# Machine Learning Engineer Nanodegree

## Capstone Project

Anjali Godbole
March 22, 2018

## 1. Definition

### Project Overview

Autism spectrum disorder (ASD) is a developmental disability that affects 1 in 68 children in the U.S. Children can be diagnosed at age 2, however the average age of diagnosis is 4 years old. Factors including lack of providers and cost contribute to this diagnosis delay. The American Academy of Pediatrics (AAP) recommends that children with an ASD diagnosis begin therapy services as soon as possible to improve long-term outcomes (1). In order to address this delay, a machine learning model could be developed to predict whether a further ASD evaluation is needed (2). This indication could initiate services earlier for the patient. ASD is currently diagnosed using the Diagnostic and Statistical Manual of Mental Disorders (DSM – 5) (3).

A Kaggle dataset of ASD screening data for adults was used to create the model (2). There are 704 patient records in this dataset. The dataset consists of the following attributes:

1. Demographic data – Age, gender, ethnicity, country of residence
2. Answers to 10 behavioral questions from the AQ-10- Adult survey
3. Medical and other information – born with jaundice, family member with ASD, person completing screening
4. ASD Class – whether the patient has ASD

### Problem Statement

In order to address the need for a new ASD screening method to alleviate diagnosis delays, a predictive classification model was developed. This model was developed using patient demographic and behavior data. This model could be used as a screening tool for patients.

The model was developed following the steps below:

1. Explore and preprocess data – Explore data distribution, data type and range and evaluate missing or invalid data. Prepare data for classification model fitting by one-hot encoding data and transforming numerical data.

2. Algorithm selection, parameter tuning and testing – Evaluate classification algorithms, such as decision trees and support vector machines (SVM) for model metrics. Perform parameter tuning on best performing model and test final model on test data set. Evaluate model performance on data subset.

3. Deep learning model – Create a deep learning model and evaluate it on the test data set.

4. Compare the final classification algorithm, deep learning model and benchmark metrics.

### Metrics

The specificity and sensitivity of the classification algorithm and deep learning model will be compared with the DSM-5 method, which has a specificity of 95% and sensitivity of 76% (4). The specificity and sensitivity are defined below:

Specificity = True Negatives/(True Negatives + False Positives)

Sensitivity = True Positives/(True Positives + False Negatives)

Sensitivity is applicable for this problem as it would be important to correctly identify all ASD patients. Specificity would also be important to avoid incorrectly identifying a patient as having ASD when they do not.

## 2. Analysis

### Data Exploration

A. Data summary and input features

A Kaggle dataset of ASD screening data for adults was used. There were 704 patient records, with 189 patients (27%) with ASD.

The input features and data types are listed below. Original column names are in parentheses for column names which were changed for clarity. Example data is shown in brackets:

1. A1_score, A2_score, A3_score, A4_score, A5_score, A6_score, A7_score, A8_score, A9_score, A10_score – These features are results of behavioral screening questions from the AQ-10 adult survey. These scores have integer values of either 0 or 1. [1]
2. age – age of patient (string) [26]
3. gender – gender of patients ( string) [f]
4. ethnicity – ethnicity of patients (string) [White-European]
5. jaundice (jundice)– jaundice in patient at birth (string) [no]
6. family_ASD (austim) – autism in family members (string) [no]
7. country (contry_of_res) – country of residence (string) [United States]
8. used_app_before – whether app was used. Whether a screening app had been used before [no]
9. result – sum of previously described behavioral screening questions (integer) [6]
10. age_desc – description of patient age group (string) [18 and more]
11. relation – who is completing the survey (string) [Self]
12. Class/ASD – whether patient has ASD (string) [NO]

B. Data abnormalities and changes

Several columns of the dataset were dropped. The column 'result' was dropped as this is dependent on the behavioral score features of A1_score, A2_score etc. The 'used_app_before' feature was dropped as this feature is specific to the screening app used during the survey. The ‘age_desc’ was ‘18 and more’ for all records as the dataset consisted of adult patients. The ‘relation’ column was dropped as the person completing the survey is not a patient characteristic.

A few anomalies were noted during data exploration. First, the ‘age’ feature has a data type of ‘string’ though it should be a numerical feature. In addition, the ‘age’ feature has a left skewed distribution. There is also an invalid ‘age’ entry of ‘383’. The ‘age’ and ‘ethnicity’ features have missing records denoted by ‘?’.

## Exploratory Visualization

Figure 1 below shows the age distribution of the dataset. The ‘age’ feature data type should be converted from string to numerical data type. In addition, the invalid ‘383’ and ‘?’ entries are seen in the figure below. The age distribution is left skewed. Please see data preprocessing section below for further discussion of the age feature.

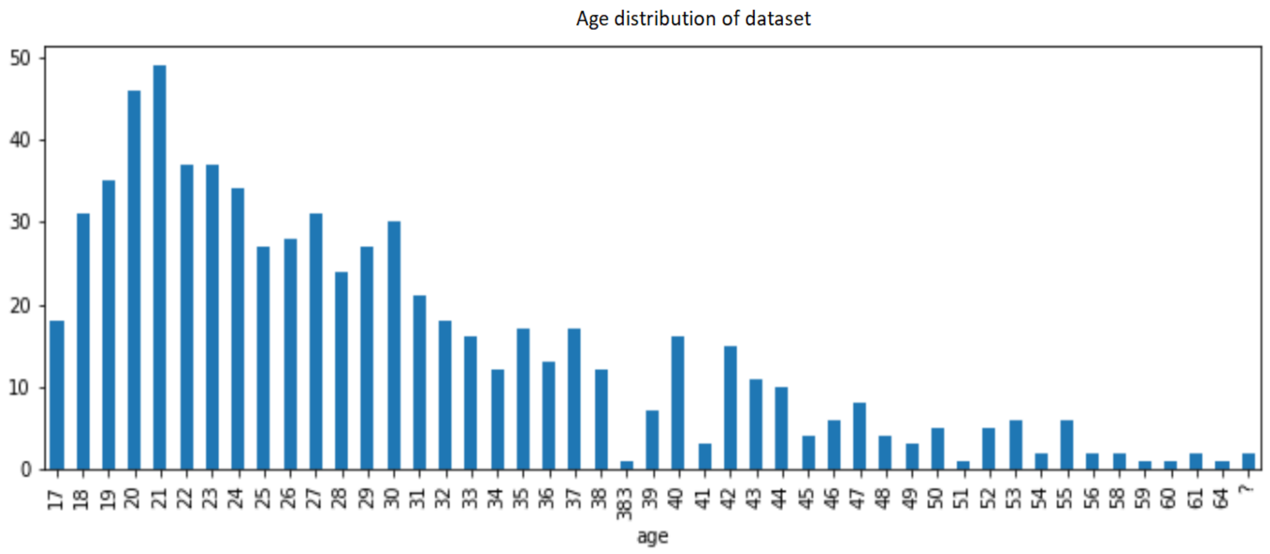

Figure 1 – Age distribution of dataset

Figure 2 below shows missing data denoted by ‘?’ for the ethnicity feature. Please see data preprocessing section below for further discussion of these missing values.

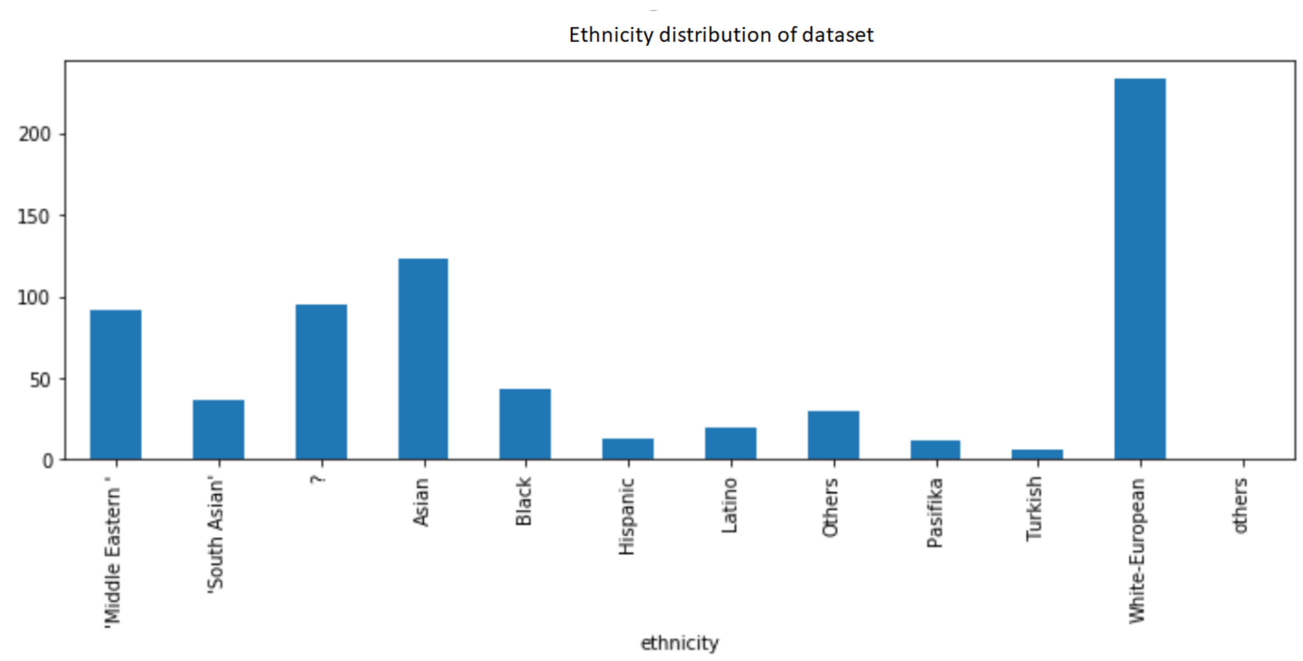

Figure 2 – Ethnicity distribution of data set

**Algorithms and Techniques**

A. Classification algorithms

Four classification algorithms were initially evaluated to determine the best performing model before parameter tuning. Classification algorithms were chosen since the goal is to classify patients as needing further ASD evaluation or not. The default parameters of each algorithm are listed below:

a. Decision tree classifier - (‘gini’ splitting criterion, minimum 2 samples to split node, 1 minimum sample per leaf)

The decision tree classifier splits data into subsets based on features that divide the data into different classes. This process would continue for different features of the dataset. This results in decision rules that are can be applied to new data (8).

The quality of the split is measured by default using ‘gini’ impurity, which is the probability of a random data point being classified incorrectly if it was classified according to the subset class distribution. At least 2 data points are required to split a subset and at least 1 data point is required after the split (10).

b. Support vector machine – (C = 1.0, kernel is ‘rbf’, gamma = 1/number of features)

The support vector machine (SVM) maps data points in multidimensional space and creates a hyperplane that separates the data points into classes. The distance between the hyperplane and the nearest data points is the margin and the hyperplane creates a decision boundary separating data into classes. New data would be divided into target classes based on this separation (11).

The default kernel is ‘rbf’ (radial base function). The kernel is used by the SVM to transform the data into higher dimensional space. The parameter ‘C’ is the error penalty for misclassification of data. Higher ‘C’ values would result in smaller margin hyperplanes while lower ‘C’ values would result in larger margin hyperplanes. (11, 12). The ‘gamma’ parameter is the kernel coefficient. A large ‘gamma’ value would make the decision boundary more dependent on the closest data points. A small ‘gamma’ would make the decision boundary more dependent on farther data points (13).

c. Random forest classifier – (number of estimators = 10, ‘gini’ splitting criterion, minimum 2 samples to split node, 1 minimum sample per leaf)

The random forest classifier uses random samples of the training data to create many decision trees. To make a prediction, each decision tree will "vote" by classifying the data point as with or without ASD. The majority vote of all of the decision trees would be the final prediction (14).

The default number of estimators (number of decision trees) is 10. As described in the decision tree classifier section above, the split quality is measured using ‘gini’ impurity. A minimum of 2 data points are required to split a subset and more than 1 data point is required after the split (10, 15).

d. Gaussian Naive Bayes –

The Gaussian Naive Bayes model makes predictions on new data based on calculated class and conditional probabilities of training data. The probability of a new data point belonging to each class would be calculated using its features and the previously calculated feature probabilities from the training data. Features are assumed to be independent and follow a Gaussian distribution (16).

B. Parameter tuning and feature selection

After selection of the classification model, GridSearchCV was used to optimize the model over a range of model parameters. In addition, SelectKBest was used with Pipeline and GridSearchCV to select a subset of features and evaluate the performance of the optimized classification model. SelectKBest selects features based on function scores. The default function used calculates the ANOVA F value between each feature and the target (17).

C. Deep learning model

A deep learning model consisting of fully connected layers was also created. Model performance was compared with the optimized classification model.

The deep learning model consisted of an input layer, hidden layer and output layer. The input layer calculates a weighted sum of the inputs and applies the activation function to the sum. The activation function applied was the ‘relu’ function. This function outputs 0 for values < 0 and has a linearly positive output for values > 0. The output from this layer is then inputted into a dropout layer, which drops out a fraction of the inputs. The output from the dropout layers is then inputted into the output layer, which applies the sigmoid function to the weighted sum of the inputs. The sigmoid function is applied to this output to produce 1 for values greater than 0.5 and 0 for values less than 0.5. The 0 and 1 outputs correspond to the target classes in the dataset. Training errors are propagated through the model through training iterations to adjust the layer weights (18, 19).

D. Cross validation

Both the optimized classification model and the deep learning model were cross-validated using Stratified Kfold. In cross-validation, the training data is randomly split into training and validation data a number of times (folds). After each iteration, predictions are made on the validation data and metrics are calculated. Using Stratified Kfold ensures that the percentage of samples labeled as with or without ASD in each fold is the same as the original dataset (20, 21).

**Benchmark**

The classification models will be compared to the current DSM-5 method with specificity of 95% and sensitivity of 76%. The DSM-5 evaluates deficits in the areas of communication and social interaction, restrictive behavior and interests, and impaired functioning (5). The evaluation is usually performed by a team consisting of a pediatrician, psychologist, speech and language pathologist and occupational therapist (6).

## 3. Methodology

**Data Preprocessing**

Several data preprocessing steps were required for invalid, missing and skewed data as previously discussed in the data exploration section.

A. 'age' feature – The 'age' feature data type was changed to numerical. In addition the record with the invalid '383' entry was removed from the data set. The 'age' distribution was left skewed so it was log transformed.

B. Missing data - The 'age' and 'ethnicity' features contained missing data denoted by '?'. These records were removed from the dataset. After deleting these records, there were 609 records in the dataset.

C. Feature scaling – The data was separated into features and targets. The numerical features of 'age' and the behavioral question scores of 'A1_Score', 'A2_Score', etc. were scaled using MinMaxScaler. The transformed data will be in the range of 0 to 1.

D. One-hot encoding and class transformation – All categorical features were transformed using one-hot encoding. In addition, the target feature of 'Class/ASD' was transformed to 1 for 'YES' and 0 for 'NO'.

**Implementation**

All implementation was performed using the sklearn library in Python. The feature and target data were separated into training and testing data with 20% of data reserved for testing (122 testing samples, 486 training samples). This was performed using the 'train_test_split' module in the sklearn.model_selection library.

A. Classification algorithms

A classification algorithm was initially chosen using default parameters. The decision tree classifier, SVM, random forest classifier and Gaussian Naive Bayes classifier were trained on the training data. The sensitivity (recall) and specificity were calculated for each classifier.

A coding complication arose in trying to efficiently train each classifier and store the sensitivity and specificity results. In order to address this complication, a function (clf_predict) was written to train each classifier, make predictions and evaluate the sensitivity and specificity. The function was called with a 'for' loop. The sensitivity (recall) was evaluated using the 'recall_score' module from the 'sklearn.metrics' library. The specificity was calculated using the 'confusion_matrix' from the 'sklearn.metrics' library. The confusion matrix returns a count of the true and false negatives, and true and false positives based on the classifier prediction. See metrics section above for specificity calculation.

Results are shown below in Table 1 with the sklearn library and module for each classifier. The SVM classifier had the highest sensitivity and specificity.

| **Classifier** | **Sensitivity (recall)** | **Specificity** | **sklearn library and module** |
|---|---|---|---|
| Gaussian NB | 0.938 | 0.078 | sklearn.naive_bayes, GaussianNB |
| SVM | 0.906 | 0.956 | Sklearn.tree, DecisionTreeClassifier |
| Decision Tree | 0.875 | 0.90 | Sklearn.svm, SVC |
| Random Forest | 0.781 | 0.978 | Sklearn.ensemble, RandomForestClassifier |

Table 1 – Classifier sensitivity and specificity

B. Deep learning model

A deep learning model was created using keras running on Tensorflow. The model was defined as 'Sequential' using the 'Sequential' module from the 'keras.models' library. The model was initially created with fully connected layers as described below from the 'keras.layers' library.

1. 512 inputs, 'relu' activation ('Dense' and 'Activation' modules)
2. Dropout layer of 20 % ('Dropout' module)
3. Output layer with 'sigmoid' activation ('Dense' and 'Activation' modules)

The model was compiled using 'binary_crossentropy' as the loss function and the optimizer was 'adam'. The model was trained for 80 training epochs. This model had a sensitivity of 0.969 and specificity of 0.944.

## Refinement

A. SVM classifier

The SVM classifier underwent further parameter tuning to optimize the recall as this was lower than the specificity. Since sklearn does not include a specificity scorer, a specificity function was created to evaluate this metric using the confusion matrix from the test predictions. Using GridSearchCV, the 'C' and 'gamma' parameters were optimized. Table 2A below summarizes the parameters and features used for each model and the specificity and sensitivity results.

| Parameter search | Parameters | Specificity | Sensitivity | Dataset features |
|---|---|---|---|---|
| 'C': from 1 to 10, interval = 1 'gamma': from $10^{-9}$ to $10^{13}$, interval = $10^{3}$ | 'C': 4, 'gamma': 0.1 | 0.978 | 0.938 | 88 (all features) |
| N/A | 'C': 4, 'gamma': 0.1 | 0.989 | 0.938 | 20 |

Table 2A – SVM parameter optimization

The optimal values were 'C' = 4 and 'gamma' = 0.1. Using these parameters, the sensitivity (recall) was 0.938 and specificity was 0.978.

In addition, SelectKBest was used to find the most important features in the dataset. Using Pipeline and GridSearchCV with the previously found optimized classifier, the optimal number of features found was 20, sensitivity (recall) was 0.938 and specificity was 0.989.

B. Deep learning model

The previous deep learning model was optimized by changing the number of inputs and evaluating the specificity and sensitivity of the test set predictions. These results are shown below in Table 2B.

| Number of inputs | Specificity | Sensitivity | Dataset features |
|---|---|---|---|
| 512 | 0.944 | 0.969 | 88 (all features) |
| 256 | 0.938 | 0.967 | 88 (all features) |
| 128 | 0.956 | 0.969 | 88 (all features) |
| 128 | 0.922 | 0.969 | 20 |

Table 2B – Deep learning model specificity and sensitivity

The models with 512 and 128 inputs had the highest sensitivity of 0.969 and the model with 128 inputs had the highest specificity of 0.956.

This model with 128 inputs will be further compared with the SVM classifier and the benchmark model since it has the highest specificity. This corresponds to the higher specificity metric for the benchmark model. Using the subset of features previously found using SelectKBest, the sensitivity was 0.969 and the specificity was 0.922.

## 4. Results

### Model Evaluation and Validation

The optimized SVM model and the deep learning model were each evaluated for overfitting and underfitting using the subset of features found using SelectKBest.

A. SVM Classifier

The optimized SVM model had parameter values of ‘C’ = 4 and ‘gamma’ = 0.1. The classifier was trained using the subset of features found from SelectKBest. The sensitivity and specificity of the SVM classifier were 0.938 and 0.989 respectively. Figure 3 below shows the learning curve for this classifier.

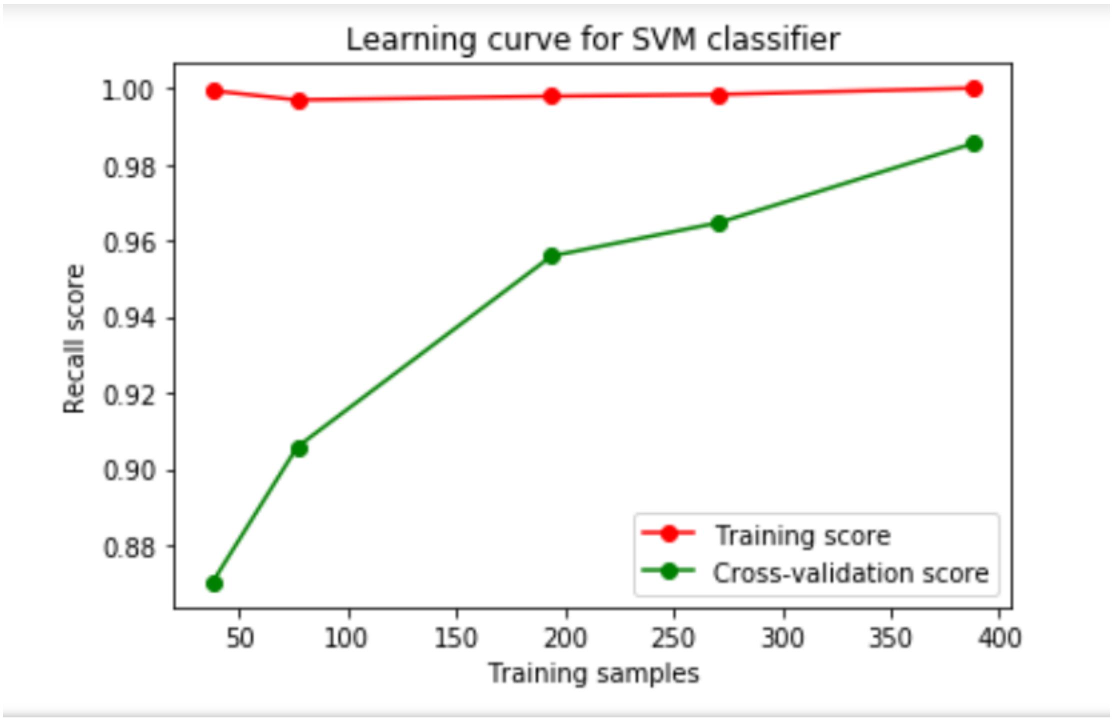

Figure 3 – SVM classifier learning curve

In the above figures, the recall (sensitivity) versus the number of training samples is shown. The training and validation scores do not converge when using the maximum number of training samples. This would imply that the model is underfitting the data. Additional training data is needed for this model.

In addition, the optimized SVM model was cross-validated using the StratifedKfold module from the sklearn.model_selection library. The model was cross-validated using 10 folds. The recall and specificity were calculated for each fold. The results from the cross-validation are shown below in Table 3.

| Fold number | Sensitivity (Recall) | Specificity |
|---|---|---|
| **1** | 0.933 | 1.000 |
| **2** | 0.933 | 1.000 |
| **3** | 0.933 | 1.000 |
| **4** | 1.000 | 1.000 |
| **5** | 0.867 | 1.000 |
| **6** | 1.000 | 0.971 |
| **7** | 0.867 | 1.000 |
| **8** | 0.867 | 0.971 |
| **9** | 1.000 | 1.000 |
| **10** | 0.929 | 0.970 |
| | | |
| **Average** | 0.933 | 0.991 |
| **Standard deviation** | 0.054 | 0.014 |

Table 3 – Results of SVM classifier Stratified K-fold cross-validation

The average sensitivity was 0.933 with a standard deviation of 0.054 and the average specificity was 0.991 with a standard deviation of 0.014. These results are comparable to the previously calculated sensitivity of 0.938 and specificity of 0.989 of the optimized SVM model. However, the model seems to be underfitting the data due to small dataset size as discussed above.

## B. Deep learning model

This deep learning model with 128 inputs had a sensitivity of 0.969 and a specificity of 0.922 when using the subset of features found from SelectKBest. Figure 4 below shows the training loss of the model versus training epoch.

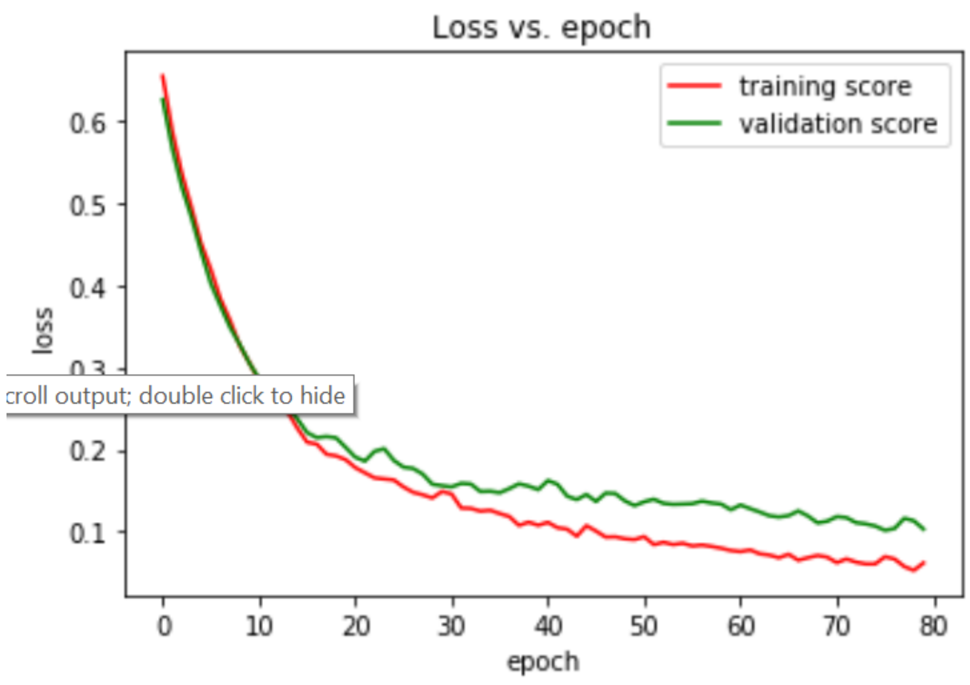

Figure 4 – Deep learning model training loss vs. epoch

The training and validation scores diverge after about 25 epochs. This implies that the model is overfitting the data when trained for 80 epochs. In order to address this concern, the model was only trained for 25 epochs and subsequently evaluated. Figure 5 below shows the training loss per epoch for 25 epochs.

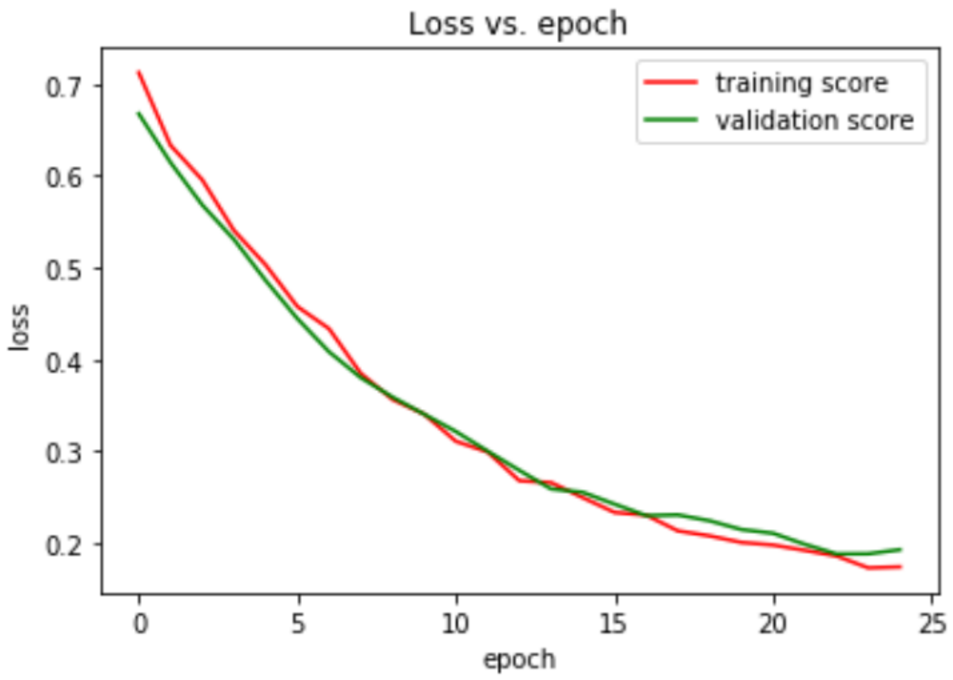

Figure 5 – Deep learning model training loss vs. epoch

Figure 5 shows the training and validation scores have converged and that the model is not overfitting the data. After training for 25 epochs, the model sensitivity was 0.875 and specificity was 0.956.

In addition, the model was cross-validated over 25 epochs using the subset of features found from SelectKBest. The model was cross-validated using StratifiedKfold with 10 folds. Table 4 below shows the results of the cross-validation.

| **Fold number** | **Sensitivity (Recall)** | **Specificity** |
|---|---|---|
| **1** | 1.000 | 1.000 |
| **2** | 1.000 | 0.971 |
| **3** | 1.000 | 0.971 |
| **4** | 0.867 | 0.971 |
| **5** | 0.933 | 0.971 |
| **6** | 0.933 | 1.000 |
| **7** | 0.933 | 0.941 |
| **8** | 1.000 | 1.000 |
| **9** | 1.000 | 1.000 |
| **10** | 0.929 | 0.878 |
| | | |
| **Average** | 0.960 | 0.970 |
| **Standard deviation** | 0.047 | 0.038 |

Table 4 – Results of deep learning model Stratified K-fold cross-validation

The average sensitivity was 0.960 with standard deviation of 0.047. The average specificity was 0.970 with standard deviation of 0.038. The results are comparable with the previously found sensitivity of 0.875 and specificity of 0.956.

**Justification**

The specificity and sensitivity results for the SVM classifier, deep learning model and benchmark are summarized in Table 5 below.

| Model | Specificity | Sensitivity |
|---|---|---|
| SVM | 0.989 | 0.938 |
| Deep learning | 0.956 | 0.875 |
| Benchmark | 0.95 | 0.76 |

Table 5 – Model specificity and sensitivity

As shown in Table 5, the SVM model had the highest specificity of 0.989 compared with the benchmark specificity of 0.95. Both the SVM model and the deep learning model had higher sensitivities (0.938 and 0.969 respectively) than the benchmark sensitivity of 0.76. However, a direct comparison between the benchmark and the models should not be made for the following reasons:

1. Number of samples – The benchmark result used data from 977 patients while the SVM and deep learning models were trained on 486 samples (4). The benchmark result used twice the number of samples in the patient study.

2. Overfitting/Underfitting – As shown in the Analysis section above, the SVM model is underfitting the data. This could be due to a lack of samples. In addition, the deep learning model is overfitting the data after training for 25 epochs, which prevents the use of additional training epochs. This also could be due to lack of a lack of samples.

3. Patient population – The benchmark result used data from a patient population with average age of 9 years, while the patient average age for the dataset used in this project was about 30 years. The survey questions used in this dataset were from an adult ASD screening survey, which would not be applicable for a child's diagnosis.

The results from this project cannot directly be compared to the benchmark result. The results are an initial step in understanding the how to build a classifier using patient demographic and screening data to screen for ASD. Both the SVM and deep learning models could potentially be used for further controlled studies to examine their effectiveness for screening ASD.

## 5. Conclusion

### Free-Form Visualization

It is informative for future studies to asses what features of the dataset have the highest SelectKBest features scores. Figure 6 below show the SelectKBest feature scores of the dataset.

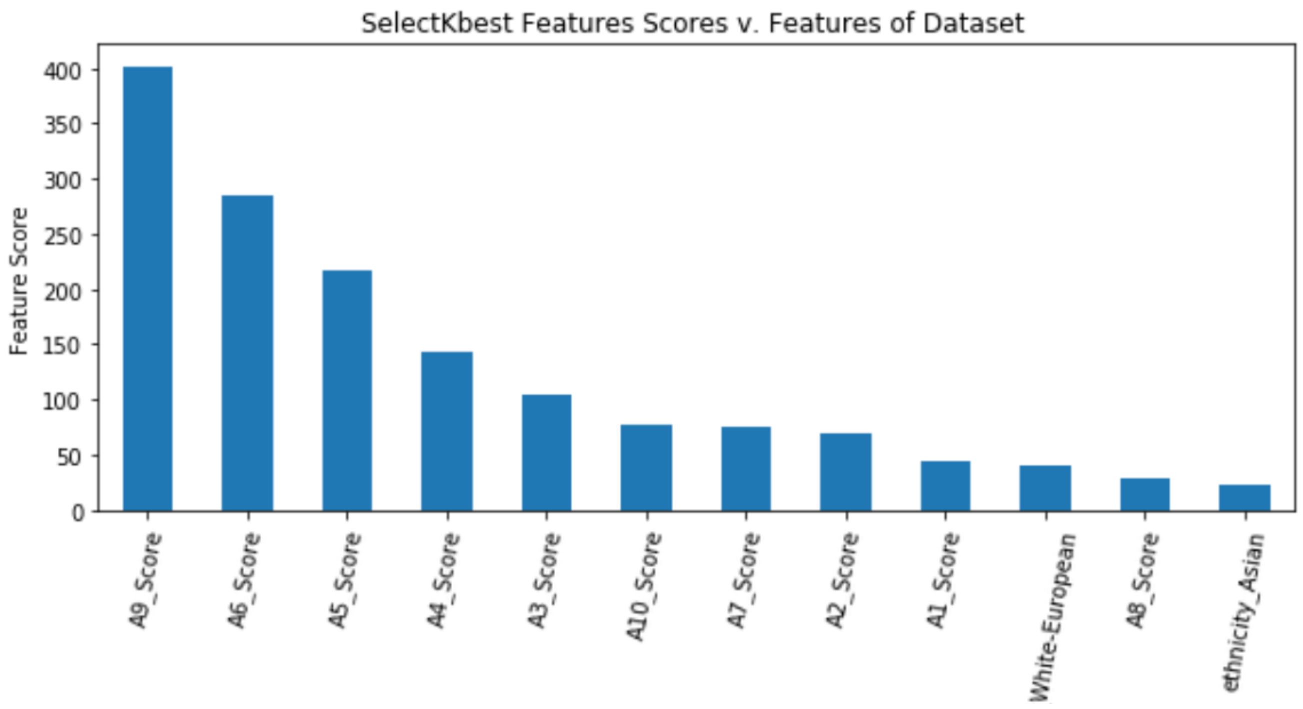

Figure 6 – SelectKbest feature scores

The three highest scoring features are the scores from questions A9, A6 and A5 from the AQ-10 adult screening survey. Patients are scored 1 point if they definitely or slightly disagree with these questions (7). These questions are listed below (7):

A9: “I find it easy to work out what someone is thinking or feeling just by looking at their face.”
A6: “I know how to tell if someone is listening to me is getting bored.”
A5: “I find it easy to ‘read between the lines’ when someone is talking to me.”

These results could be used in future studies to decrease screening questions further and focus on the questions and patient characteristics that are the most likely predictors of ASD.

**Reflection**

In this project, the goal was to create a classifier that could be used for screening patients for further ASD evaluation. The dataset consisted of patient screening questions scores and demographics such as age and country of residence. The dataset was first explored for missing, invalid or skewed data. The dataset was then pre-processed to normalize and transform skewed data.

Initially, four classifiers were assessed for sensitivity and specificity metrics on data predictions. The SVM classifier had the highest performance. The SVM classifier then underwent parameter tuning. A deep learning model was also created using 512 inputs. Next, SelectKBest was used to select a subset of features. In addition, the deep learning model was evaluated using different numbers of inputs. The final model had 128 inputs.

The SVM model was then validated over different numbers of training samples. It was observed from this analysis that the SVM model was underfitting the data. The deep learning model was also validated over different numbers of epochs. It was found that the deep learning model was overfitting the data after 25 epochs.

The overfitting/underfitting of these models could be due a lack of samples. In addition, data from adult patients was used in this project, while the benchmark study used data from children with ASD. Therefore, results from these models cannot be compared to the benchmark result.

It is interesting to understand how different features, such as certain screening questions, are most relevant to the ASD patient prediction. This could be used in the future to reduce the number of features required to predict a requirement for ASD evaluation.

There was some difficulty in understanding how to assess underfitting and overfitting of the classifiers There are a relatively low number of samples in this dataset. In addition, some records were removed due to missing entries, which also reduced the number of samples.

## Improvement

One aspect of the project that could be improved would be to test more variations of the deep learning model. The deep learning model in this project only tested models with a few different number of inputs. More variations such as number of inputs, layers, learning rate, and optimizers could be used to construct different deep learning models. These models could exceed the performance of the SVM and deep learning models assessed in this project and offer more choices for further studies.

## References

1. Gordon-Lipkin E, Foster J, Peacock G. Whittling Down the Wait Time: Exploring Models to Minimize the Delay from Initial Concern to Diagnosis and Treatment of Autism Spectrum Disorder. *Pediatric clinics of North America*. 2016;63(5):851-859. doi:10.1016/j.pcl.2016.06.007.
2. https://www.kaggle.com/faizunnabi/autism-screening
3. https://www.psychiatry.org/psychiatrists/practice/dsm
4. McPartland JC, Reichow B, Volkmar FR. Sensitivity and specificity of proposed DSM-5 diagnostic criteria for autism spectrum disorder Running Head: DSM-5 ASD. *Journal of the American Academy of Child and Adolescent Psychiatry*. 2012;51(4):368-383. doi:10.1016/j.jaac.2012.01.007.
5. https://www.autismspeaks.org/what-autism/diagnosis/dsm-5-diagnostic-criteria
6. https://www.autismspeaks.org/what-autism/diagnosis
7. http://docs.autismresearchcentre.com/tests/AQ10.pdf
8. https://en.wikipedia.org/wiki/Decision_tree_learning
9. http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
10. https://en.wikipedia.org/wiki/Support_vector_machine
11. http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
12. https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel
13. https://www.quora.com/What-are-C-and-gamma-with-regards-to-a-support-vector-machine
14. https://en.wikipedia.org/wiki/Random_forest
15. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
16. https://machinelearningmastery.com/naive-bayes-for-machine-learning/
17. http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html

18. https://devblogs.nvidia.com/deep-learning-nutshell-core-concepts/
19. https://en.wikipedia.org/wiki/Rectifier_(neural_networks)
20. https://www.openml.org/a/estimation-procedures/1
21. http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html