [==============================] - 0s 14us/step - loss: 0.0414 - acc: 0.9558 - val_loss: 0.0845 - val_acc: 0.8889
Epoch 293/300
906/906 [==============================] - 0s 15us/step - loss: 0.0356 - acc: 0.9636 - val_loss: 0.0766 - val_acc: 0.9000
Epoch 294/300
906/906 [==============================] - 0s 14us/step - loss: 0.0355 - acc: 0.9647 - val_loss: 0.0703 - val_acc: 0.9111
Epoch 295/300
906/906 [==============================] - 0s 17us/step - loss: 0.0354 - acc: 0.9658 - val_loss: 0.0774 - val_acc: 0.9000
Epoch 296/300
906/906 [==============================] - 0s 17us/step - loss: 0.0433 - acc: 0.9514 - val_loss: 0.0784 - val_acc: 0.9000
Epoch 297/300
906/906 [==============================] - 0s 13us/step - loss: 0.0338 - acc: 0.9680 - val_loss: 0.0710 - val_acc: 0.9333
Epoch 298/300
906/906 [==============================] - 0s 15us/step - loss: 0.0370 - acc: 0.9614 - val_loss: 0.0801 - val_acc: 0.9000
Epoch 299/300
906/906 [==============================] - 0s 18us/step - loss: 0.0350 - acc: 0.9647 - val_loss: 0.0697 - val_acc: 0.9222
Epoch 300/300
906/906 [==============================] - 0s 17us/step - loss: 0.0386 - acc: 0.9614 - val_loss: 0.0749 - val_acc: 0.9111
```

Training neural network

performance Metrics

The following metrics were used to compare and validate implemented algorithms during ***training*** and ***cross validation***:

1. R square value (coefficient of determination):

   R-squared is a statistical measure of how close the data are to the fitted regression line

   R-squared is always between 0 and 1

- 0 indicates that the model explains none of the variability of the response data around its mean.
- 1 indicates that the model explains all the variability of the response data around its mean.

   it can be negative (because the model can be arbitrarily worse).

2. f1 score: The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

Calculated as:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

3. MSE(mean squared error): It is calculated as:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

where Y is target output and Y’ is predicted output.

4. Accuracy: calculated as:

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

where TP: true positives , TN: true negatives, FP: false positives , FN: false negatives.

Simply calculated as: *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

Results

	Training				Validation			
	R squared	MSE	f1 score	Accuracy	R squared	MSE	f1 score	Accuracy
perceptron ANN	0.813	0.043	0.942	95.81	0.585	0.081	0.814	91.11
Random forests	1	0.03	0.976	90	0.665	0.075	0.872	91.5
logistic regression	0.751	0.057	0.915	92.7	0.585	0.088	0.857	91
K nearest neighbour	1	0	0.957	96.88	-0.424	0.31	0.575	69

Interpretation of results

1. Accuracy:

- From the results we can conclude that the K nearest neighbour had the highest training accuracy. (where K=1)
- Based on cross validation test data, random forest had the highest accuracy followed very closely by perceptron.

2. f1 score:

- Again , the random forests outperformed all other algorithms based on the f score. This shows that the outputs predicted by random forests were more correctly recalled .

3. MSE

- The 1 nearest neighbour minimized its MSE the most during training of input dataset.
- However, while validation , random forests had the least MSE.

4. R-squared value

- Again, the 1-nearest neighbour r squared value was negative during validation because it underfit the decision boundary by a huge extent , hence performing the worst amongst other algorithms.
- While , the random forests had the highest r square values during training and cross-validation.

Conclusion

Based on the overall performance:

- The **random forest outperformed** all other algorithms based on MSE, R-square, f-score and accuracy during cross-validation.
- Though, K-NN trained the best , it was the worst performing algorithm during cross validation.
- Though the training accuracy of perceptron model was higher than random forests, it wasn’t able to minimize MSE and recall the predicted outputs(f1-score) as well as random forests.

THANK YOU