AdaBoost Classification Experiment Report

Manikanta Kalyan Gokavarapu
Department of Computer Science and Engineering
University at Buffalo, Buffalo, NY 14260
mgokavar@buffalo.edu

1 AdaBoost Classification Experiment Report

AdaBoost is one of the Ensemble techniques. Ensemble means combining various models and these models are utilized to train the data and get a needed output. In ensemble techniques we have two types one is bagging and other one is boosting. AdaBoost is one of the techniques that uses boosting concept. In boosting techniques, A group of weak classifiers are combined to produce a strong classifier. Strong classifiers have error rates that are nearly zero. Boosting algorithms can track the model which made an inaccurate prediction, and these algorithms are less affected by the overfitting problem. AdaBoost is an iterative ensemble method. AdaBoost classifier combines multiple poorly performing classifier to build a strong classifier which achieves high accuracy. Generally, AdaBoost is commonly used in combination with decision trees. Even though random forest also uses decision trees there is a direct difference between random forest and AdaBoost algorithms i.e., In random forest algorithm some trees might be bigger than others, but there is no predetermined maximum depth. In contrast, in a forest of trees made with AdaBoost, the trees are usually just a node and two leaves. These trees are called "Stumps". So, AdaBoost can be called as 'forest of stumps' rather than trees. Stumps are not great at making accurate classifications. This is because for example if we take a sample of features to classify a target variable then a full-size decision tree would take advantage of all the available input features but whereas a stump only uses one feature to make decision. So, stumps are technically weak learners. So that is the reason why we combine these stumps in AdaBoost algorithm. There is one more key difference between AdaBoost and random forest i.e., each tree in random forest has an equal vote on the final classification. In contrast, in a forest of stumps made with AdaBoost, some stumps get more say in the final classification than others. In a forest of stumps made with AdaBoost, order is important. The error that the previous stump makes influence how the latter stump is made. In total the three ideas behind AdaBoost are as follows. 1. AdaBoost combines a lot of "weak learners" to make classifications. The weak learners are almost always stumps. 2. Some stumps get more say in classification than others. 3. Each stump is made by taking the previous stump's mistakes into account. The forest of stumps using AdaBoost are created as follows. Suppose if we have a dataset with f features and n samples, The first thing we do is give each sample a weight that indicates how important it is to be correctly classified. Initially all the samples have equal weight i.e., 1/total number of samples = 1/n indicating all the samples are equally important. To make the first stump in the forest we need to find the feature that does the best job classifying the samples. We start by seeing how well each input feature classifies the samples w.r.t target variable. So, stumps will be created for each input feature. The stump with lowest 'Gini' index will be the first stump in the forest. We determine how much say a stump has in the final classification based on how well it classified the samples. To evaluate this, we find the total error of a stump which is the sum of weights associated with the incorrectly classified samples. As the sample weights add up to 1, Total error will always be between 0, for a perfect stump, and 1, for a horrible stump. We use the total error to determine Amount of Say this stump has in the final classification with the following formula.

$$Optimal\ weight\ or\ Amount\ of\ Say\ (performance) = \frac{1}{2}\log\left(\frac{1 - Total\ Error}{Total\ Error}\right)$$

When a stump performs well, and the total error is small then the amount of say is a large positive value. If half of the samples are correctly classified and half are incorrectly classified by a stump, then the total error becomes 0.5 and the amount of say is zero. If the stump performs bad and total error is close to 1 the amount of say will be a large negative value. We can calculate the performance of each stump using the above formula and this is

how the sample weights for the incorrectly classified samples are used to determine the Amount of say each stump gets. We should also modify the weights so that the next stump will take the errors that the current stump made into account. To do this we increase the sample weight for incorrectly classified sample and decreasing the other sample weights. We use below formula to increase the incorrectly classified sample weight.

$$New\ Sample\ weight = sample\ weight * e^{amount\ of\ say}$$

To decrease the sample weights of correctly classified samples we use the below formula.

$$New\ Sample\ weight = sample\ weight * e^{-amount\ of\ say}$$

After finding the new sample weights for each sample we normalize the weights. So, if we add all the normalized sample weights, we get one. So, these final normalized weights are used as sample weights to make the latter stump in the forest. Weighted Gini Indexes are calculated to determine which variable should split the next stump. The weighted Gini index puts more emphasis on correctly classifying the misclassified sample in the last stump because the sample has the largest sample weight. To make the classification in AdaBoost we make use of the created stumps and their amount of say values. If we assume each stump produces a classification either 1 or 0 now we find sum of the amount of say values of all the stumps that produces 1 and similarly for 0. The greater sum category will be given as the final classification.

2 Data Set Description and Analysis.

For the Experiment purposes I have taken a dataset which contain 14 attributes. The attributes data is as follows.

- Age - age in years
- Sex -1=male, 0=female
- Cp - chest pain type
- Trestbps - resting blood pressure (in mm Hg on admission to the hospital)
- Chol - serum cholesterol in mg/dl
- Fbs - (fasting blood sugar > 120 mg/dl) (1=true; 0=false)
- Restecg – resting electrocardiographic results
- Thalach – maximum heart rate achieved.
- Exang – exercise induced angina (1=Yes; 0=No)
- Oldpeak – ST depression induced by exercise
- Slope – the slope of the peak exercise ST segment
- Ca – number of major vessels (0-3) colored by flourosopy
- Thal – 3 = normal; 6= fixed defect; 7= reversable defect
- Target - have disease or not (1=yes, 0=no)

The first five rows of the dataset are given in the figure [3.1]

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Figure [3.1]

The attribute information such as count, mean, std, min of the dataset are given in the figure [3.2].

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000

thal	target
303.000000	303.000000
2.313531	0.544554
0.612277	0.498835
0.000000	0.000000
2.000000	0.000000
2.000000	1.000000
3.000000	1.000000
3.000000	1.000000

Figure [3.2]

So, As I want to build a AdaBoost classification model to classify the people who will have heart disease or not (target) using some input features. First, I need to choose the input parameters or features to classify. Based on the correlation matrix output I have chosen all the features as all the features are co-related with the target attribute to classify whether a person has heart disease or not. The correlation matrix output is given the below figure [3.3]

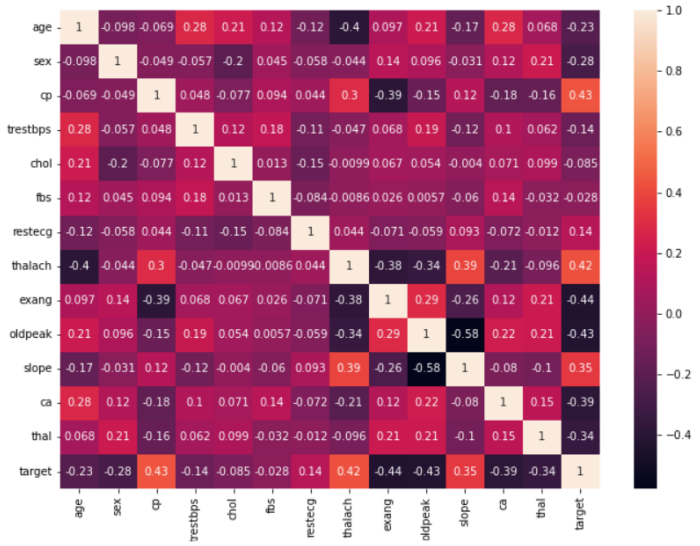

Figure [3.3]

I have plotted the box plots for some highly correlated features like chest pain, thalach (maximum heart rate achieved), Restecg (resting electro cardiography results), slope to understand the relation between input features and target variables better. The plot is given in the figure 3.4.

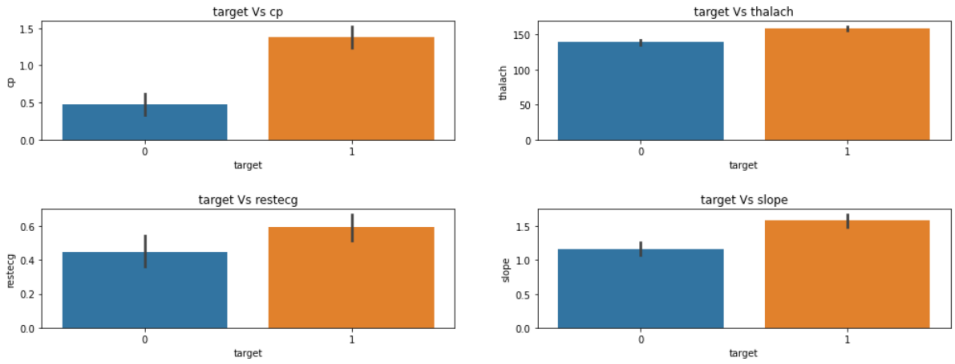

Figure [3.4]

The pair plots of all the features are plotted in below figure 3.5. From the pair plots we can infer that patients who are most likely to not suffer from the disease have a slightly greater blood pressure than the patients who have heart diseases.

Figure [3.5]

3 Experiment and Results

I have considered variable Y which holds the target variable (Having heart disease or not) and variable x which holds all the features except the target variable. I have selected all features because all the features are affecting the target variable either positively or negatively which we can observe in the data analysis above. I have split the data into 80% as train data and rest 20% as test data. I have given n_estimators parameter as 24 because the AdaBoost model has highest accuracy at this value. We can find the same below in the figure 3.6.

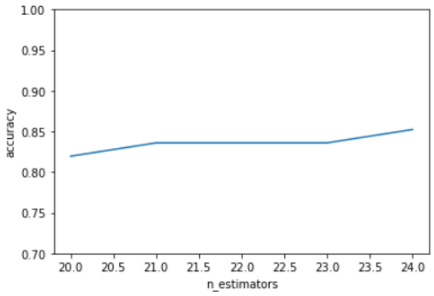

Figure [3.6]

The test results are in below figure [3.7] plotted via confusion matrix and from the confusion matrix we can infer that 23 patients were predicted that they will not have heart disease, the prediction was correct (True-negative). 29 patients were predicted that they will have heart disease, the prediction was correct (True-positive). 5 patients were predicted that they will have heart disease, but the prediction was wrong (False-positive). 4 patients were predicted that they will not have heart disease, but the prediction was wrong (False-Negative). As False-Negative and False-Positive values are very less our model is performing good and has good accuracy of **85.24**%.

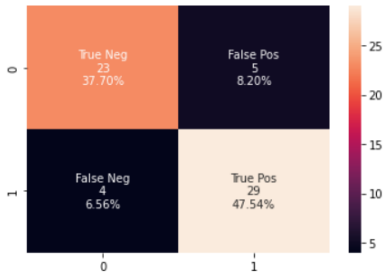

Figure [3.7]

The overall accuracy or classifier score of the model is **85.24%.** And remaining results of model like precision score, roc_auc score, recall score, F1 score are given in the below figure [3.8].

	precision	recall	f1-score	support
0	0.85	0.82	0.84	28
1	0.85	0.88	0.87	33
accuracy			0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.85	0.85	0.85	61

Figure [3.8]

References

[1] Charan kakararaparthi, Heart_Diseases, Prediction of Heart Disease by Effective Machine Learning techniques.

[2] Chengyou Chen, Ensemble Learning, Lecture Slides, 10/11/2022.