Tree Project

Decision Trees and Ensemble Methods Predicting Autism Spectrum Disorder

Connor Kordes - JCK160130
Nathan Tompkins - NAT180002
University of Texas at Dallas
CS 4372

Results Analysis

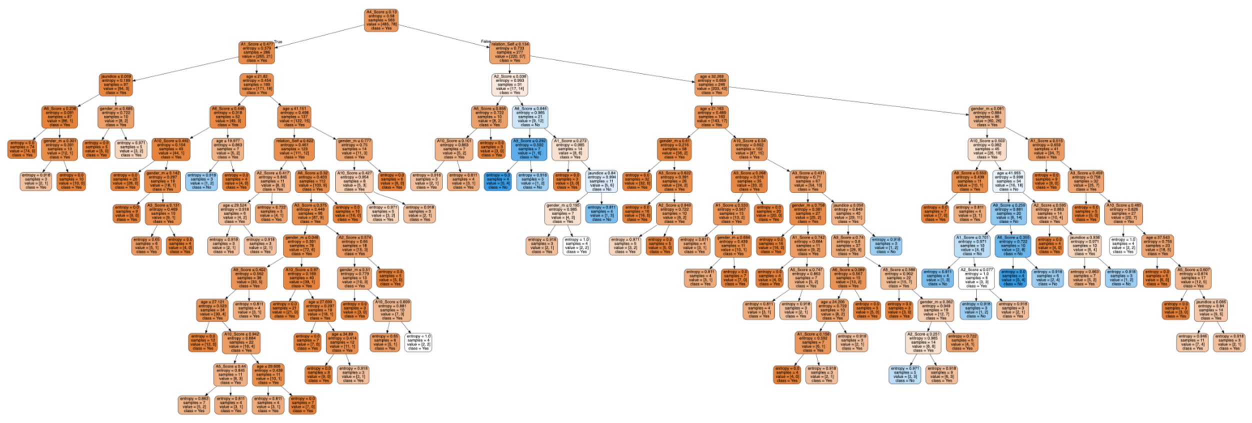
Decision Trees

A decision tree is a method which uses a tree that splits based on identifying attributes about the data. The model determines each split by maximizing the amount of information gained according to the attribute. The first split of the data generally gives a large amount of information. This could be an attribute like age or gender. In our case, the most information is gained by selecting the score of question 4. Decision trees tend to be very complex models, as they don't use ensemble methods like our other models. We can see from the max_depth = 50 parameter just how deep it goes. Parameters can be reused to split the data, as long as the max amount of information is gained from the split.

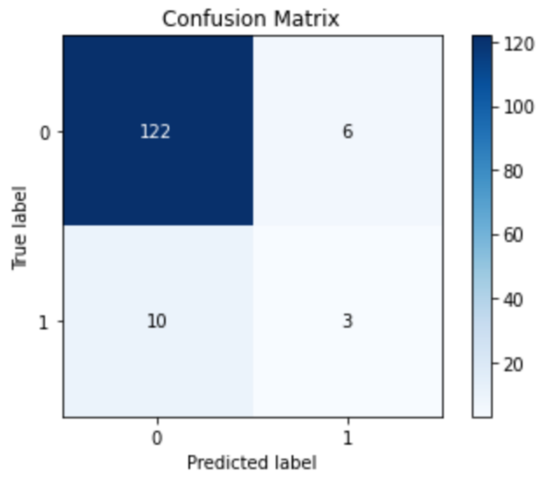
Looking at the confusion matrix, we can see a majority of the data is true negative, meaning that the model classifies most people as not having autism correctly. This can be misleading however, as our data is unbalanced in favor of those without autism. The sample of people with autism is much smaller than those without it, so we have to balance it using balanced_accuracy. This gives the true positive category a higher weight so that our model is not biased towards true negative.

One downside of decision trees is that they rely on a single tree and have a bias towards some estimators. This problem can be fixed by moving to a Random Forest classifier.

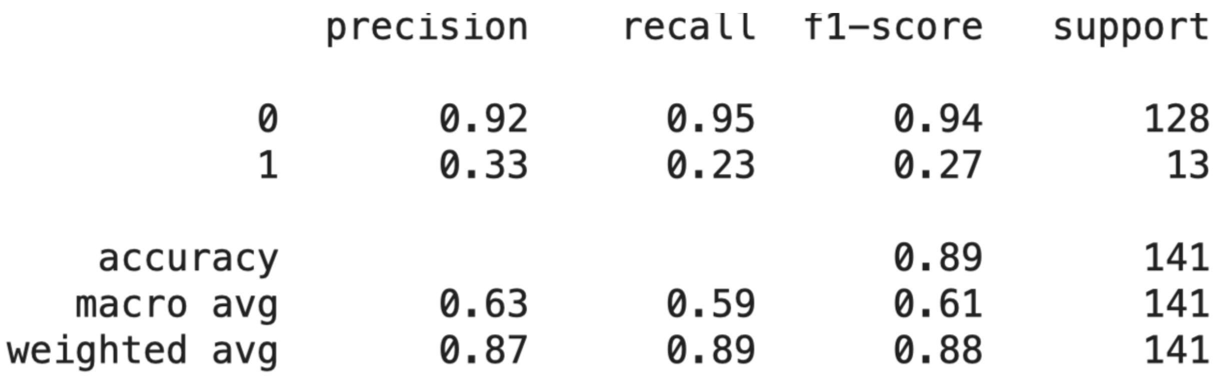
Decision Tree Visualization



Confusion Matrix



Metrics



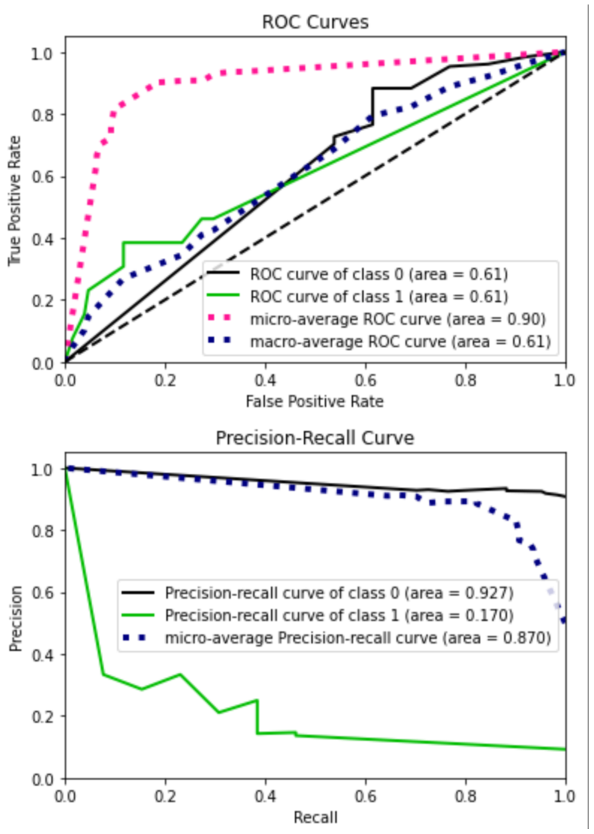
```
              precision    recall  f1-score   support

           0       0.92      0.95      0.94       128
           1       0.33      0.23      0.27        13

    accuracy                           0.89       141
   macro avg       0.63      0.59      0.61       141
weighted avg       0.87      0.89      0.88       141
```

Hyperparameters:

criterion: entropy, max_depth: 50, min_samples_leaf: 3, min_samples_split: 5, splitter: random

We can see that the precision, recall, and f-statistic is much higher for the negative class than the positive class. With our unbalanced dataset, we expect this because of the amount of people without ASD.

ROC and Precision-Recall Curves:



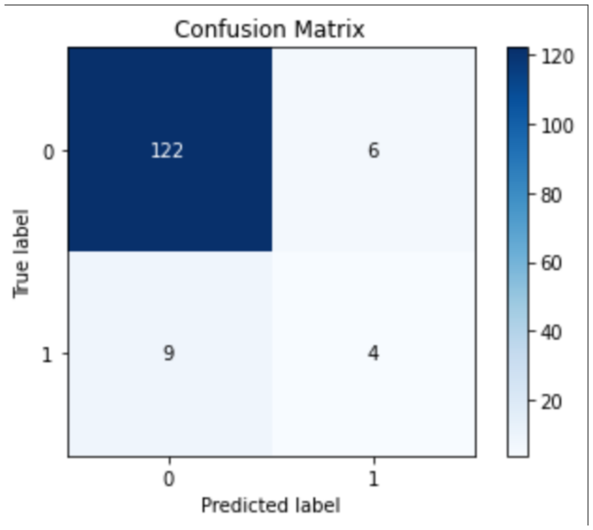
For the decision tree classifier, we see from the ROC curve that it does not perform very well. The ROC curve of class 1 (green line) barely increases its slope at any point. This leads to a smaller area under the curve (.61), indicating under-performance.

Random Forests

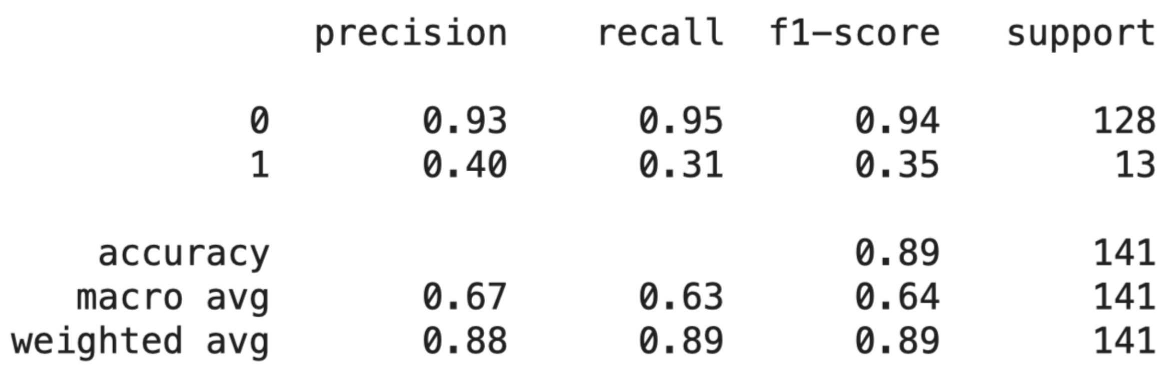
Random forests are like decision trees except they generally perform better. Random forests also use tree models like decision trees, however, it creates multiple trees to use together. It selects a random subset of features and creates a decision tree for each subset. One upside of this is that the entire model is not so biased towards one feature that the first decision tree would normally split on. After all the decision trees are created, we combine the trees together to get our model. Since we are combining the trees, this model is defined as using the ensemble method. This leads to a higher accuracy and reduces variance on the test data.

Looking at the confusion matrix, we see that it is almost the same as the decision tree's confusion matrix. This is to be expected because of the unbalanced dataset. We use balanced_accuracy again to ensure that the model is not biased towards the true negative category.

Confusion Matrix



Metrics



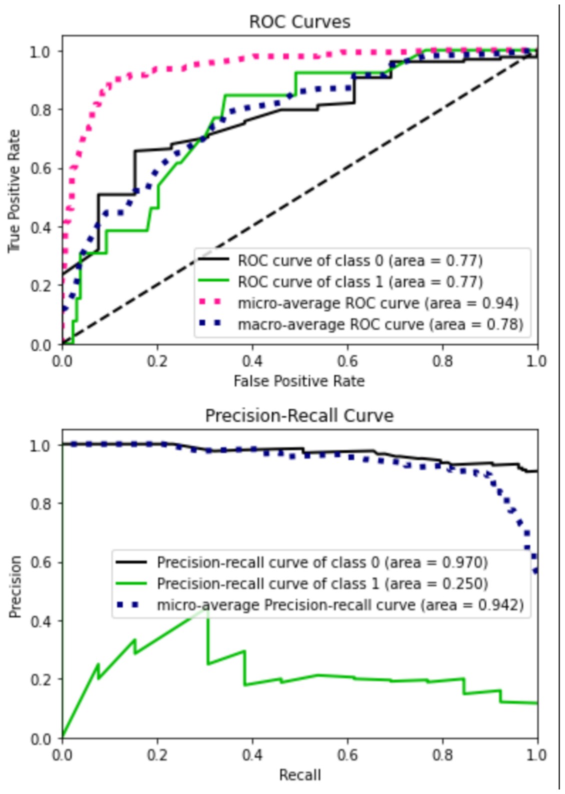
```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       128
           1       0.40      0.31      0.35        13

    accuracy                           0.89       141
   macro avg       0.67      0.63      0.64       141
weighted avg       0.88      0.89      0.89       141
```

Hyperparameters

max_depth: 21, max_features: log2, min_samples_leaf : 1, min_samples_split: 2, n_estimators: 100

Compared to the decision tree results, we can see that the accuracy of this model is considerably better. The precision, recall, and f-score of the positive class are all higher. This is to be expected because of the randomness we have introduced into the classification which lowers the bias for certain features.

ROC and Precision-Recall Curves:



Compared to the last ROC curve, we see that the curve for class 1 is much better. We can see the slope increase around an FPR of about 0.2. The area under the curve (.77) is much higher than the last classifier, so we have already found a better classifier than the decision tree.

Adaboost

Adaboost is an ensemble method that takes advantage of weak classifiers that are completely uncorrelated to produce an additive model that will make classifications that generally should perform better on test data than large decision trees or random forests in which there is still some correlation between the trees which increases the variance in test predictions.

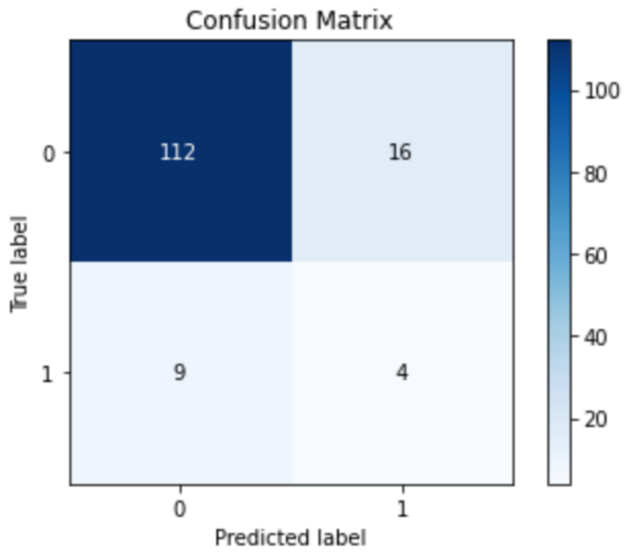
Hyperparameters

```
'ab__base_estimator': DecisionTreeClassifier(max_depth=2),
'ab__learning_rate': 1.35, 'ab__n_estimators': 45
```

Before we evaluate the model, it is important to discuss the hyperparameters that were chosen for the model. For this we can see that there was still a decision tree classifier with max depth_2 as the stump was not able to interpret the data well. Additionally the learning rate, the parameter that weights new information was 1.35 and the number of estimators was 45, a relatively small number.

Moving to the confusion matrix of this data set, one can see that we had a varied performance in which most observations were correctly classified as negative and 16 observations were classified as positive when they should have been classified as negative. This phenomenon is most likely due to the scoring metric that was used in cross validation, balanced accuracy, a useful metric for unbalanced data sets like this one. However it will make the unbalanced class predictions less accurate. Moving to true positives the algorithm correctly found four. This was particularly important to the group because these observations represent someone who had ASD and were correctly predicted to have it based on the features. With this, it still missed nine observations and classified them negative. This was slightly discouraging but speaks to the difficult nature of the problem.

Confusion Matrix

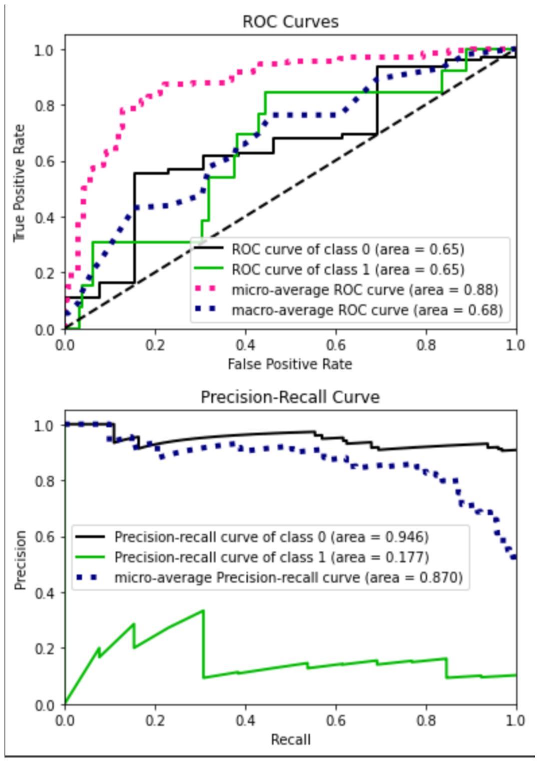


Metrics

```
              precision    recall  f1-score   support

           0       0.93      0.88      0.90       128
           1       0.20      0.31      0.24        13

    accuracy                           0.82       141
   macro avg       0.56      0.59      0.57       141
weighted avg       0.86      0.82      0.84       141
```

The metrics above are based on the confusion matrix discussed earlier. Since this was a binary classification problem, we can see two metrics for precision and recall. Again, the group was working with an unbalanced data set so the challenge was balancing precision and recall amongst the classes. With this however, there was a significantly lower precision and recall for the positive class than for the negative class. In terms of precision this meant we incorrectly said many values were positive without them actually being positive and in terms of recall we incorrectly predicted negative for many observations that were positive. Thus, adaboost did not succeed in finding a good separation in the data. Next we will look at the ROC and precision and recall curves below.

Precision, Recall, ROC Curve



Looking at the ROC curves we can see we had an Area Under the Curve (AUC) of 0.65. Anything below 50 would be unacceptable so this is not terrible, but still not ideal. Additionally the curve only makes a small jump at the beginning meaning it cannot distinguish that well between the classes.

Moving to the Precision and Recall Curve, we can see that the precision and recall remain much lower for the positive class despite different probability thresholds and that the algorithm generally does poorly at predicting positives. Next we will move the XGBoost, the model of champions.

XGBoost

XGboost coined the algorithm of champions due to its large success rate in prediction and the secret sauce to its conception, clearly defining tree weights. This allows regularization to be performed quickly and optimally. Let's take a look at the hyperparameters that were best on this data set for XGBoost

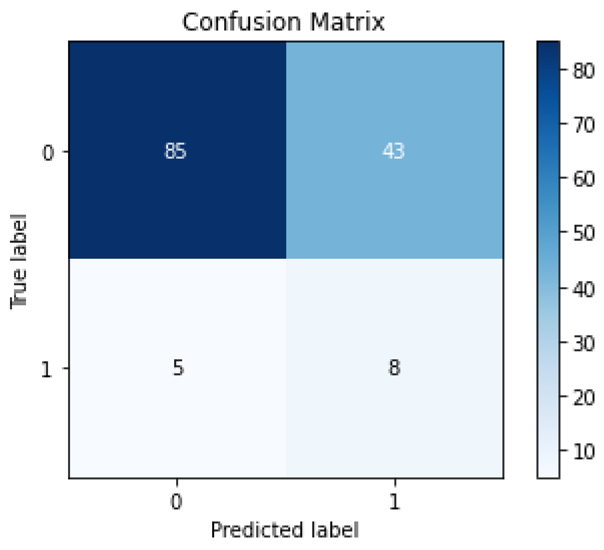
Hyperparameters

```
'eta': 0.005, 'lambda': 0, 'max_depth': 1, 'n_estimators': 100,
'scale_pos_weight': 6}
```

The eta refers to the learning rate, in this case a very low learning rate was optimal. Additionally, no regularization was preferred by the algorithm with a relatively low amount of estimators sitting at 100. Finally the scale_pos_weight is used for unbalanced data sets and is usually the number of negative observations over the positive. Here 6 was chosen by the algorithm which roughly approximates the TN/TP ratio.

Confusion Matrix

Next we will look at the confusion matrix. Here we can see something much different than before. We get most of the positive observations classified correctly but in term we see a sacrifice in our precision of the positive class. Consequently this algorithm classifies many as positive who are actually negative which could be a big mistake given that in this scenario it means telling someone they have ASD when they don't.

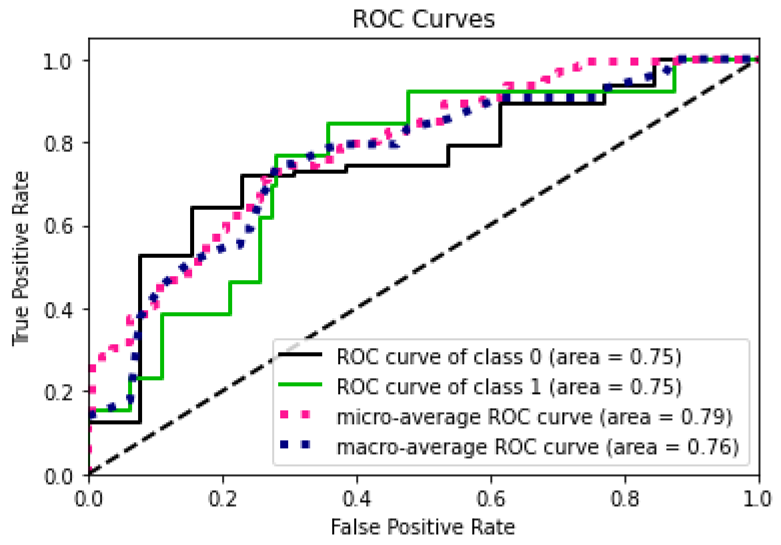


Metrics:

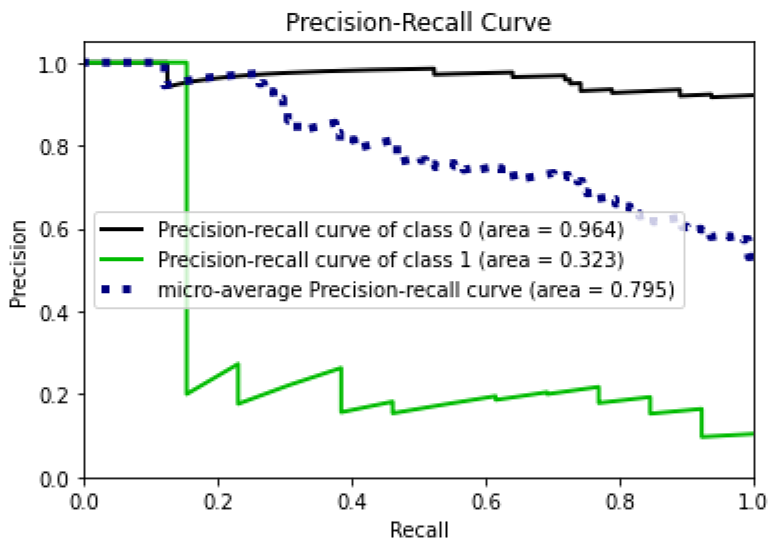
As mentioned earlier we can see a drop in our precision for the positive class with a similar drop in recall of our negative class. With this however, the algorithm could be much better than the others at finding positive classes given its recall score. Additionally, these two facts bleed into the f1-score, the harmonic mean of precision and recall.

```
              precision    recall  f1-score   support

           0       0.95      0.73      0.83       128
           1       0.19      0.62      0.29        13

    accuracy                           0.72       141
   macro avg       0.57      0.67      0.56       141
weighted avg       0.88      0.72      0.78       141
```

ROC and Precision-Recall Curves:



Looking at ROC we can see we have a more spread out curve that does not jump immediately but does take a good jump around FPR .25. Additionally, it looks like the AUC of .75 was only beaten by Random Forests. This shouldn't happen as XGBoost should have more uncorrelated trees with better regularization but this data set is unique and there is never one algorithm that dominates all others in every aspect.



Moving to the Precision-Recall Curve we can see we have a very high precision and recall at a low probability threshold for class one. Additionally, the Precision recall curve for the negative3 class sits at a about the same point and only decreases slightly as recall increases.