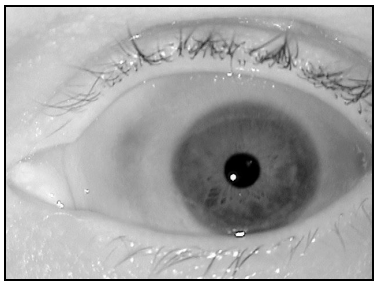
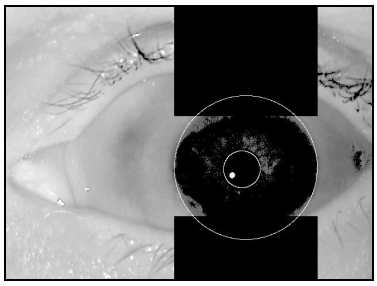
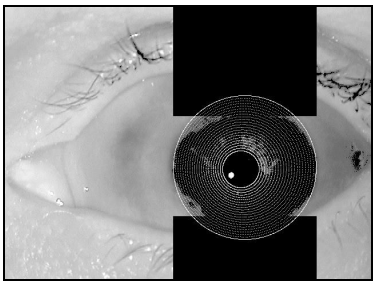
Jeremy Midvidy, jam658
EECS 395: Biometrics, Fall 2018
Iris Analyzer Report

Approach:

First, I downloaded the data set. Then, in Python, I gathered 2 left-eye images and 2 right-eye images for each folder in each of the three data sets (using the data in the .txt file in each folder). Within each folder, I put each of these two images in a subfolder called “left_eyes” and “right_eyes” respectively.

Then, in Matlab, I iterated through each folder in the data set. In each folder, I went into each “left_eyes” and “right_eyes,” and called “createiristemplate.m” on each image in each folder. This created the “-polar.jpg” and “-polarnoise.jpg” needed for each image that I would need later to perform testing and classification.

Now, I had images of that can be structured like this:

Input	**Noise**	**Normal**
		

Polar	**Polar-Noise**
	

Then, for testing, I tested each probe set (two total). For each folder in probe, I calculated the hamming distance between each 2 left_eyes and each right_eyes with each 2 left_eyes and 2 right_eyes in the testing set. For example, if probe was folders: “ABCDE,” and gallery was folders, “HIJKL”, then each test:

AH, AI, AJ, AK, AL, BH, BI, BJ, BK, BL, CH, CI, CJ, CK, CL etc…

Within each test, like “AH,” my script tested each of A’s 2 left eyes against H’s 2 left eyes, and A’s 2 right eyes against H’s 2 right eyes. That’s 4 tests per eye and 8 tests total. Then, I returned hamming distance was the min(min(left), min(right)). This is how I constructed my classification.

For all output, I found a threshold value by averaging the iris score for each value that was supposed to be accept claims. For example, test folder, ‘20463’ was also in the gallery, so this test should be accepted. The similarity score for ‘20463’ in test against ‘20463’ in train was

`0.0807`. I then got the similarity score for every claim that should be accepted (e.g. the similarity score for every folder in test against itself in train), and then averaged the result.

Then, after finding the thresholds, I iterated through the similarity scores. For each score, if it was below the threshold, I classified it as accept. If above the threshold, I classified it as reject. I then updated my counters, `[true_accept, true_reject, false_accept, false_reject]` as follows:

IF:

A true claim was accepted → `true_accept += 1`
A false claim was accepted → `false_accept += 1`
A true claim was rejected → `false_reject += 1`
A false claim was rejected → `true_reject += 1`

Results:

Test1

Gallery:	LG2200-2008-03-11_13
Probe:	LG2200-2010-04-27_29

Full Data can be found in the csv file, "`test1_results.csv`" which is stored in the same directory as this report.

First, the learned threshold was:

True Claim Values:
[0.0807, 0.1754, 0.1488, 0.1893, 0.0238, 0.2424, 0.0533, 0.2, 0.1905, 0.1122, 0.075, 0.0682]

With the average of the true claim values being the threshold:

Threshold:	0.1299666666666667

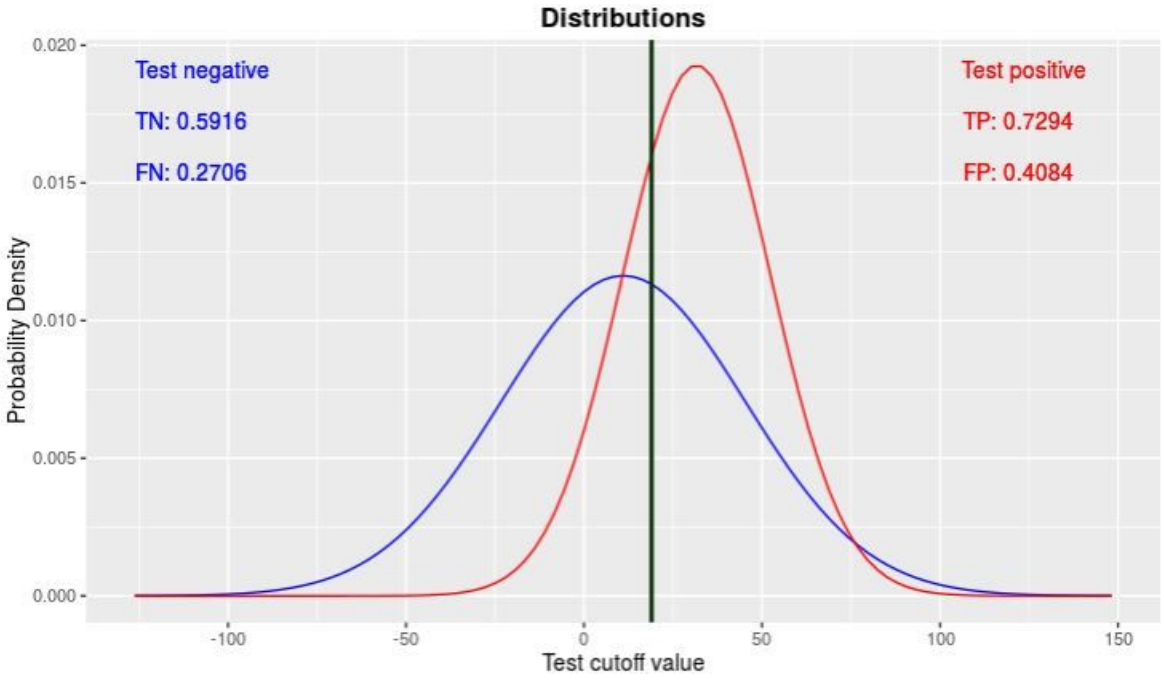
Then, after learning the threshold, I iterated through the data and found the values for the four counters:

Category	**Absolute Number**	**Rate**
False reject	13	13/19 = 68%
False accept	3661	3661/18901 = 19.4%
True accept	6	6/19 = 31.6%
True reject	14860	14860/18901 = 78.6%

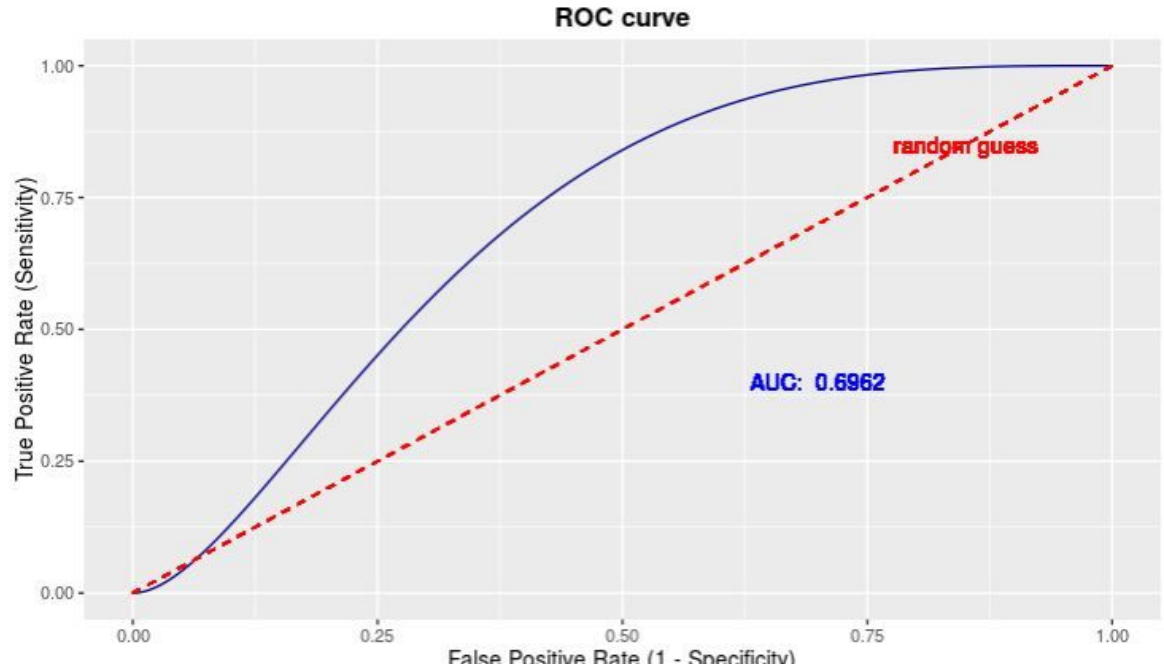
Since there were 88 training files and 215 testing files, that is 88*215 = 18,920 total iris scores. Of these, 19 represent true claims and 18,901 represent false claims. Thus false accept and true reject rates equal their numbers over 18,901 (total false claims), and false reject and true accept rates equal their numbers over 19 (total number of true claims).

With this data, I can construct the genuine and imposter distributions: TPR = 31.6%, FPR = 19.4%, cutoff=threshold.

Distributions for: (Probe) *LG2200-2010-04-27_29* against *LG2200-2008-03-11_13* (Gallery)

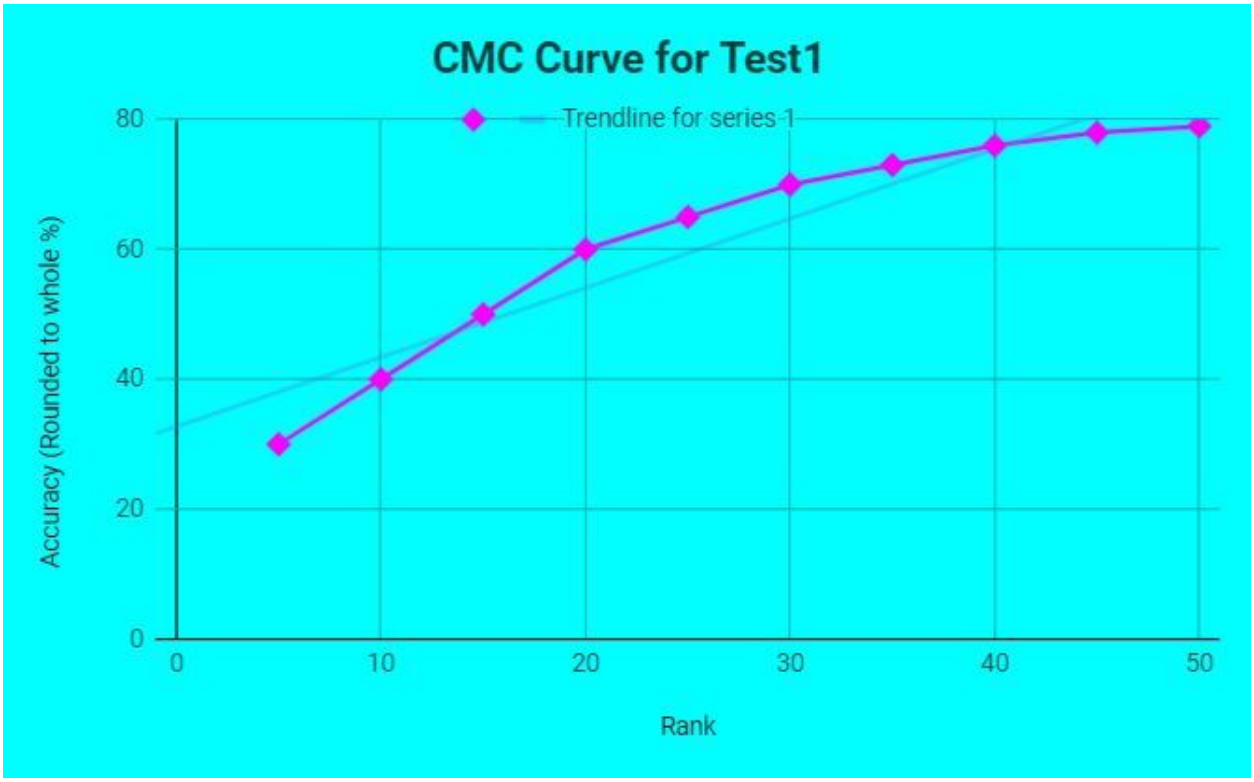


ROC for: (Probe) *LG2200-2010-04-27_29* against *LG2200-2008-03-11_13* (Gallery)



Now, since I know the FRR=68% and TRR=78.6%, I can construct my CMC curves by multiplying the FRR*FPR for every input, and TRR*TPR for every input.

CMC for: (Probe) *LG2200-2010-04-27_29* against *LG2200-2008-03-11_13* (Gallery)



Results for Test2:

Test1

Gallery:	LG2200-2008-03-11_13
Probe:	LG4000-2010-04-27_29

Full Data can be found in the csv file, "`test2_results.csv`" which is stored in the same directory as this report.

First, the learned threshold was:

True Claim Values:
[0.071, 0.2133, 0.1443, 0.1629, 0.0959, 0.2836, 0.08, 0.0078, 0.2273, 0.1975, 0.0571, 0.1711, 0.1748, 0.0385]

With the average of the true claim values being the threshold:

Threshold:	0.13750714285714286

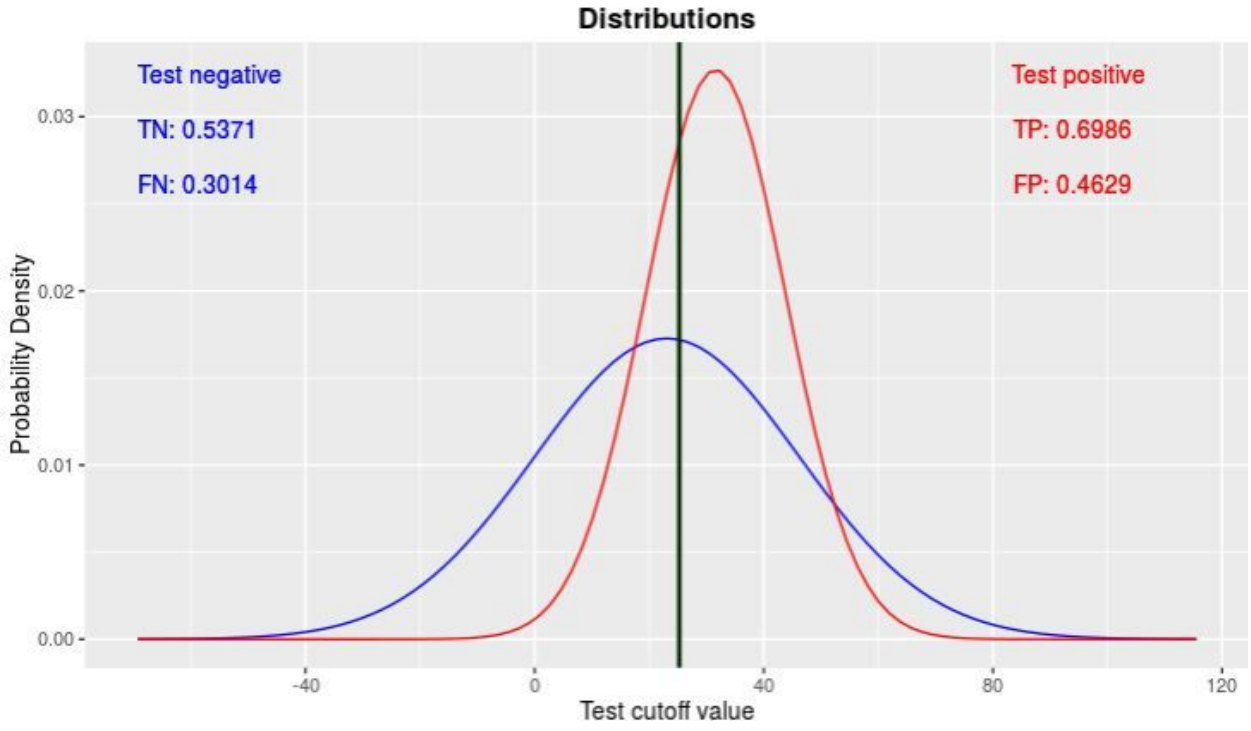
Then, after learning the threshold, I iterated through the data and found the values for the four counters:

Category	**Absolute Number**	**Rate**
False reject	13	13/19 = 68%
False accept	4431	4431/19165 = 23.1%
True accept	6	6/19 = 31.6%
True reject	14298	14298/19165 = 74.6%

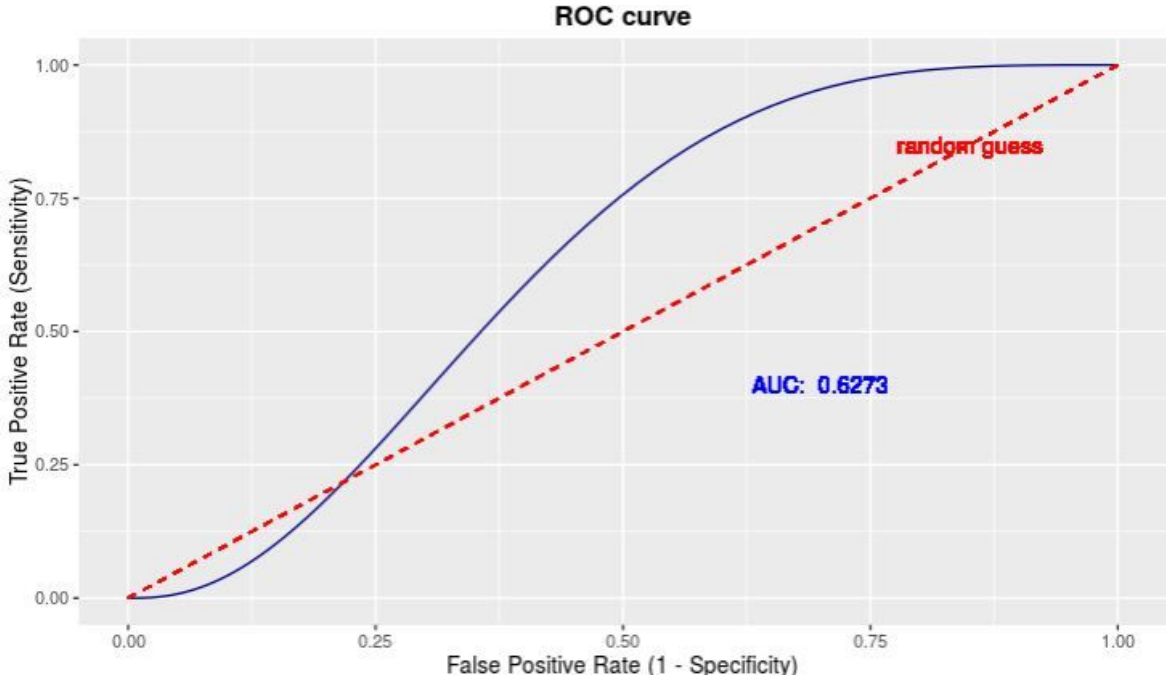
Since there were 88 training files and 218 testing files, that is 88*218 = 19,184 total iris scores. Of these, 19 represent true claims and 19,165 represent false claims. Thus false accept and true reject rates equal their numbers over 19,165 (total false claims), and false reject and true accept rates equal their numbers over 19 (total number of true claims).

With this data, I can construct the genuine and imposter distributions: TPR = 31.6%, FPR = 23.1%, cutoff=threshold.

Distributions for: (Probe) *LG4000-2010-04-27_29* against *LG2200-2008-03-11_13* (Gallery)

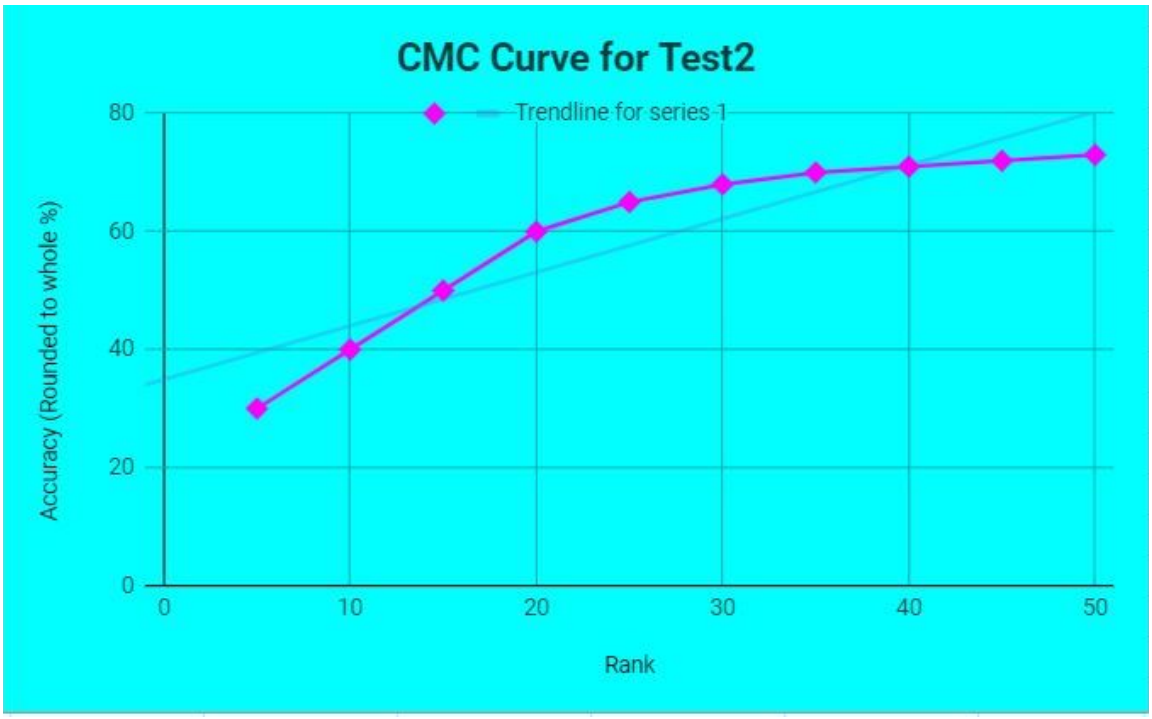


ROC for: (Probe) *LG4000-2010-04-27_29* against *LG2200-2008-03-11_13* (Gallery)



Now, since I know the FRR=68% and TRR=74.6%, I can construct my CMC curves by multiplying the FRR*FPR for every input, and TRR*TPR for every input.

CMC for: (Probe) *LG4000-2010-04-27_29* against *LG2200-2008-03-11_13* (Gallery)



Analysis:

In all, my classifer did not perform very well. In each test, there was a 31.6% true accept rate, which is low. Likewise, the false accept rate was 19.4% and 21.3% for each test case, respectively. Those rates might seem low, but they need to be lower. Accepting 19.4% or 21.3% of all false claims would make this iris classifer very ineffective in the real world.

I think my error comes from the hamming distance function in my python classifier. For some reason, it does not work as intended. I think I should have used the matlab version, but I had not realized that error until it was too late (I would have had to re-make all of the iristemplates).

However, I do think my current classifier is rather precise. For each test, I was able to independently reproduce similiar results, with very close FAR, FRR, TPR, TRR rates. This signals that the error is localized somewhere in my code: if the error is in the hamming distance function, then I should be able to fix the error in one location. Then, this would translate to fixing the entire system as a whole because the test results in each test would get much better.

The ROC curves reflect a classifer that rejects ⅛ of invalid inputs and accepts ⅓ of valid inputs. This is illustrated in the overlapping distributions. In each test, while they overlap, the SD of true accepted claims is lower, leading to a higher hill than the distribution for false accept claims. This illustrates that this portion of my classifer is working correctly: it accepts a broader range of true claims and rejects a narrower range of false claims. However, there are so many false claims that the sheer volume of false claims comes to even out with the accepted true claims. In future versions of my classifer, I will need to address this.

Finally, the CMC curves express how the classifer increases in correct classifications as false claims move closer to the mean. The upward bound of true reject rates in test1 is 78.6% and in test2 is 74.6%. As the values move closer to the average false reject claim, the classifer is more confident in its ability to correctly classify that input as false. This shows how the CMC curve can classify with increasing confidence. Clearly, it's accuracy is not good enough to be considered for comerical grade deployment. But I think my data trends in the correct direction.

Submitted Materials:

- *IrisClassifer.py*
    - This python file will iterate through the data set and make classifications/collect data for the two probes against the one training gallery.
- *LRSplit.py*
    - This python script will take a data set, go through it, and add a "left_eyes" and a "right_eyes" folder to each folder in each data set.
- *Main.m*
    - This matalb script will make an iris template on each of the eyes in "left_eyes" and "right_eyes" in each folder in each data set.
- *Test1_results.csv*
    - The raw data of testing the first probe against the gallery.
- *Test2_results.csv*
    - The raw data of testing the second probe against the gallery.