Machine learning approaches to the classification problem for autism spectrum disorder

Shrishti Bhasin

May 12, 2023

1 Definition

1.1 Project Overview

Autistic Spectrum Disorder (ASD) is the name for a group of developmental disorders impacting the nervous system. ASD symptoms range from mild to severe: mainly language impairment, challenges in social interaction, and repetitive behaviors. Many other possible symptoms include anxiety, mood disorders and Attention-Deficit/Hyperactivity Disorder (ADHD).

ASD has a significant economic impact in the healthcare domain, both due to the increase in the number of ASD cases, and because of the time and costs involved in diagnosing a patient. Early detection of ASD can help both patients and the healthcare sector by prescribing pa- tients the therapy and/or medication they need and thereby reducing the long term costs associated with delayed diagnosis. Thus, health care professionals across the globe have an urgent need for easy, time-efficient, robust and accessible ASD screening methods that can accurately predict whether a patient with certain measured characteristics has ASD and inform individuals whether they should pursue formal clinical diagnosis.

However, challenges remain. Pursuing such research necessitates working with datasets that record information related to behavioral traits and other factors such as gender, age, ethnicity, etc. 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. At present, very limited autism datasets associated with clinical or screening are available and most of them are genetic in nature. These data are extremely sensitive and hard to collect for social and persnal reasons and the regulations around them.

1.2 Origins of ASD Data Set

I was able to find open source data available at **UCI Machine Learning Repository**. The data was made available to the public recently on December 24th, 2017. The data set, which I will be referring to as the ASD data set from here on out, came with a .csv file that contains 704 instances that are described by 21 attributes, a mix of numerical and categorical variables. A short description of ASD dataset can be found on this **page**. This data set was denoted by Prof.Fadi Fayez Thabtah, Department of Digital Technology, MIT, Auckland, New Zealand, fadi.fayez@manukau.ac.nz. Further details of the raw data follows in Section 2.

1.3 Problem Statement

With the available ASD data on individuals my goal is to make predictions regarding new patients and classify them into one of two categories: “patient has ASD” or “patient does not have ASD ”. In other words, we are working on a binary classification problem with the ultimate goal of being able to classify new instances, i.e. when we have a new adult patient with certain characteristics we would like to able to predict whether or not that individual has high probability of having ASD.

We will use *supervised machine learning* to refer to creating and using models that are learned from data, i.e., there is a set of data labeled with the correct answer for the model to learn from. I will also apply a *feature selection* algorithm to figure out which of the 20 variables are most important in determining whether an individual has ASD or not.

This work aims to explore several competing supervised machine learning classification techniques namely:

- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- k-Nearest Neighbors (kNN)
- Naive Bayes
- Logistic Regression
- Linear Discriminant Analysis (LDA)
- Multi-Layer Perceptron (MLP)

It also amis to implement the one that proves most effective in terms of correct classification or a combination of classifiers (*Ensemble Learning*) like Random Forests to arrive at a deci- sion i.e., to identify which patient has ASD. All algorithms have been coded using Python and its various packages (*Scikit Learn* and *Tensorflow with Keras*).

No one model is perfect or universally applicable and so the question that naturally arises is how to determine which machine learning algorithm to choose for a particular classification problem such as predicting whether a person has ASD. We will discuss performance statis- tics, using standard metrics that are defined below in Subsection 1.4, and the strengths and weaknesses of each model will be discussed in Subsection 2.2. To summarize, our goal with these implementations is to construct a model that accurately predicts whether an individual with certain characteristics has ASD or not evaluating the 8 methods listed above.

In addition to being able to predict whether an individual with the given characteristics will have ASD or not, we would also like to be able to identify the most influential autistism traits. The hope is to identify individuals who have a high chance to be diagnosed with ASD and provide them with relevant treatment, therapy and counseling in a time sensitive fashion.

1.4 Metrics

In order to choose the appropriate model that avoids *Underfitting* or *Overfitting* the data we will analyze the *Bias-Variance Trade-Off*, *Model Complexity Graph, Learning Curves* and *Receiver Operator Characteristic Curves (ROC)*. To measure the effectiveness of each classi- fication model we will study the *accuracy score* along with the *precision, recall*, *F-Beta Score*

and *confusion matrix*.

Definition 1.1. Model Complexity Graph
Our goal is to always find the model that will generalize well to unseen data. A model-complexity graph plots the training error and cross validation error as the model's com- plexity varies, i.e., the x-axis represents the complexity of the model (such as degree of the polynomial for regression or depth of a decision tree) and the y-axis measures the error in training and cross validation. Note that the size of the data set remains constant even while model complexity varies.

Figure (1) below shows a typical model complexity graph. On the left we have a model that underfits and we see high training and cross-validation errors, while on the right we have a model that overfits and gives us low training error but high cross validation error. The model in the middle is just right, with relatively low training and testing errors, and this is the model we should select. The best predictive and fitted model (neither underfitting nor overfitting) would be where the cross validation error achieves its global minimum.

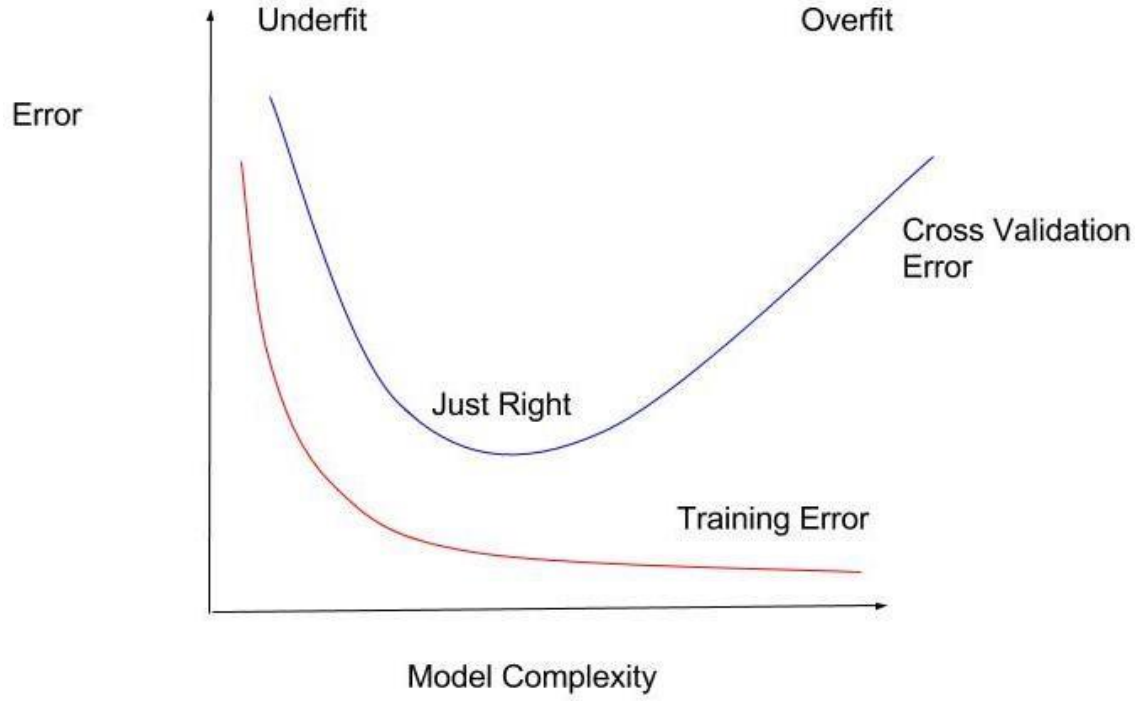


Figure 1: Model Complexity Graph

Definition 1.2. Learning Curves
Learning curves provide a way for us to distinguish between underfitting, overfitting, or the model that is just right. Learning curves are a plot of both the training error and cross val-idation error versus the size of the training set. In other words, a learning curve in machine learning is a graph that compares the errors of a model on training and cross validation data over a varying number of training instances. By separating training and cross validation sets and graphing errors of each, learning curves help us understand how well the model will potentially generalize to the unseen testing data.

Typically, we observe that the training error increases with the size of the training set, since we have more points to fit the model to. A learning curve allows us to verify when a model has learned as much as it can about the data. When this occurs, both the training and cross validation errors reach a plateau and there is a consistent gap between the two error rates. In the case of underfitting or a high-bias model the two curves converge to a high point. In the perfect model the curves converge to a low point. In the case of overfitting or a high-variance model, the curves do not converge; the training curve stays low and the cross validation error stays high, since models that overfit tend to memorize the training data. (See Figure 2).

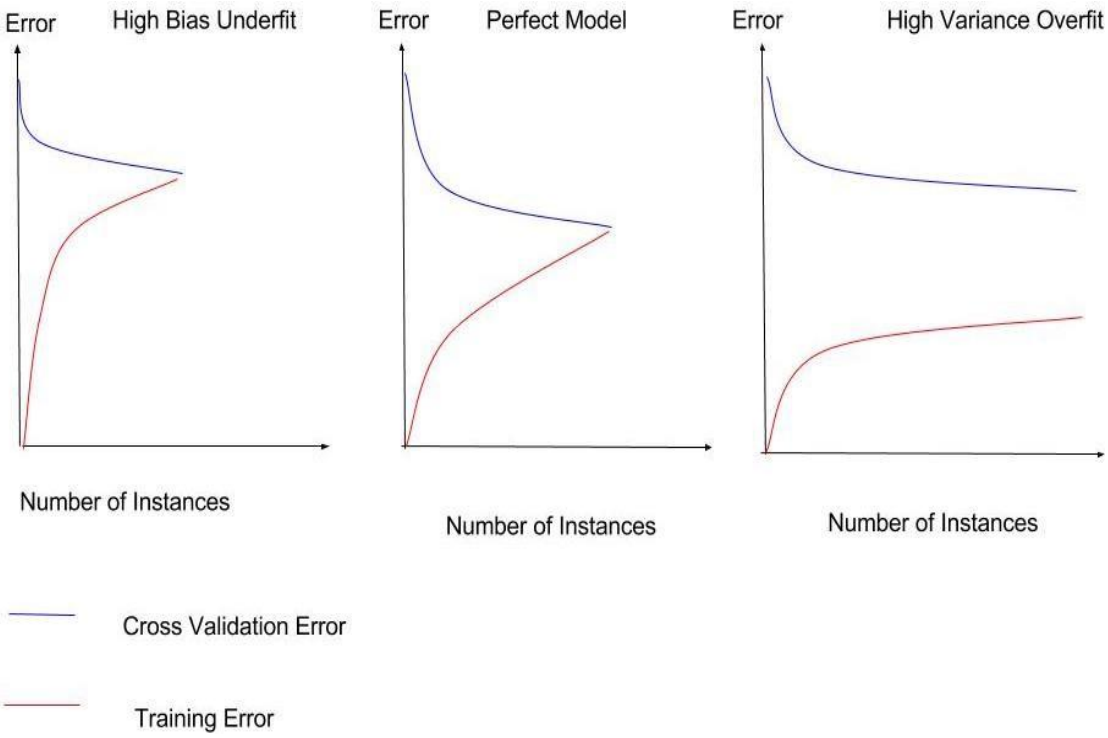


Figure 2: Learning Curves

Definition 1.3. Receiver Operating Characteristic (ROC) Curves
The receiver operating characteristic curve is a commonly used summary for assessing the diagnostic ability of a binary classifier over all possible thresholds. It is a plot of the true-positive rate (also known as *sensitivity* or *recall*) versus the false-positive rate (also known as *specificity*). The area under the ROC curve gives an idea about the overall performance of a classifier, summarized. The bigger the area under its curve, the better the overall performance of the binary classifier.

Given a set of labeled data and a predictive model, every data point will lie in one of the four categories:

- *True positive*: The adult individual DID have ASD and we correctly predicted that the individual would have ASD.
- *True negative*: The individual did NOT have ASD and we correctly predicted that the individual would NOT have ASD.

- *False positive (Type 1 Error)*: The individual did NOT have ASD, but we incorrectly predicted that the individual would have ASD.
- *False negative (Type 2 Error)*: The individual DID have ASD, but we incorrectly predicted that the individual would NOT have ASD.

These counts can also be represented in a *confusion matrix* (see Table 1) that allows us to visualize the performance of a supervised machine learning algorithm:

	individual with ASD	individual with no ASD
predicted: ASD= 'YES'	True Positive	False Positive
predicted: ASD='NO'	False Negative	True Negative

Table 1: Confusion Matrix

Definition 1.4. Accuracy
Accuracy measures how often the classifier makes the correct prediction. In other words, it is the ratio of the number of correct predictions to the total number of predictions (the number of test data points). Accuracy is defined as the fraction of correct predictions.

$$\text{accuracy} = \frac{\text{true positive} + \text{truenegative}}{\text{true positive} + \text{false positive} + \text{false negative} + \text{true negative}} \quad (1)$$

Definition 1.5. Precision
Precision measures how accurate our positive predictions were i.e., out of all the points predicted to be positive how many of them were actually positive.

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} \quad (2)$$

Definition 1.6. Recall
Recall measures what fraction of the positives our model identified, i.e., out of the points that are labelled positive, how many of them were correctly predicted as positive. Another way to think about this is what fraction of the positive points were my model able to catch?

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}} \quad (3)$$

For classification problems that are skewed in their classification distributions like ours where we have a total of 609 records (after data preprocessing) with 180 individuals diagonosed with ASD and 429 individuals not diagonosed with ASD, accuracy by itself is not a very good metric. Thus, in this case precision and recall come in very handy. These two metrics can be combined to get the F1 score, which is weighted average(harmonic mean) of the precision and recall scores. This score can range from 0 to 1, with 1 being the best possible F1 score(we take the harmonic mean as we are dealing with ratios).

Definition 1.7. $F1 - Score$
The $F1 - Score$ is the harmonic mean of precision and recall and thus must lie between them. The $F1 - Score$ is closer to the smaller of precision and recall than the higher number. As a result, if one of the precision or recall value is low the $F1 - Score$ raises a flag.

$$F1\_Score = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

Definition 1.8. F_β *Score*

The F_β score is the weighted harmonic mean of precision and recall, reaching its optimal value at 1 and its worst value at 0. The β parameter determines the weight of precision in the combined score, i.e, $\beta < 1$ lends more weight to precision, and $\beta > 1$ favors recall.

Further, $\beta = 0$ considers only precision and $\beta = \infty$ considers only recall.

2 Analysis

2.1 Data Exploration

Our **data set** involves ten behavioral features ("AQ-10-Adult") (binary data) and ten indi- vidual characteristics such as "Gender", "Ethnicity", "Age", etc (categorical data) and one numerical data ("result"). Table 2) below lists all variables involved in the ASD data set.

Variable Name	Description
Age	age in years
Gender	male or female
Ethnicity	list of common ethnicities in text format
Born with Jaundice	whether case was born with jaundice
Family member with PDD	whether any immediate family member has a PDD
Who is completing the test	parent, self, caregiver, medical staff, clinician, etc
Country of Residence	list of countries in text format
Used the screening app before	whether the user has used screening app
Screening Method Type	type of screening method chosen based on age category
Question 1 Answer	the answer code of the question based on the screening method used
Question 2 Answer	the answer code of the question based on the screening method used
Question 3 Answer	the answer code of the question based on the screening method used
Question 4 Answer	the answer code of the question based on the screening method used
Question 5 Answer	the answer code of the question based on the screening method used
Question 6 Answer	the answer code of the question based on the screening method used
Question 7 Answer	the answer code of the question based on the screening method used
Question 8 Answer	the answer code of the question based on the screening method used
Question 9 Answer	the answer code of the question based on the screening method used
Question 10 Answer	the answer code of the question based on the screening method used
Screening Score	final score obtained based on scoring algorithm of screening method used

Table 2: List of Attributes in ASD dataset.

Our raw data set contains 704 instances with 189 individuals diagnosed with ASD. Thus, the percentage of individuals diagnosed with ASD is 26.85%. Here is a sample of what the ASD data looks like:

```
# Import libraries necessary for this project
import numpy as np
import pandas as pd
from time import time
from IPython.display import display # Allows the use of display() for DataFrames

# Import supplementary visualization code visuals.py
import visuals as vs

# Pretty display for notebooks
%matplotlib inline


data = pd.read_csv('ASD.csv')
display(data.head(n=5))
```

core	A7_Score	A8_Score	A9_Score	...	gender	ethnicity	jundice	austim	contry_of_res	used_app_before	result	age_desc	relation	Class/ASD
	1	1	0	...	f	White-European	no	no	United States	no	6	18 and more	Self	NO
	0	1	0	...	m	Latino	no	yes	Brazil	no	5	18 and more	Self	NO
	1	1	1	...	m	Latino	yes	yes	Spain	no	8	18 and more	Parent	YES
	1	1	0	...	f	White-European	no	yes	United States	no	6	18 and more	Self	NO
	0	1	0	...	f	?	no	no	Egypt	no	2	18 and more	?	NO

Figure 3: A quick look at the dataset.

Below in Figure 4 you will find the summary statistics for each variable:

```
asd_data.describe()
```

	id	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	A10_Score	
count	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	704.000000	702.
mean	352.500000	0.721591	0.453125	0.457386	0.495739	0.498580	0.284091	0.417614	0.649148	0.323864	0.573864	29.6
std	203.371581	0.448535	0.498152	0.498535	0.500337	0.500353	0.451301	0.493516	0.477576	0.468281	0.494866	16.5
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	17.0
25%	176.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	21.0
50%	352.500000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	1.000000	27.0
75%	528.250000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	35.0
max	704.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	383.

Figure 4: A quick view at the summary statistics on ASD data.

Exploratory Visualization

Before proceeding to apply any algorithm, we take a moment to visualize the ASD data set using the *Seaborn* module of **Python**.

We first generate side-by-side box plots, which present various distribution of the feature 'result' with respect to 'gender' and 'relation'. In Figure 5, 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.

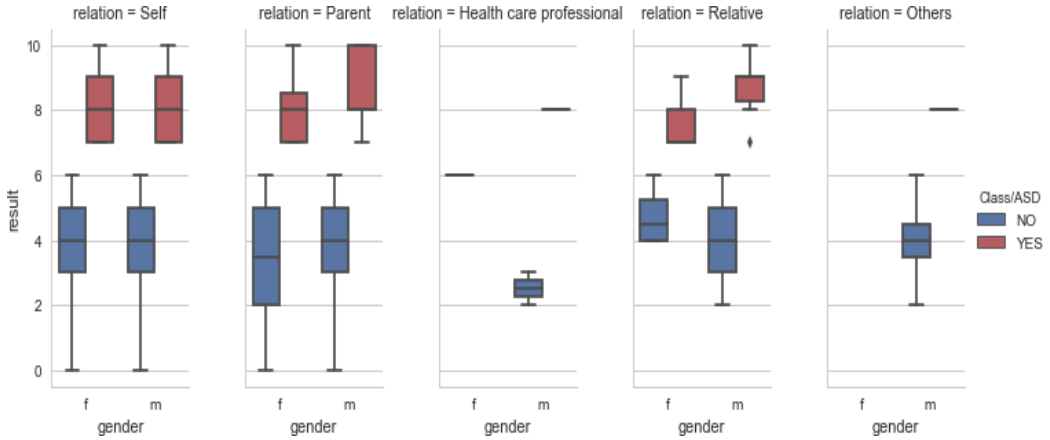


Figure 5: Factorplot: kind = 'boxplot'

Next we present Figure 6, a special form of Factorplot where we display the connection between several attributes from our data set and how they are related with our target class. 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.

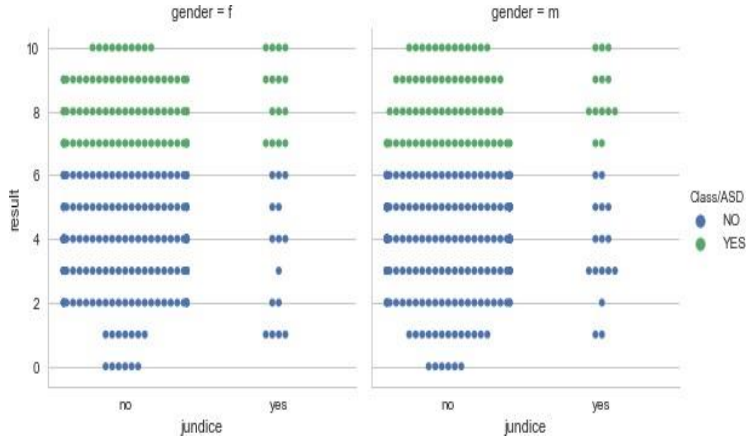
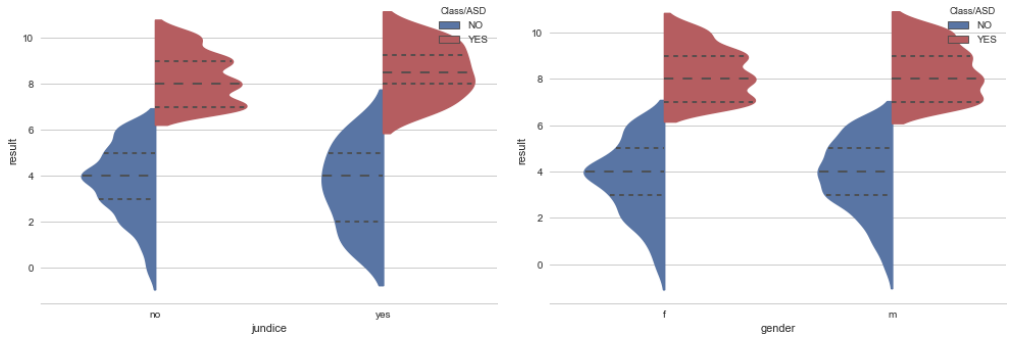


Figure 6: Factorplot: kind = 'swamp'

We move on to a series of violin plots to compare how different features contribute to the likelihood of autism. Figure 7a and Figure 7b show a similar relationship between 'result' and 'gender' versus 'result' and 'jaundice'. In both cases, an individual with a higher 'result' score is more likely to have autism, independent of the other feature. The variables 'jaundice' or 'gender' do not seem to have a lot of influence in deciding the ASD class.

Lastly we present another variation of a violin plot in Figure 8, where we look at how the distribution of autism varies by relation (Self, Parent, ...), subdivided by whether the patient was born with jaundice ('jundice') and the patient's gender.



(a) (b)

Figure 7: (a) A violine plot showing how the ASD classes is related with the attributes 'result' & 'jundice'. (b) A violine plot showing how the ASD classes is related with the attributes 'result' & 'gender'.

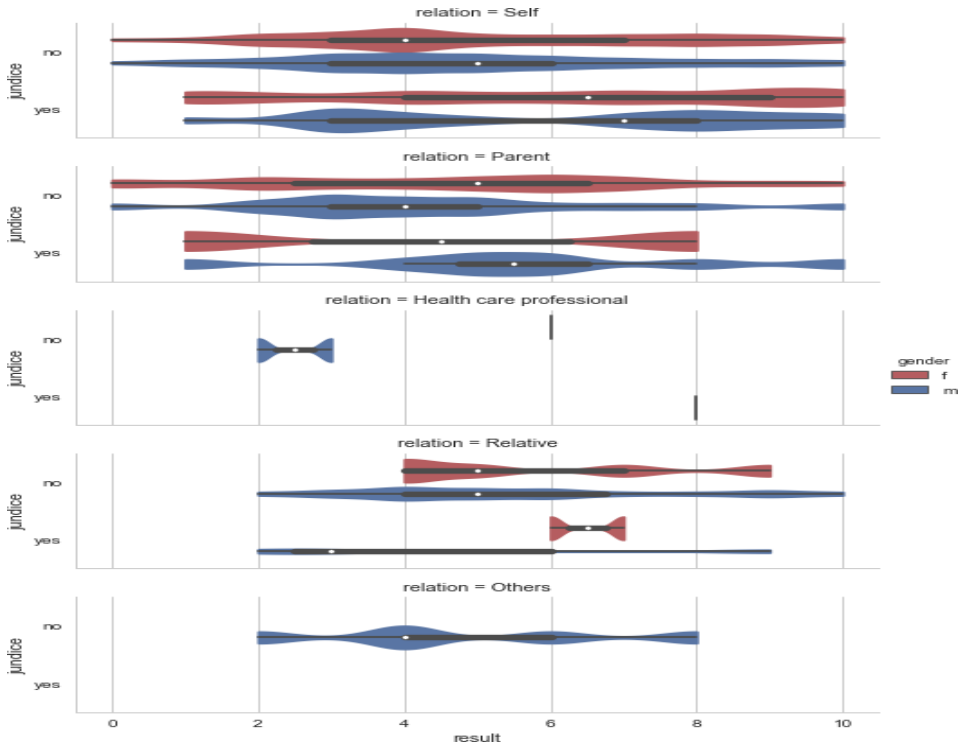


Figure 8: A violine plot depicting multi-feature relationship.

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.

2.2 Algorithms and Techniques

Below we discuss the algorithms we have applied to our preprossed ASD data:

I. Decision Trees.

We will begin with creat ng a Decision Tree Classifier, also known as ID3 Algorithm and fit our training data. A Decision Tree uses a tree structure to represent a number of possible decision paths and an outcome for each path. They can also perform regression tasks and

they form the fundamental components of Random Forests which will be applied to this data set in the next section.

Decision Tree models are easy to use, run quickly, are able to handle both categorical and numerical data, and graphically allow you to interpret the data. Further, we don't have to worry about whether the data is linearly separable or not.

On the other hand, Decision Trees are highly prone to overfitting but one solution to that is pruning the branches so that not too many features are included and the use of ensemble methods like random forests. Another weakness of Decision Trees is they don't support online learning, so you have to rebuild your tree when new examples come in.

A Decision Tree model is a good candidate for this problem as they are particularly adept at binary classification, however it may run into problems due to the high number of features so care will have to be taken with regards to feature selection. Due to these advanatges and the ease of interpretation of the results, we will use the Decision Tree Classifier as the benchmark model.

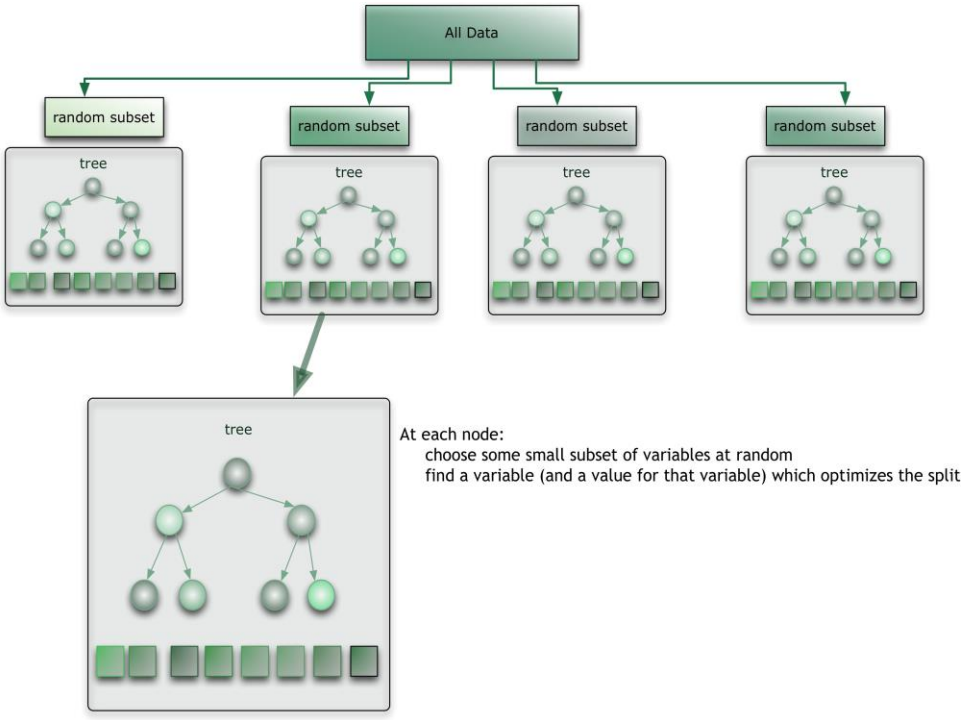


Figure 9: Random Forest Diagram

II. Random Forests.

One way of avoiding overfitting that Decision Trees are prone to, is to apply a technique called *Random Forests*, in which we build multiple decision trees and let them vote on how to classify inputs. Random forests (or random decision forests) are an ensemble learning method for classification, regression and other tasks, that operate by constructing a mul- titude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random deci- sion forests correct for decision trees' habit of overfitting to their training set.

A depiction of the algorithm is presented in the Figure 9 .

III. Support Vector Machines (SVM):

Next we move on to the Support Vector Machine algorithm (SVM) which is, by far, my favorite machine learning algorithm. Support Vector Machine is a supervised machine learn- ing algorithm that is commonly used in classification problems. It is based on the idea of finding the hyperplane that 'best' splits a given data set into two classes. The algorithm gets its name from the *support vectors* (the data points closest to the hyperplane), which are points of a data set that if removed would alter the position of the separating hyperplane. (See Figure 10).

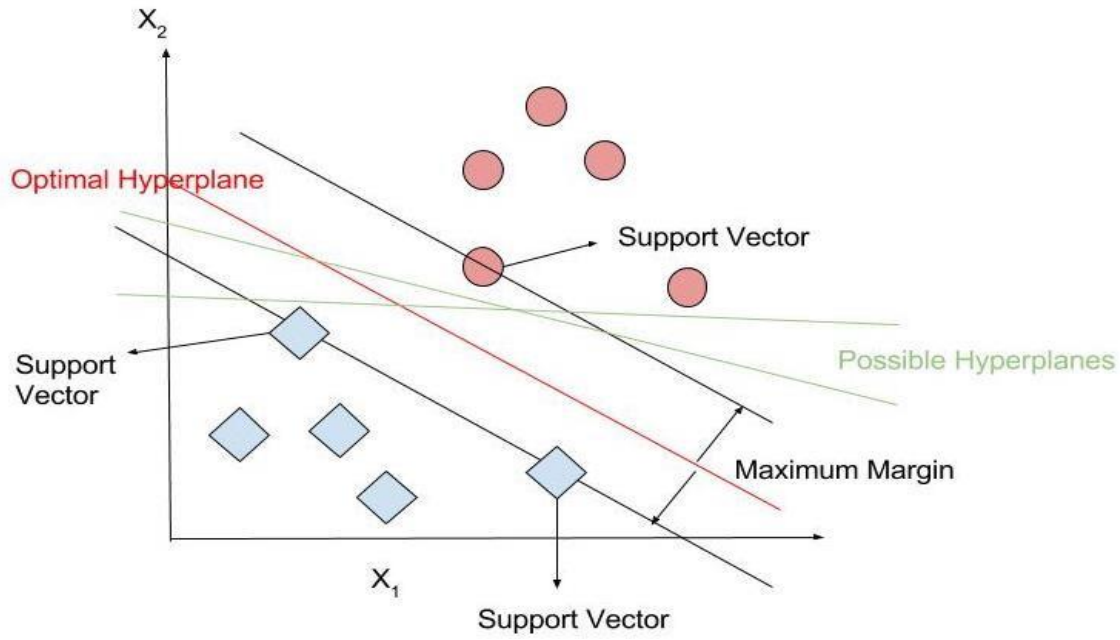


Figure 10: Support Vector Machine Diagram

The distance between the hyperplane and the nearest training data point from either set is known as the *margin*. Mathematically, the SVM algorithm is designed to find the hyperplane that provides the largest minimum distance to the training instances. In other words, the optimal separating hyperplane maximizes the margin of the training data.

IV. k-Nearest Neighbors (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 Neigh- bor algorithm works by finding the k training data points x_1, x_2, x_k 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.

V. Naive Bayes:

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 fea- tures 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.

VI. Logistic Regression:

The goal of logistic regression is to find the best fitting model to describe the relationship be- tween the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of presence of the characteristic of interest:

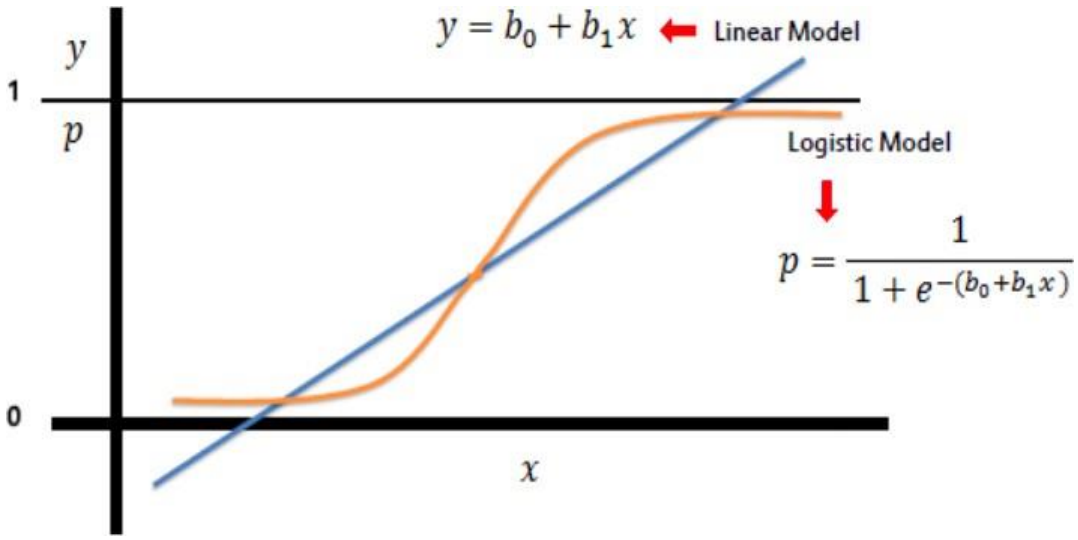


Figure 11: Logistic Regression Model

Logistic regression can be called as binary classification problems. A key point to note here is that Y can have 2 classes only and not more than that. If response variable has more than 2 classes, it would become a multi class classification and you can no longer use the vanilla logistic regression for that. Yet, Logistic regression is a classic predictive modelling technique and still remains a popular choice for modelling binary categoricalvariables.

Another advantage of logistic regression is that it computes a prediction probability score of an event. More on that when you actually start building the models. Logistic regression achieves this by taking the log odds of the event $ln(\frac{P}{1-P})$ where, P is the probability of event. So P always lies between 0 and 1.

$$z_i = \ln \frac{P_i}{1 - P_i} = \alpha + \beta_1 x_{1\,1} + \beta_2 x_{2\,2} + \cdots \beta_n x_{n\,n},$$

Taking exponent on both sides of the equation gives:

$$P_i = E(y = 1/x_i) = \frac{e^z}{1 + e^z} = \frac{e^{\alpha+\beta_i x_i}}{1 + e^{\alpha+\beta_i x_i}}$$

VII. Linear Discriminant Analysis (LDA).

A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes theorem. The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix. The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions.

VIII. Multi-Layer Perception(MLP).

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable. Multilayer perceptrons are sometimes colloquially referred to as 'vanilla' neural networks, especially when they have a single hidden layer.

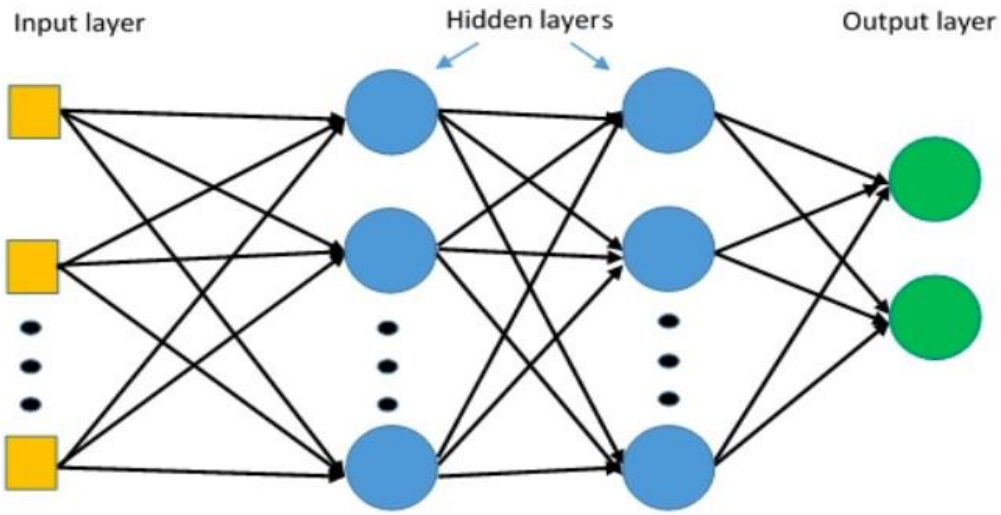


Figure 12: MLP Architecture

2.3 Benchmark Model

Our **data set** was released on the **UCI repository** on the 24th of December, 2017. Very little work has been done with this particular data. We are unaware of any rigorous data classification problems related to this ASD data using a machine learning approach but in

[8] there is some mention of a Decision Tree algorithm can be applied for this classification

problem although no numerical results or measurable metric were presented in the article.

The reasons for my choice to use the Decision Tree Classifier as the Benchmark Model for this problem has been discussed in details in Section 2.2.

3 Methodology

3.1 Data Preprocessing

Unfortunately, this data set does have a lot of invalid or missing entries that are represented with question marks. Thus, we must preprocess our data so it is ready to be used as input for machine learning algorithms. Here, we begin, by replacing entries with the symbol "?" and convert them into 'NAN' (not a number). We come to find that there are quite a few missing values in the data set, but the missing values are random (verified through Python code, see Figure 13). In other words, it seems appropriate to simply drop every row that contains the missing data and we will not run the risk of biasing our data. This brings the number of instances in our data set to 609 from 704.

Step 1: Clean Datasets

```
In [34]: asd_data.loc[(asd_data['age'].isnull()) |(asd_data['gender'].isnull()) |(asd_data['ethnicity'].isnull())
         |(asd_data['jundice'].isnull())|(asd_data['austim'].isnull()) |(asd_data['contry_of_res'].isnull())
                  |(asd_data['used_app_before'].isnull())|(asd_data['result'].isnull())|(asd_data['age_desc'].isnull())
                  |(asd_data['relation'].isnull())]
```

	id	A1_Score	A2_Score	A3_Score	A4_Score	A5_Score	A6_Score	A7_Score	A8_Score	A9_Score	...	gender	ethnicity	jundice	austim	contry_of_res
4	5	1	0	0	0	0	0	0	1	0	...	f	NaN	no	no	Egypt
12	13	0	1	1	1	1	1	0	0	1	...	f	NaN	no	no	Bahamas
13	14	1	0	0	0	0	0	1	1	0	...	m	NaN	no	no	Austria
14	15	1	0	0	0	0	0	1	1	0	...	f	NaN	no	no	Argentina
19	20	0	0	0	0	0	0	1	1	0	...	m	NaN	yes	no	United Arab Emirates
20	21	0	1	1	1	0	0	0	0	0	...	m	NaN	no	no	United Arab Emirates

Since the missing data seems randomly distributed, I go ahead and drop rows with missing data.

Figure 13: A view at the locations of the missing values in ASD data

Next, we split the data into features and and target label and normalize the numerical vari- ables 'age' and 'result', using the MinMax Scaler feature in Python. For the categorical variables we use the one-hot encoding scheme. This results in 94 total features after one-hot encoding. Additionally, we need to convert the non-numeric target variable 'Class/ASD' to numerical values for the learning algorithm to work. Since there are only two possible categories for this label ("yes" or "no"), we can avoid using one-hot encoding and simply encode these two categories as 0 and 1, respectively.

During the process of cleaning data, we had to drop several rows in order to make the data set suitable for using different algorithms from *SciKit Learn*. We did not feel comfortable dropping these instances, since droping data may lead to biases and insightful information could be lost. Specifically for this data when we know that autism related information

is very difficult to get for various social and administrative regulations, we usually try to replace missing entries with the 'median' value of that corresponding column. But that strategy did not pan out here as many of the categorical columns had missing data and it is difficult (though not impossible using 'imputation' method) to find a suitable replacement for a missing value. As a result, we have to drop them which was almost 14% of the original data set. It is frustrating not to be able to use all the precious information available.

3.2 Implementation

Assuming that the available data for analytics fairly represents the real world process that we wish to model and the process is expected to remain relatively stable, then the data we currently have should be a reasonable representation of the data we expect to see in the future. As a result, withholding some of the current data for testing is a fair and justified way to provide an honest assessment of our model. Thus, we split the given data into two parts. 80% of the data will be used to *train* the model and this data will be referred to as the *training data set* and 20% of the data will be reserved for *testing* the accuracy and effectiveness of the model on data that the model has never seen before and will be referred as the *testing data set*. Thus, our training set has 487 samples, and the testing set has 122 samples. Below is a flowchart that represents the whittling process of the raw data:

The random partitioning of data into testing and training data also helps us determine whether our model is *underfitting* (too simple, high bias, low variance) or *overfitting* (too complicated, high variance, low bias). A model that has high training and testing error is a model that underfits. This means our model is too simplistic. A model that has low training error, but high testing error is one that overfits. This means that our model is memorizing the data rather than trying to understand the intrinsic trends or patterns in the data. In other words, it's important to test our model and see how it generalizes to unseen data by applying the model with testing data which was not a part of the model creation.

Definition 3.1. Cross Validation
Cross-validation is a model validation technique for assessing how the results of a machine learning algorithm will generalize to an unseen data set. The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the cross validation dataset), in order to limit problems like under fitting or overfitting, and give an insight on how the model will generalize to an independent unknown dataset.

Definition 3.2. k-Fold Cross Validation
One of the disadvantages in performing cross-validation is that we lose quite a bit of data that could have been used to train the model and thus possibly arrive at more correct pre- dictions. In a traditional train-test split, the error metric can have high variance, i.e., the error may depend heavily on which data points end up in the training set and which end up in the test set, and thus the evaluation may be significantly different depending on how the division is made. To overcome these difficulties, another popular technique for model assessment with the same flavor as cross-validation but slight variation is called *k-Fold Cross Validation* is used.

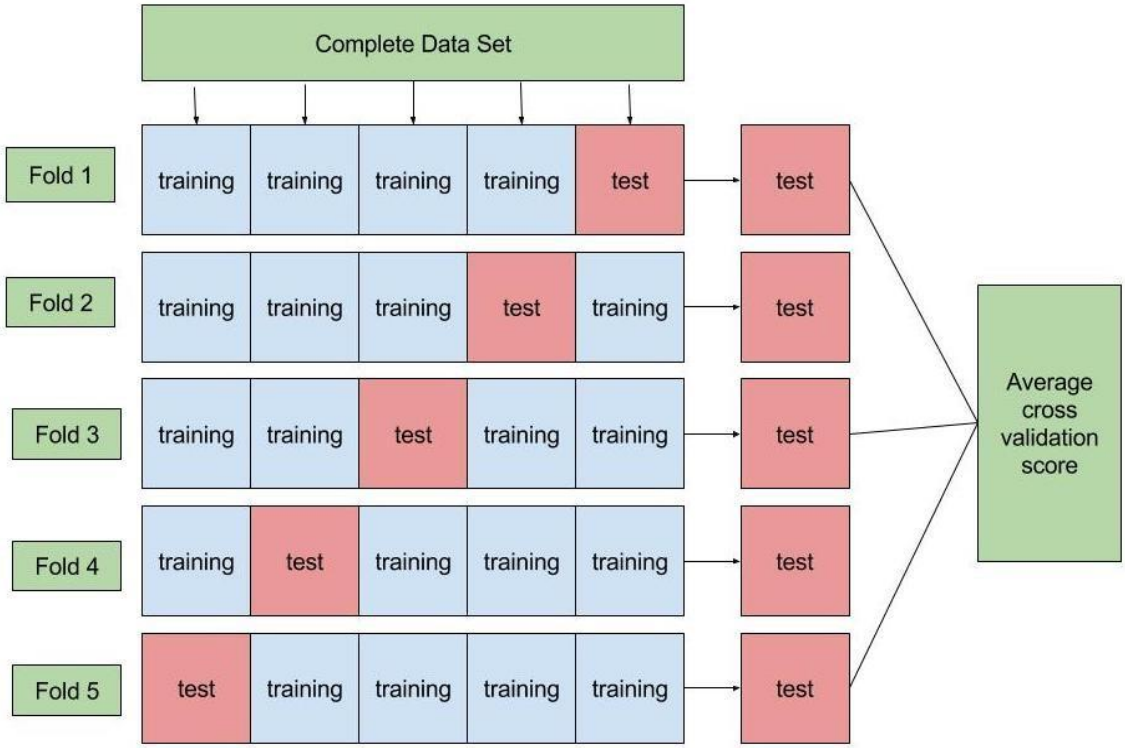


Figure 14: k-fold Cross Validation

The k-fold cross validation technique is designed to serve two purposes, namely model selection and to obtain a conservative error estimation for the model. The first step in the technique is to split our available data into two sets: the training set and the testing set. Next, k-fold cross validation splits the training data into k equal sub-buckets (also known as groups or folds). The learning function is then trained using (k - 1) sub-buckets of training data, and the remaining bucket (fold) is used to validate the model, i.e., to compute an error metric. We repeat this process k times, using all possible combinations of (k-1) sub-buckets and one bucket for validation. Next, we average out the error metric of all the k different trial runs on the model. The model with the lowest average error score is then selected as the final model. Then, we evaluate the final model against the pre-withheld test data to obtain a conservative error estimation. An illustration is provided to explain the method in Figure (14).

The advantage of k-fold cross validation is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set (k-1) times. The variance of the resulting error estimate is reduced as k is increased.

Since we preprocessed the ASD dataset, we didn't run into any problems or difficulties. To implement each of the mentioned methods, we imported and used the following Python modules from Scikit Learn.

- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import RandomForestClassifier
- from sklearn import svm
- from sklearn import neighbors
- from sklearn.naive_bayes import MultinomialNB
- from sklearn.linear_model import LogisticRegression

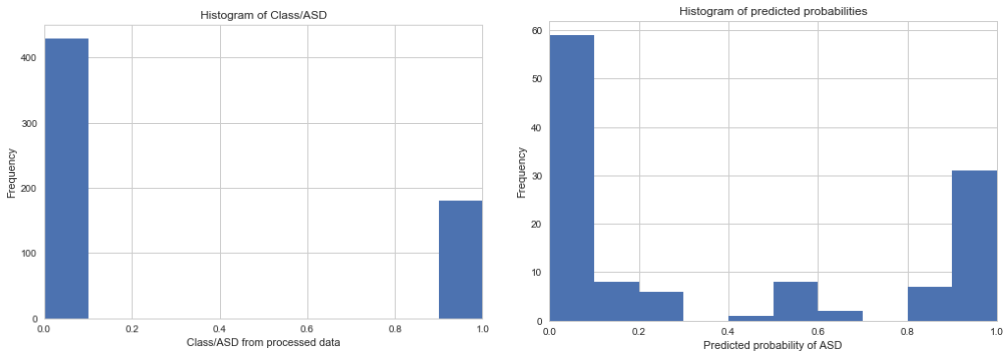
- from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
- from keras.models import Sequential
  from keras.layers import Dense, Dropout, Activation

As we can see in the Figure 15a, there is a clear class imbalance in the target class of in- dividual with ASD. In this situation, we decided to apply AUC-score as well as F1-score in addition to k-Cross validation accuracy as accuracy may not be a proper metric to use in cases like ours where we can observed a definitive class imbalance.

To calculate the desired metrics I imported and used the following Python modules:

- from sklearn.cross_validation import cross_val_score
  cross_val_score(*, features_final, asd_classes, cv=10, scoring='roc_auc').mean()
- from sklearn.metrics import fbeta_score
- from sklearn.model_selection import cross_val_score
  clf = Classifier(random_state=1)
  cv_scores = cross_val_score(clf, features_final, asd_classes, cv=10) cv_scores.mean()
- model = Sequential()
  model.add(Dense(8, activation='relu', input_dim= 94))
  model.add(Dropout(0.2))
  model.add(Dense(1, kernel_initializer='normal', activation='sigmoid')) model.summary()

  score = model.evaluate(X_test, y_test, verbose=0)



(a) Histogram for ASD classes for (b) Histogram for predicted ASD classes

with DT.

cleaned data. Figure 15: Histograms for original & ID3 predicted ASD classes.

I. Decision Trees: ID3Algorithm.

The confusion matrix in this case takes the form:

	individual with ASD	individual with no ASD
predicted: ASD = 'YES'	79	0
predicted: ASD = 'NO'	0	43

Table 3: Confusion Matrix for ASD Data Set

The accuracy of the Decision Tree classifier which measures overall how often thisclassifier is correct is:

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN} = 0.9508$$

and the precision of the Decision Tree Classifier which measures how often the prediction is correct when a positive outcome is predicted is:

$$\text{precision} = \frac{TP}{TP + FP} = 0.9302$$

The sensitivity of the method measures how often the prediction is correct given that the actual outcome is positive:

$$\text{sensitivity} = \frac{TP}{FN + TP} = 0.9302$$

and the specificity measures how often the prediction is correct given that the actual outcome is negative:

$$\text{specificity} = \frac{TN}{TN + FP} = 0$$

Finally, we calculate the false positive rate as

$$\text{false positive rate} = \frac{FP}{TN + FP} = 0.0379$$

which measures how often the prediction is correct given that the outcome was actually negative. Note the values of the following evaluation metrics.

- cross-validation score=1.0
- AUC Score=1.0
- F-Beta Score=1.0 with Beta=0.5.

Note that the Decision Tree Classifier will serve as our Benchmark Model. This means that all future models that we study will be compared to the Decision Tree classifier and may have an accuracy only as good as but not better than the Decision Tree classifier. In other words, no future model can surpass the Decision Tree classifier in terms of performance statistics.

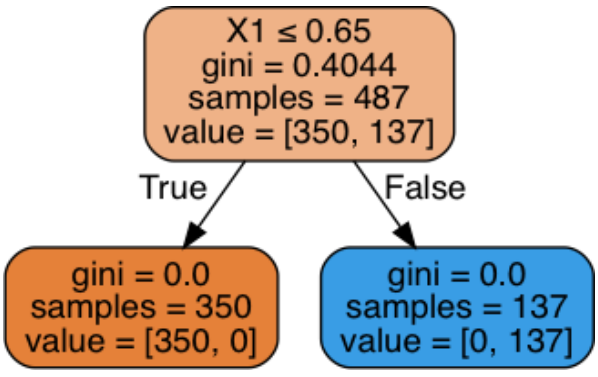


Figure 16: A gini diagram of Decision Tree.

II. Random Forest.

- cross-validation score=0.9933

- AUC Score=0.9988
- F-Beta Score=1.0.

III. Support Vector Machine (SVM).

We start the SVM algorithm with a linear kernel and gamma=2 and find:

- cross-validation score=1.0
- AUC Score=1.0
- F-Beta Score=1.0

IV. K-Nearest Neighbors (KNN).

We apply K-nearest neighbor algorithm with an initial value of K = 10 and observe:

- cross validation score=0.94745
- AUC Score=0.9930
- F-Beta Score=0.9148

V. Naive Bayes.

On applying NB to our data set we find:

- cross-validation score= 0.885
- AUC Score=0.9445 and
- F-Beta Score=0.8370.

VI. Logistic Regression.

We note the following values for evaluation metrics:

- cross validation score=0.9704,
- AUC Score=0.9974, and
- F-Beta Score=0.9307.

VII. Linear Discriminant Analysis.

- cross validation score= 0.9326
- AUC Score=0.9850
- F-Beta Score=0.9148

VIII. Multi-Layer Perception(MLP).

- training accuracy =0.9979
- testing accuracy=0.9836

3.3 Refinement

Here we discuss the possible ways one could further improve the above models.

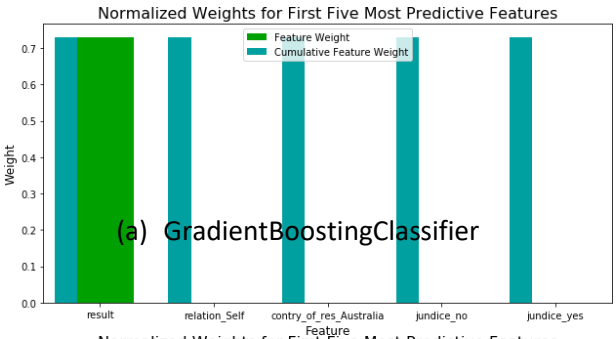
SVM: I found that a non-linear kernel did not yield a good cross-validation score.

kNN: Choosing K is always tricky and we decided to run KNN algorithm using different values of K running through a loop with values ranging from K = 11 to K = 99. The results show that no major improvement in the cross validation scores for different values of K. In fact the best we achieve is K = 94 with a cross validation score of 0.9606.

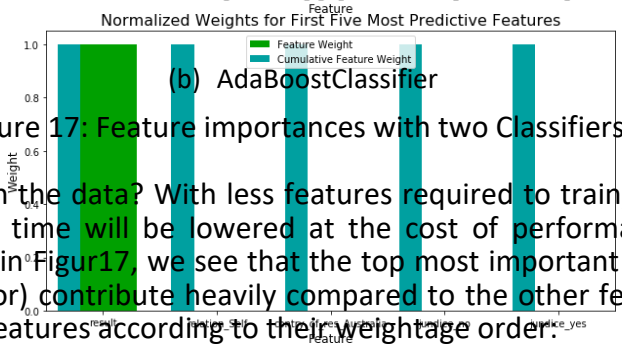
Feature Importance: An important task when performing supervised learning on a dataset like the autistic data we study here is determining which features provide the most predictive

power. By focusing on the relationship between only a few crucial features and the target la- bel we simplify our understanding of the phenomenon, which is almost always a useful thing to do. In case of this project, that means we wish to identify a small number of features that most strongly predict whether an adult individual has ASD or not.

Choose a scikit-learn classifier (such as, gradientBoosting, adaboost, random forests) that has a '.feature_importances_' attribute, which is a function that ranks the importance of features according to the chosen classifier. In Figure 17 shows the top 5 most important features for the ASD dataset using two different classifiers which has that *feature importance* attribute. We need to ask ourselves how does a model perform if we only use a subset ofall



(a) GradientBoostingClassifier



(b) AdaBoostClassifier

Figure 17: Feature importances with two Classifiers

the available features in the data? With less features required to train, the expectation is, the training and prediction time will be lowered at the cost of performance metrics. From the visualization presented in Figur17, we see that the top most important feature 'result' (in term of their weightage factor) contribute heavily compared to the other features. Here we list the first 5 most important features according to their weightage order.

1.'result'
2.'relation self'
3.'country of residence'
4.'jundice-no'
5.'jundice-yes'

This hints that we can attempt to reduce the feature space and simplify the information required for the model to learn. By doing so we can train the model with even with big data set with reduced time.

Learning Algorithm	AUC	F Beta	CVS	Final parameters
Decision Trees	1.0	1.0	1.0	default
Random Forest	0.9988	1.0	0.9933	n-estimators=5, random-state=1
Support Vector Machines	1.0	1.0	1.0	C=1, γ = 2, kernel='linear'
KNN	0.9930	0.9148	0.9474	k=10
Naive Bayes	0.9445	0.8370	0.8850	default
Logistics Regression	0.9974	0.9307	0.9704	default
Linear Discriminant Analysis	0.9850	0.9148	0.9326	default

Table 4: Comparism of metrices using different learning algorithms.

4 Results

In this section, we present the numerical results from different machine learning algorithms applied to our ASD data set. In all these numerical computation, I used *SciKit Learn* mod- ule which is written in Python.

As important as it is to find the right model it is equally important to establish which models may not be the best choice. Evaluation metrics such as AUC Score(AUC), F Beta-score with beta=0.5(F-Beta), Cross-validation score(CVS) for each method are summarized in the table below:

4.1 Model Evaluation and Validation

After exploring the ASD dataset with different learning algorithms, we have arrived at the conclusion that all of our model work extremely well with the data. we have used three different metrics (accuracy, AUC score and F-score) to measure the performance of the models, and it seems like all of the metrics indicated an almost perfect classification of the ASD cases.

Learning Algorithm	Training Accuracy	Testing accuracy
Multi Layer Perceptron (MLP)	0.9979	0.9836

Recall that we had decided to use the Decision Tree Classifier as the Benchmark Model for this problem. Looking over the Table 4, it is abundantly clear that the SVM algorithm with a linear kernel does the best job in classifying new instances into one of the two categories:“ patient has ASD” and “patient does not have ASD”. Hence we should consider SVM as our final model, with the Logistic Regression and Random Forest being a close second. On the other hand the Naive Bayes classifier would not be the method of choice for prediction of new instances given the previously discussed disadvantages of the method and the metric performances that can be observed from Table 4.

Here it is reasonble to infer that all the model performances are so high because of the fact that only one feature('result') is predominant over all others features which we have discussed in the Feature Importance section. We will further explore the issue in more details in Subsection 5.1 below.

4.2 Justification

TOur SVM model achieves the same results as the benchmark model. However, the reader should also note that using the Decison Tree Classifier as the benchmark model which has an accuracy of 1 implies that no other model can superseed the Decision Tree Classifier, but can only achieve the same accuracy.

5 Conclusion

5.1 Free-Form Visualization

As in “Feature Selection” exploration, we have seen the attribute named ‘result’ has such a powerful presence in the ASD dataset, all other attributes have little to no contribution in deciding the final class. In the Figure 18 below, we have drawn a swarmplot with ‘result’ as x-axis and ASD-class (say ‘yes’ is 1 and ’no’ is 0) as y-axis, and reconfirmed the underlying association between the given variables where the target class is easily separable.

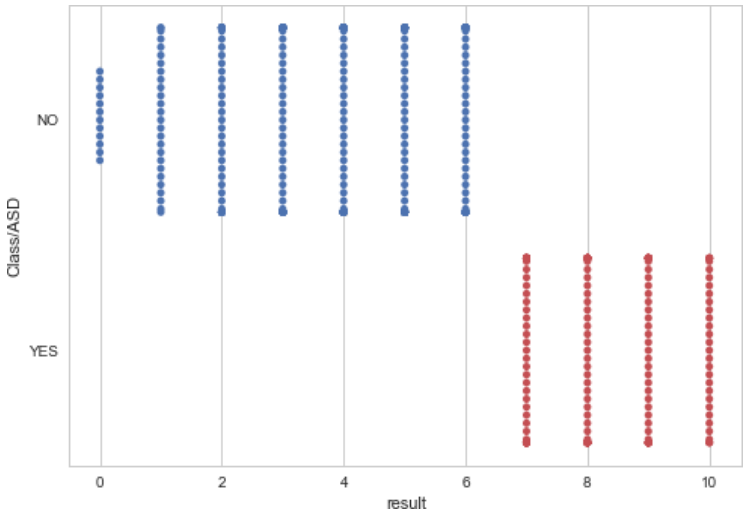


Figure 18: Free-Form Visualization

5.2 Reflection

During pre-processing of the data, we dropped about 95 of rows of data due to its ‘NaN’ entries. Ideally one should try to retain as much data as possible, as there could always be some valuable insight that could be lost. we usually try to fill the missing entries by ‘median filling’ or with a fancier approach, one can also run a supervised learning model to ‘predict’ the ‘NaN’ value. But that approach is not applicable for our dataset as many inputs that was missing are of categorical type and it is not feasible to make a median of a category). While there are some advanced procedures like ‘imputation process’ to take care of these issues of missing values of categorical types without having them dropped, we didn’t explore those techniques in this current work.

Even without implementing more sophisticated techniques like ‘imputation’ in order to re- tain valueable information, we managed to find very efficient models that can classify new instances with accuracy score 1. The reason scores close to 1.0 is because a single feature

is almost entirely responsible for deciding the output. This means that the output can be determined by a high confidence. The present work is certainly quite interesting and is a noble area where technology can be used to actually change lives.

5.3 Improvement

Thus, to summarize, we set out with the hopes of applying machine learning algorithms, specifically, supervised machine learning techniques that can classify new patients (new instances) with certain measurable characteristics (the variables) into one of two categories "patient has ASD" or "patient does not have ASD". Cleaning the data set (which docu- mented the characteristics associated with ASD), was challenging in that we had mostly categorical variables and just two umerical variables, but ultimately we were able to build such models and found that the algorith that performs the best in all aspects is the SVM machine learning algorithm, using a 'linear kernel'. The SVM outperformed all other models with repsect to cross-validation score, AUC Score, and F-Beta Score, all of score were 1, and thus was as good as our benchmark model. Although the data association made the prediction very simple, but we feel this work can certainly serve as an invaluable aid for physicians for detection of new autistic cases.

In our consideration, to build an accurate and robust model, one needs to have larger datasets. Here the number of instances after cleaning the data were not sufficient enough to claim that this model is optimum. Looking at the performances of our learning models, nothing can be improved with this current data set as models are already at their best. After discussing this issue with a researcher directly working on adult autism, we have realised that it is extremely difficult to collect a lot of well documented data related to ASD. This ASD dataset has recently been made public (available from December 2017), and thus not much work has been done. With this in consideration, our research has resulted in well developed models that can accurately detect ASD in individuals with given attributes regarding the persons behavioral and medical information. These models can serve as benchmarks for any machine learning researcher/practitioner who is interested in exploring this dataset further or other data sets related to Autism screening disorder.

References

[1]Brian Godsey, Think Like a Data Scientist *Manning*, ISBN: 9781633430273

[2]H. Brink, J. Richards, M. Fetherolf, Real World Machine Learning, *Manning*, ISBN: 9781617291920

[3]D. Cielen, A. Meysman, M. Ali, Introducing Data Science, *Manning* ISBN: 9781633430037.

[4]J. Grus, Data Science From Scratch First Principles With Python, *O'Reilly* ISBN: 9781491901427

[5]A. Géron, Hands-On Machine Learning with Scikit-Learn & Tensor Flow, *O'Reilly* ISBN: 9781491962299

[6]T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Second Edition, *Springer*

[7]G. James, D. Witten, T. Hastie, R. Tibshirani, An Introduction to Statistical Learning with Applications in R, *Springer*, ISBN 9781461471370

[8]Tabtah, F. (2017). Autism Spectrum Disorder Screening: Machine Learning Adaptation and DSM-5 Fulfillment. *Proceedings of the 1st International Conference on Medical and Health Informatics 2017*, pp.1-6. Taichung City, Taiwan, ACM.

[9]Thabtah, F. (2017). ASDTests. A mobile app for ASD screening. www.asdtests.com [accessed December 20th, 2017].

[10]Thabtah, F. (2017). Machine Learning in Autistic Spectrum Disorder Behavioural Re- search: A Review. *Informatics for Health and Social Care Journal. December, 2017*(in press)