North Carolina State University

ECE 763 Computer Vision

Project 02 Report:

Adaptive Boosting (AdaBoost) for Face Detection

Author: Adarsh Puri (200324466)

Instructor: Dr. Tianfu Wu

Objective :

- Adaptive Boosting (AdaBoost) for Face Detection
- Plot Roc Curve
- Compute Accuracy, Error rate, False positive Rate and False negative Rate, True positive Rate and True Negative Rates.

Code: TP, TN, FP, FN, variables stands for True Positive, True Negative, False Positive, False Negative.

A total of 4094 Haar features extracted from one image patch.

TOTAL:

Positive Samples =	**Case 1. & 2:** 100	**Case 3 & 4:** 2000
Negative Samples =	**Case 1. & 2:** 100	**Case 3 & 4:** 1000

Training Images:

Positive Samples (Face) =	**Case 1. & 2:** 85	**Case 3 & 4:** 1700
Negative Samples (Non Face) =	**Case 1. & 2:** 85	**Case 3 & 4:** 850

Test Images:

Positive Samples (Face) =	**Case 1. & 2:** 15	**Case 3 & 4:** 300
Negative Samples (Non Face) =	**Case 1. & 2:** 15	**Case 3 & 4:** 150

HAAR FEATURE AND WEAK CLASSIFIER:

The Cascade of Viola-Jones classifiers consists of many weak classifiers, each one of them only has to be better than a random guess, error should be less than 0.5. The weak classifier is the

Haar-feature, it's location in the subwindow, size of the Haar feature and the threshold which says what is face (Positives) and what is not Face(Negatives). Figure 1 shows what Haar features were used in this project. The score of the feature for each sample is computed by subtracting pixel values under the black region of the Haar feature from the sum of the pixel values under the white region. This is done by using integral image, which allows computing the sum of a block with up to 4 array references.

With a good "weak" classifier, it turned out that the data can be separated quite well already. However, we see from the output that there are quite a lot of false negatives and some false positives. In order to improve, we combine many weak classifiers around 50 to 100 and perform a weighted majority voting by all of them and then we decide what is face and what is not face.

The labels in our framework are "1" for positive samples and "-1" for negative samples, and not 1-0.

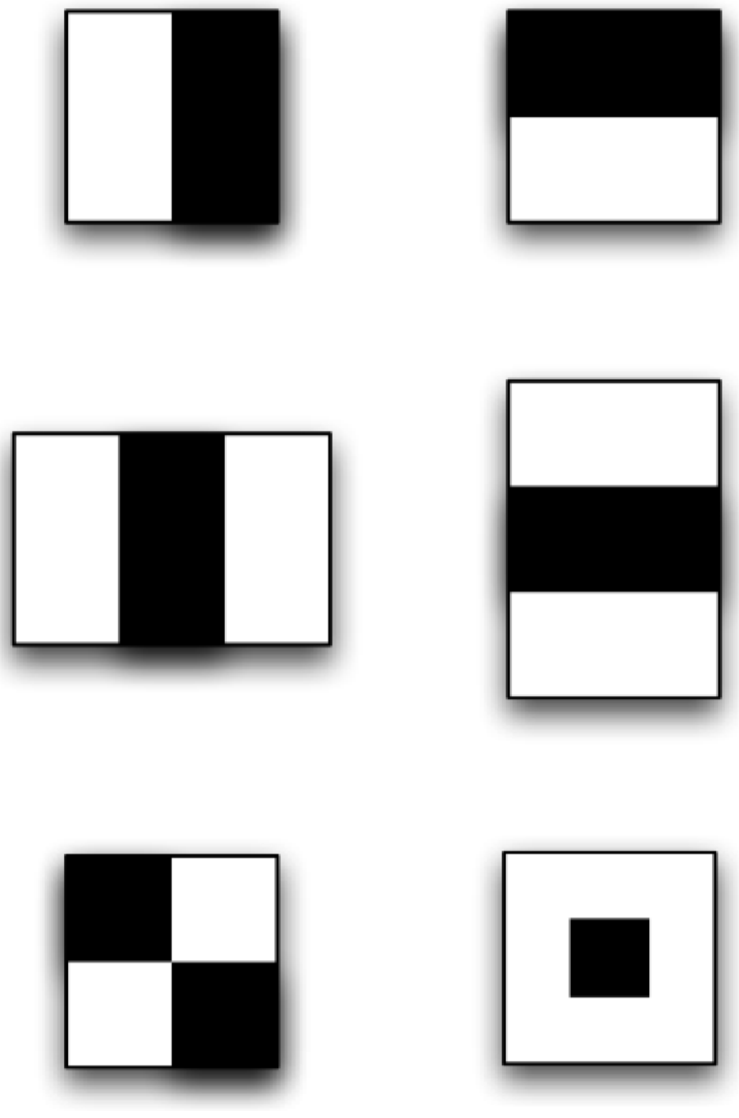
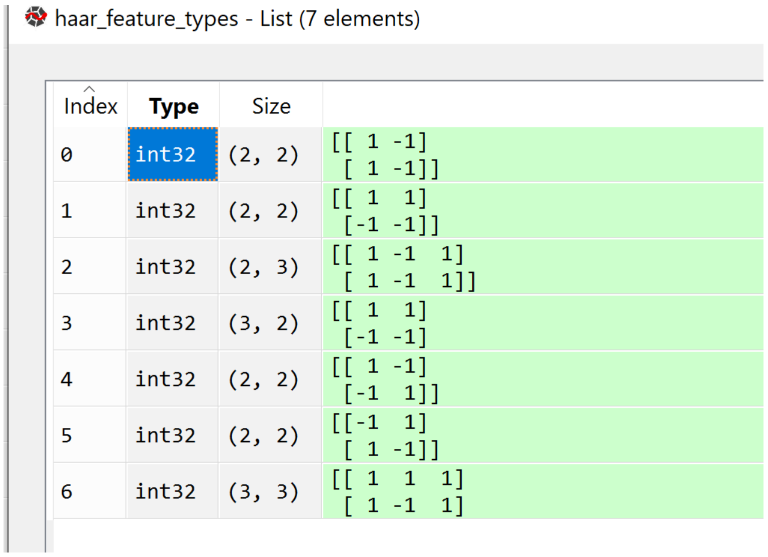
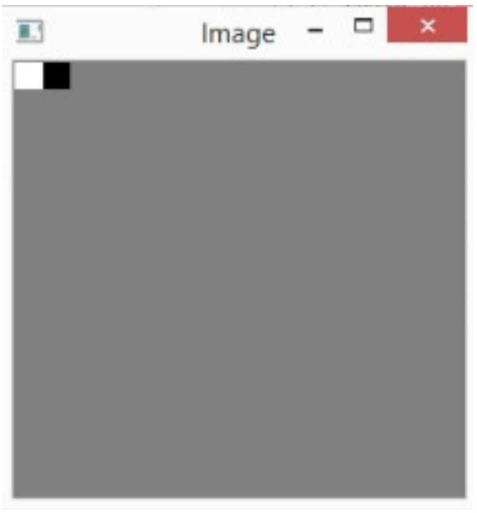


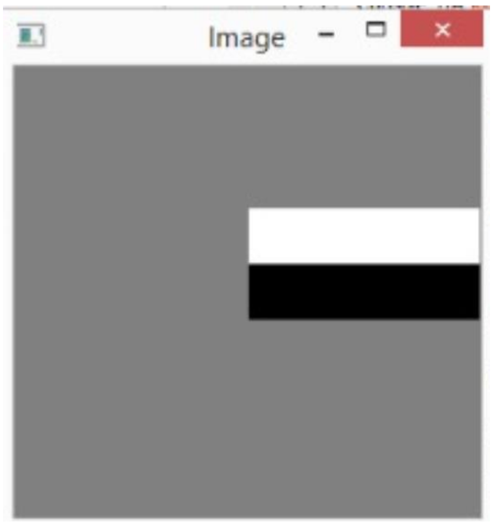
Figure 1: Haar features used for weak classifiers.

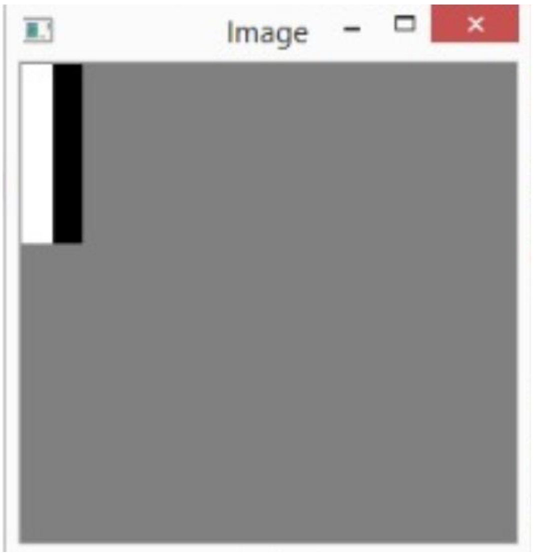


haar_feature_types - List (7 elements)

Index	Type	Size	
0	int32	(2, 2)	[[1 -1] [1 -1]]
1	int32	(2, 2)	[[1 1] [-1 -1]]
2	int32	(2, 3)	[[1 -1 1] [1 -1 1]]
3	int32	(3, 2)	[[1 1] [-1 -1]
4	int32	(2, 2)	[[1 -1] [-1 1]]
5	int32	(2, 2)	[[-1 1] [1 -1]]
6	int32	(3, 3)	[[1 1 1] [1 -1 1]







CASCADE OF WEAK CLASSIFIERS:

The result of the Adaboost, the strongest of weak classifiers, is used in the final algorithm in the form of a cascade. This means that after applying few strongest weak classifiers as a first stage of the cascade, many negative windows can be eliminated already and there is no need to apply more number of weak classifiers (stages two, three etc.) to those windows. This approach will speed up the detection process.

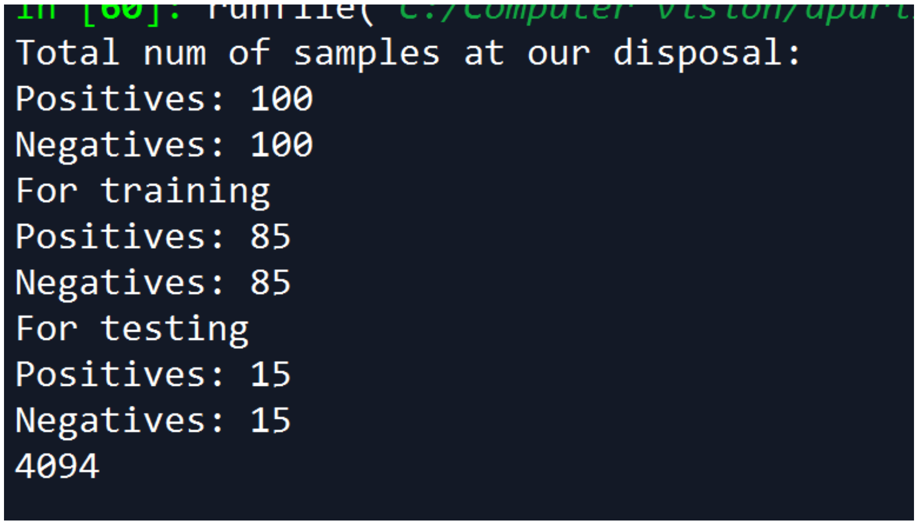
There are some additional features in the code, such as visualizing how the data is separated by every individual weak classifiers or visualizing how the cascade evolves (separates data better and better) as the new weak classifier is added to the cascade.

The Variable in Code named as '**Cascade**' consists final weak classifiers used for One strong classifier. We can view the final weak classifier along with haar feature type, size, shape, start position.

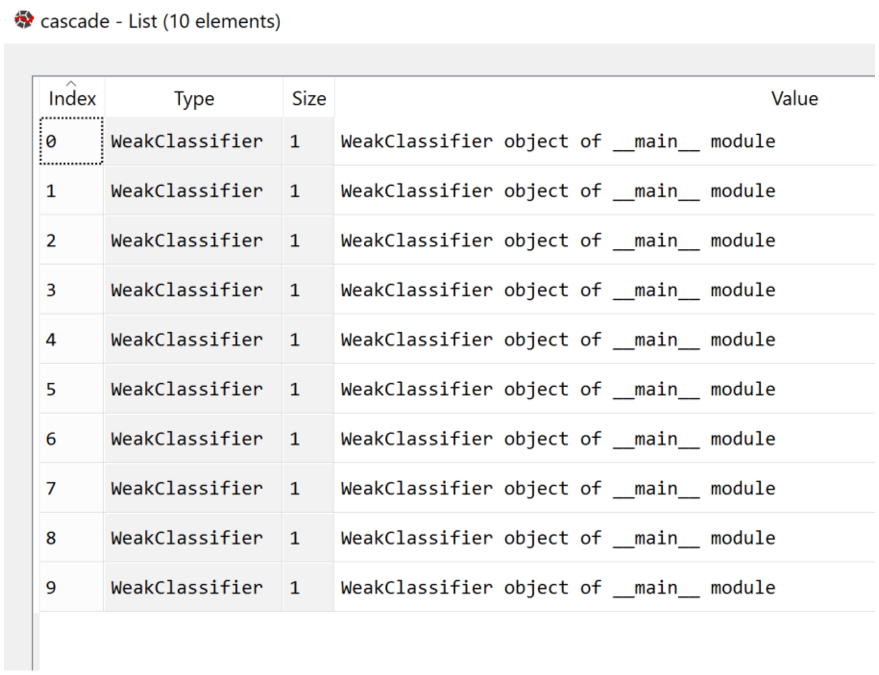
Outputs:

Case 1 & 2: Trained and tested for small number of training and testing data as shown.

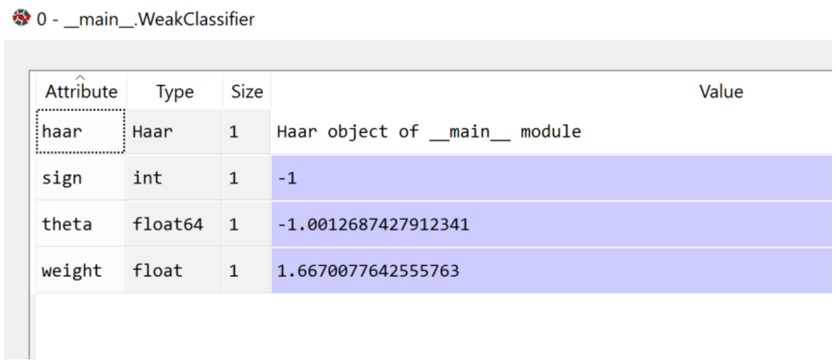
Final Weak Classifier that make up strong Classifier = 10



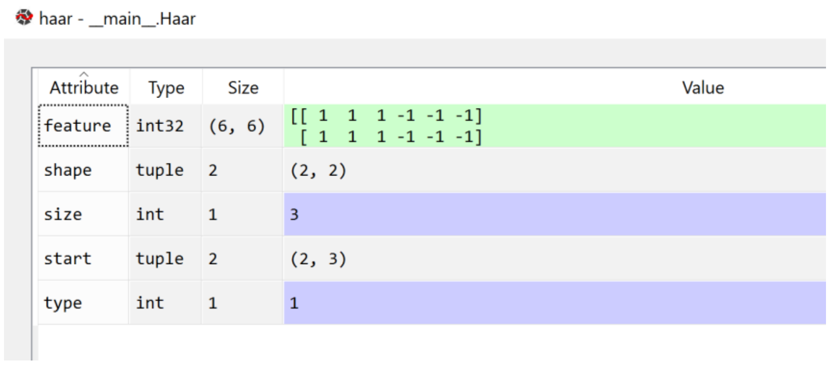
```
In [60]: runfile('C:/Computer Vision/apurt
Total num of samples at our disposal:
Positives: 100
Negatives: 100
For training
Positives: 85
Negatives: 85
For testing
Positives: 15
Negatives: 15
4094
```


cascade - List (10 elements)

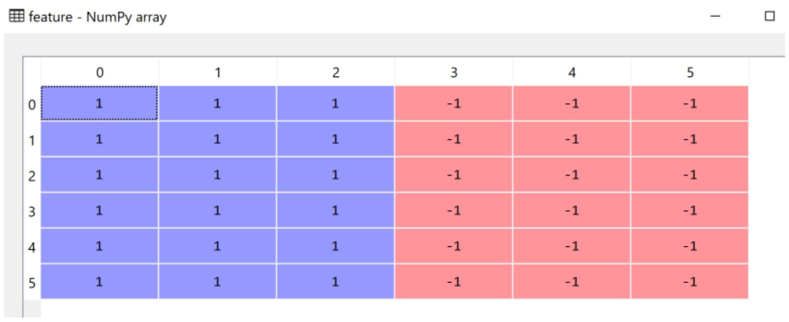
Index	Type	Size	Value
0	WeakClassifier	1	WeakClassifier object of __main__ module
1	WeakClassifier	1	WeakClassifier object of __main__ module
2	WeakClassifier	1	WeakClassifier object of __main__ module
3	WeakClassifier	1	WeakClassifier object of __main__ module
4	WeakClassifier	1	WeakClassifier object of __main__ module
5	WeakClassifier	1	WeakClassifier object of __main__ module
6	WeakClassifier	1	WeakClassifier object of __main__ module
7	WeakClassifier	1	WeakClassifier object of __main__ module
8	WeakClassifier	1	WeakClassifier object of __main__ module
9	WeakClassifier	1	WeakClassifier object of __main__ module


0 - __main__.WeakClassifier

Attribute	Type	Size	Value
haar	Haar	1	Haar object of __main__ module
sign	int	1	-1
theta	float64	1	-1.0012687427912341
weight	float	1	1.6670077642555763

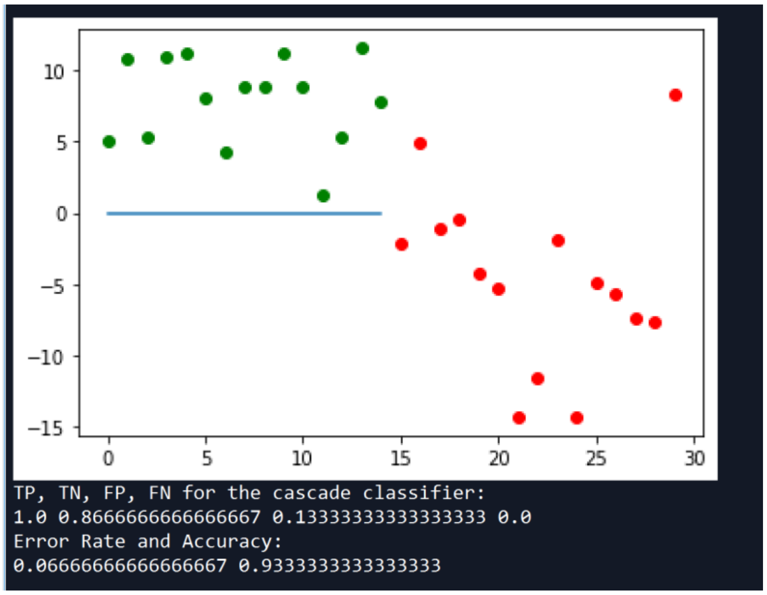

haar - __main__.Haar

Attribute	Type	Size	Value
feature	int32	(6, 6)	[[1 1 1 -1 -1 -1] [1 1 1 -1 -1 -1]
shape	tuple	2	(2, 2)
size	int	1	3
start	tuple	2	(2, 3)
type	int	1	1

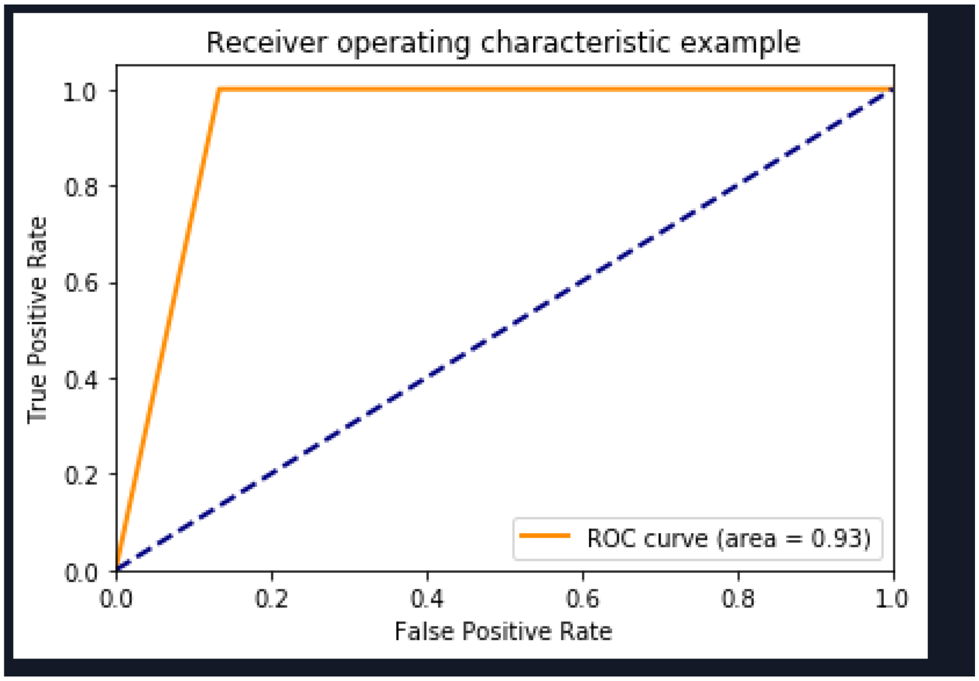


feature - NumPy array

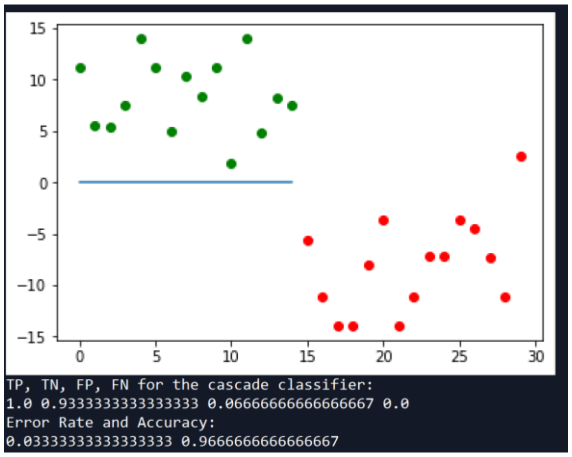
	0	1	2	3	4	5
0	1	1	1	-1	-1	-1
1	1	1	1	-1	-1	-1
2	1	1	1	-1	-1	-1
3	1	1	1	-1	-1	-1
4	1	1	1	-1	-1	-1
5	1	1	1	-1	-1	-1



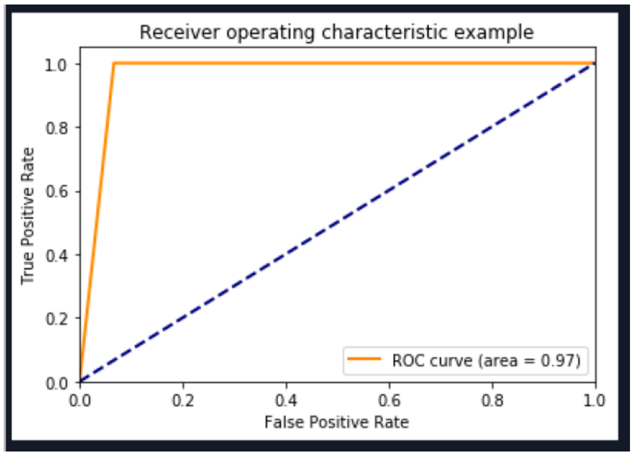
```
TP, TN, FP, FN for the cascade classifier:
1.0 0.8666666666666667 0.13333333333333333 0.0
Error Rate and Accuracy:
0.06666666666666667 0.9333333333333333
```



Again, ran the above Case for same number of test and training Cases but different set of data as data set are shuffled.



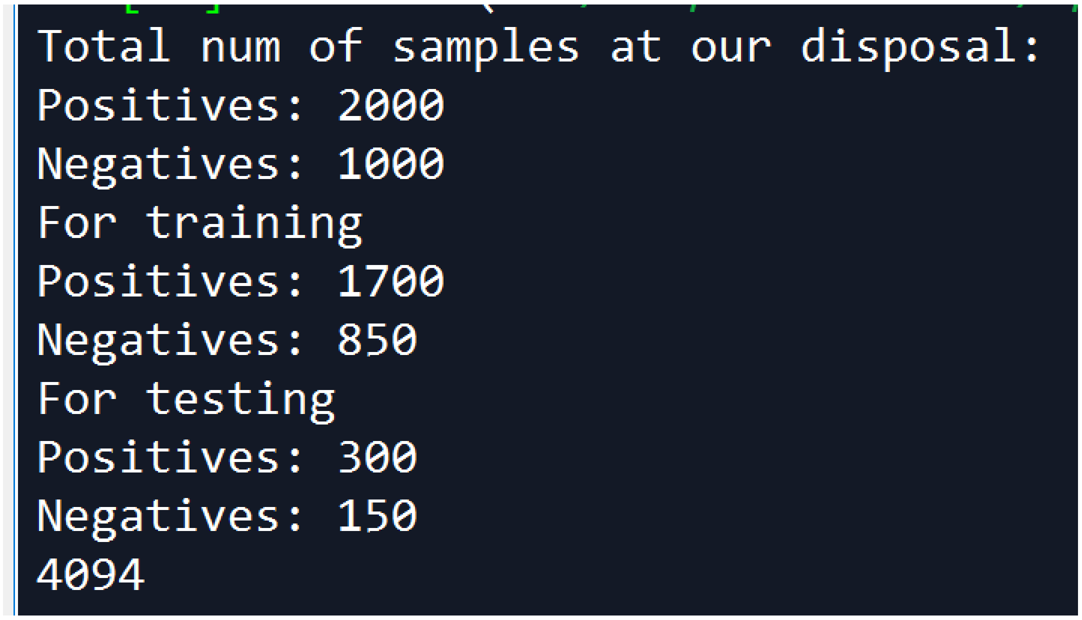
```
TP, TN, FP, FN for the cascade classifier:
1.0 0.9333333333333333 0.06666666666666667 0.0
Error Rate and Accuracy:
0.03333333333333333 0.9666666666666667
```



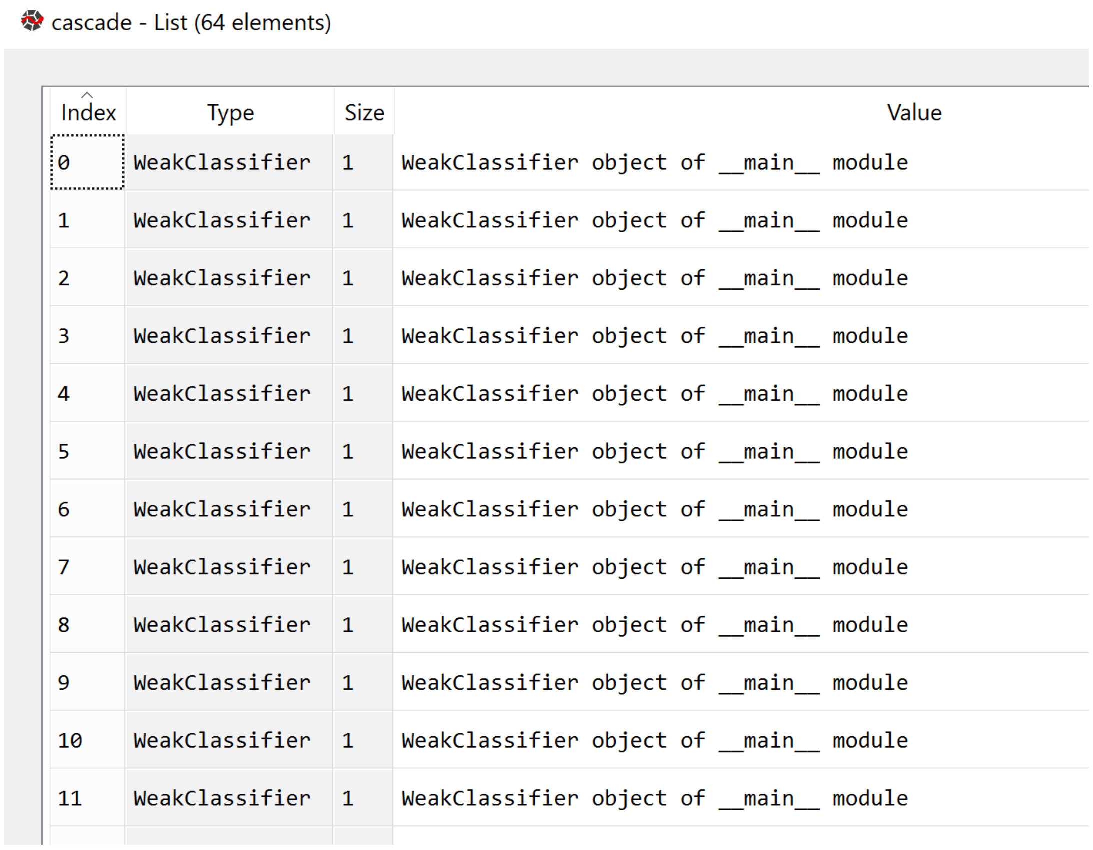
As the number of Training and Test Cases are increased, will increase the Weak Classifiers and hence the Performance improved as shown in testing output below:

Case 3 & 4: Trained and tested for Large number of training and testing data as shown, improved the ROC curve (area) and Accuracy.

Final Weak Classifier that make up strong Classifier = 64

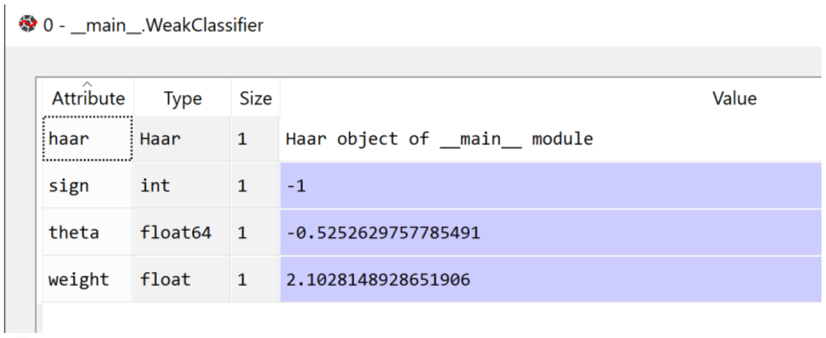


```
Total num of samples at our disposal:
Positives: 2000
Negatives: 1000
For training
Positives: 1700
Negatives: 850
For testing
Positives: 300
Negatives: 150
4094
```



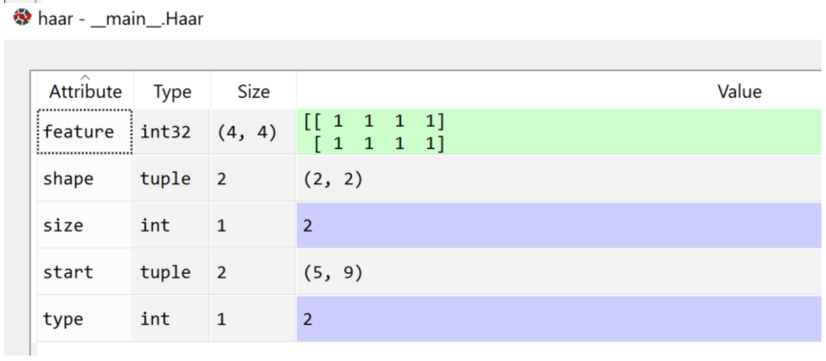
cascade - List (64 elements)

Index	Type	Size	Value
0	WeakClassifier	1	WeakClassifier object of __main__ module
1	WeakClassifier	1	WeakClassifier object of __main__ module
2	WeakClassifier	1	WeakClassifier object of __main__ module
3	WeakClassifier	1	WeakClassifier object of __main__ module
4	WeakClassifier	1	WeakClassifier object of __main__ module
5	WeakClassifier	1	WeakClassifier object of __main__ module
6	WeakClassifier	1	WeakClassifier object of __main__ module
7	WeakClassifier	1	WeakClassifier object of __main__ module
8	WeakClassifier	1	WeakClassifier object of __main__ module
9	WeakClassifier	1	WeakClassifier object of __main__ module
10	WeakClassifier	1	WeakClassifier object of __main__ module
11	WeakClassifier	1	WeakClassifier object of __main__ module



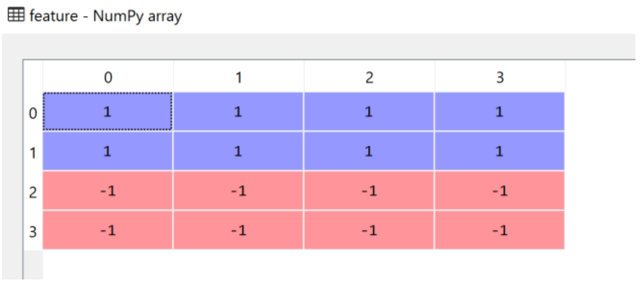
0 - __main__.WeakClassifier

Attribute	Type	Size	Value
haar	Haar	1	Haar object of __main__ module
sign	int	1	-1
theta	float64	1	-0.5252629757785491
weight	float	1	2.1028148928651906



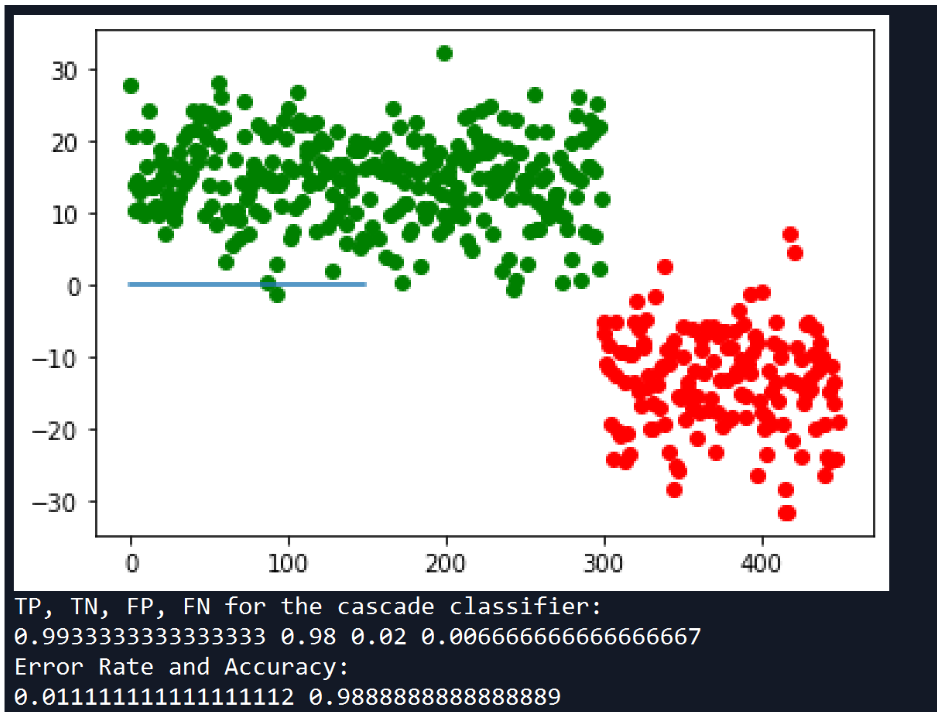
haar - __main__.Haar

Attribute	Type	Size	Value
feature	int32	(4, 4)	[[1 1 1 1] [1 1 1 1]
shape	tuple	2	(2, 2)
size	int	1	2
start	tuple	2	(5, 9)
type	int	1	2

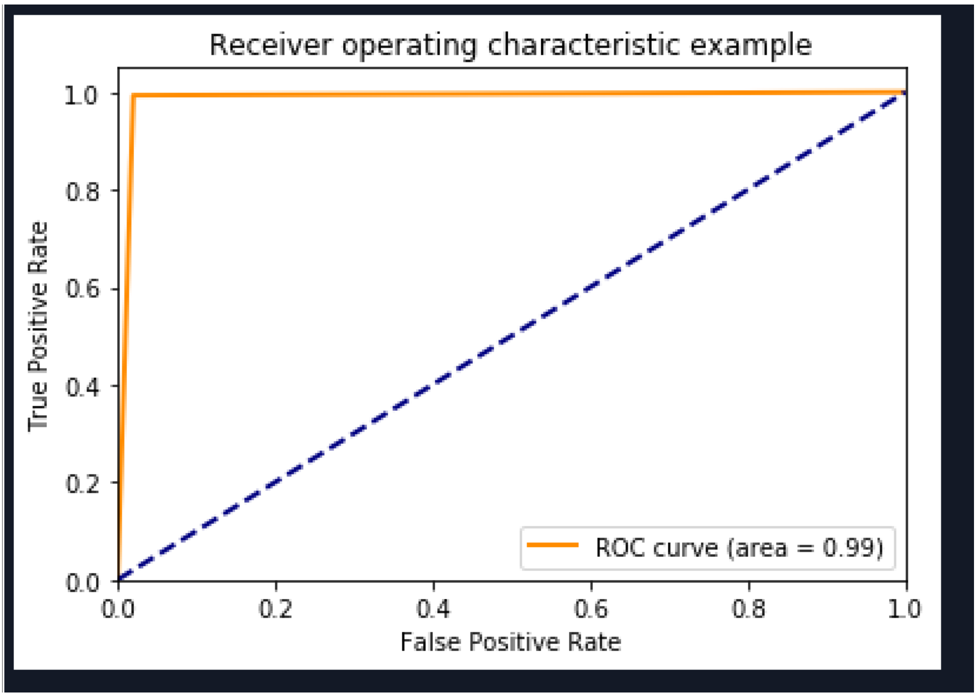


feature - NumPy array

	0	1	2	3
0	1	1	1	1
1	1	1	1	1
2	-1	-1	-1	-1
3	-1	-1	-1	-1



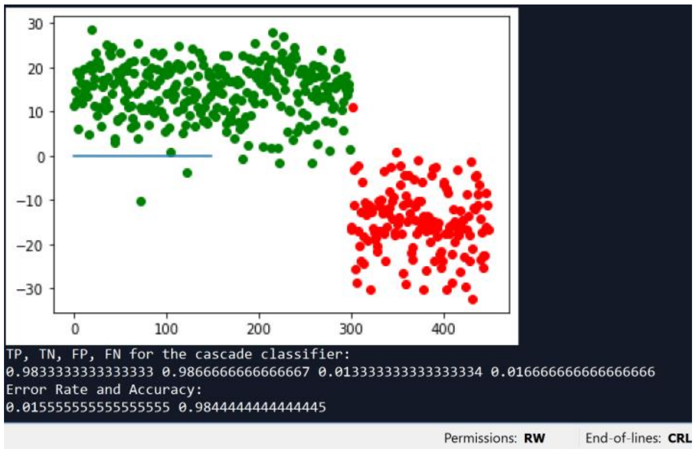
```
TP, TN, FP, FN for the cascade classifier:
0.9933333333333333 0.98 0.02 0.006666666666666667
Error Rate and Accuracy:
0.011111111111111112 0.9888888888888889
```



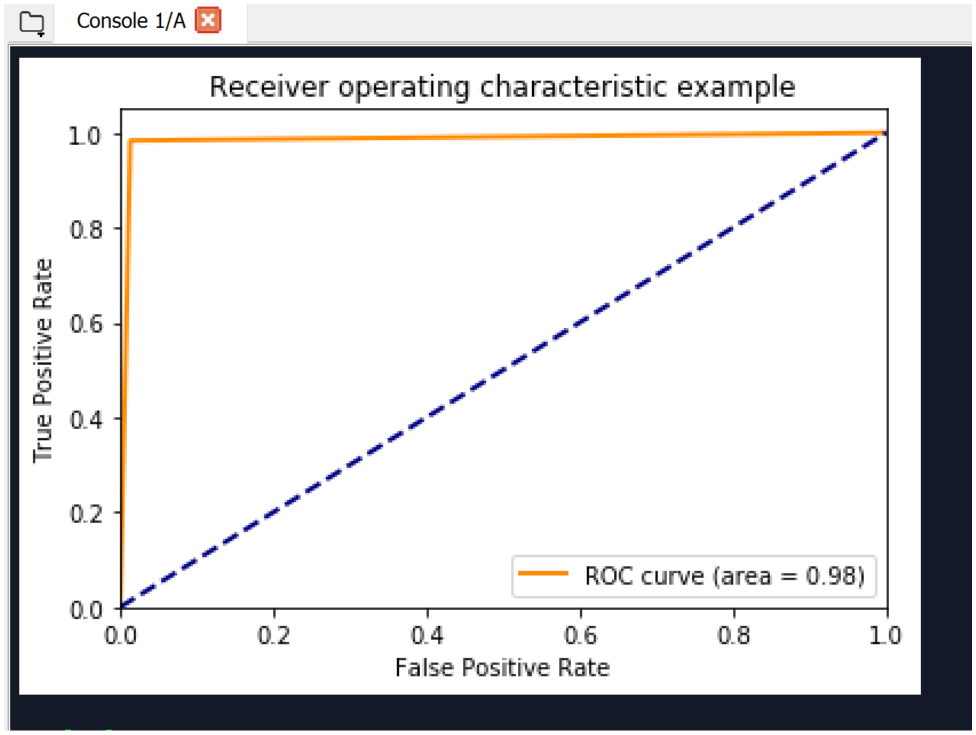
The accuracy as seen from ROC Curve and its area, also seen from the true positive rates came in the range of 95 percent to 99 percent.

3. Again, ran the above Case for same number of test and training Cases but different set of data (data set are shuffled before to increase randomness of data) to validate ROC performance.

Cascade Classifier size = 62



```
TP, TN, FP, FN for the cascade classifier:
0.9833333333333333 0.9866666666666667 0.013333333333333334 0.016666666666666666
Error Rate and Accuracy:
0.015555555555555555 0.9844444444444445
```



References:

1. Adaptive Boosting (AdaBoost) for Face Detection Project 02 pdf
2. P.Viola, M.Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features", CVPR 2001.
3. https://en.wikipedia.org/wiki/Haar-like_feature
4. https://github.com/anastasiabolotnikova/adaboost/blob/master/adaboost_report.pdf
5. https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html
6. https://en.wikipedia.org/wiki/Summed-area_table