CMPE 58Z BIOMETRICS

Homework I

İpek Erdoğan

2019700174

Introduction

In this homework, my aim was to process similarity matrices and it's corresponding class labels for 4 different datasets and evaluate the system according to these inputs. I basically label the similarity scores as "genuine" or "impostor" according to the threshold. Then i calculate true positives, false positives, true negatives and false negatives according to the ground truth labels and calculate FAR, FRR and GMR values. I draw the FAR-FRR graph and ROC curve. I also collect genuine and impostor scores in two different lists and plot them as a normalized histogram, to see the score distributions for impostor and genuines.

There are some points I want to clarify about my project:

1) I used class label txt files to supervise my code from outside. I basically checked all of the 4 class label files with such a code to understand the class distribution.

```
#with open('data1_Class_Labels.txt') as infile:
#    counts = collections.Counter(l.strip() for l in infile)
#for line, count in counts.most_common():
#    print(line, count)
```

   And I realized that all of these 4 data has same equal sized classes. There are always 10 samples for a class. That's why in my code, for checking the similarity matrix according "ground truth" labels, I took the number of 10 as a base. First 10 elements in both row and columns, then next 10 elements in row and columns, then next 10 elements etc. If there was a different distribution of sample sizes for classes, I would parse this information from class label files and take these different sample sizes as base numbers. You can check this "base" logic in "metric_calculate" function in my code, between lines 74-90.

2) During the threshold range determination process, I benefited from the score distributions. Trying to start with "1" FAR also leaded me. I tried to select the smallest threshold step for getting the smoothest graphics and values. But the smaller the number you choose, the more time it takes. So for extracting the graphs, I choosed normal step sizes (25 for data1, 0.05 for data2, 0.05 for data3 and 0.02 for data4). But these threshold steps were not enough to extract the accurate FAR points such as 0.1, 0.01, 0.001. Also finding the true EER point was not possible with these step sizes too.

   That's why I calculated these values by myself recursively. I ran the code with step sizes up above and checked for the nearest FAR points to the 0.1,0.01 and 0.001. Also for the ERR, I checked for the threshold ranges where FAR and FRR comes closer. Then, after determining the threshold ranges, I ran the code between these threshold ranges with much smaller threshold steps (such as 0.001). This way, I could find more specific and accurate FAR,FRR and ERR values.

I also implemented find_frr and find_err functions (you can check lines 99 and 114) in my code to calculate these values, but again, you need to run the code with smaller threshold steps to get accurate results. Since it's not always possible to find exactly equal numbers, I determined some small error rates in these functions.

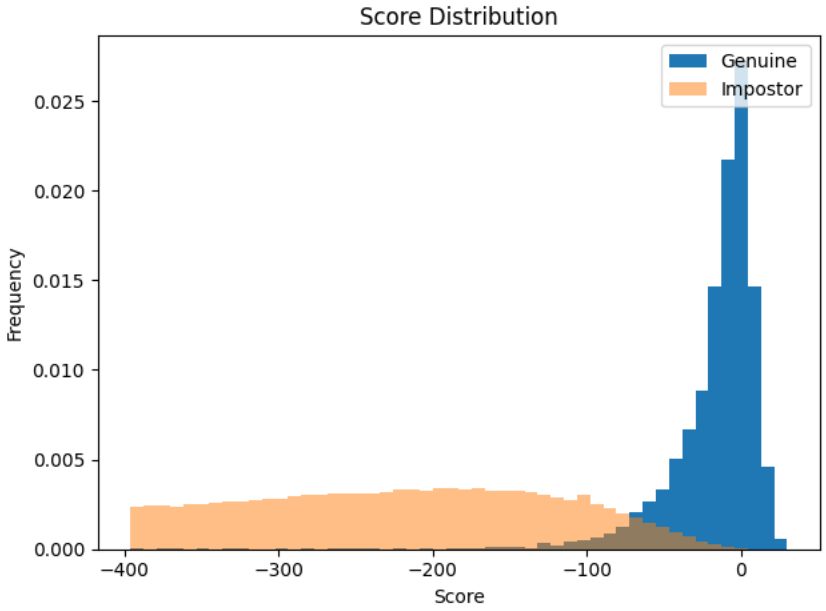
The recursive approach I mentioned above also can be implemented as code (find the best threshold range and change the threshold step size and run again in this "best" threshold range recursively).

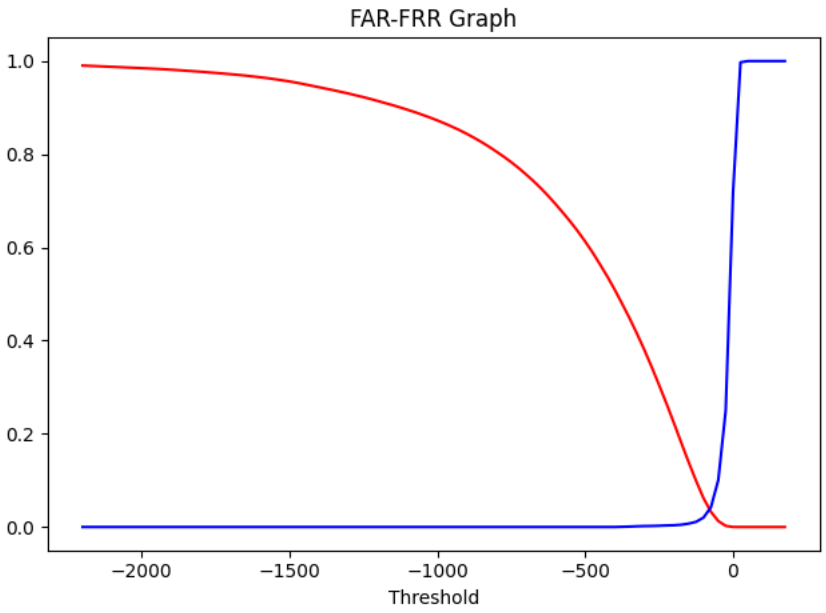
Data 1

1)

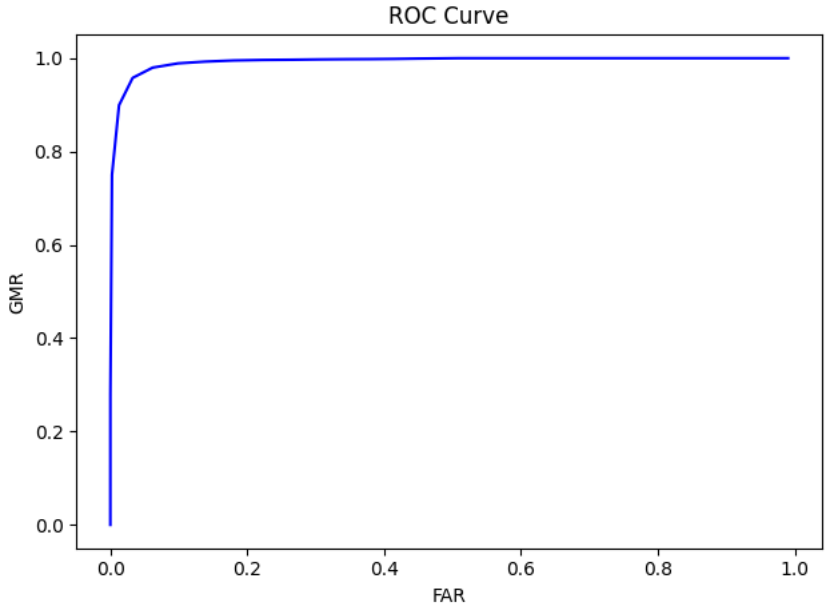
EER	0.036	EER Threshold	-79
FRR	0.348	at FAR point	0.1%
FRR	0.117	at FAR point	1%
FRR	0.010	at FAR point	10%

2)





3)



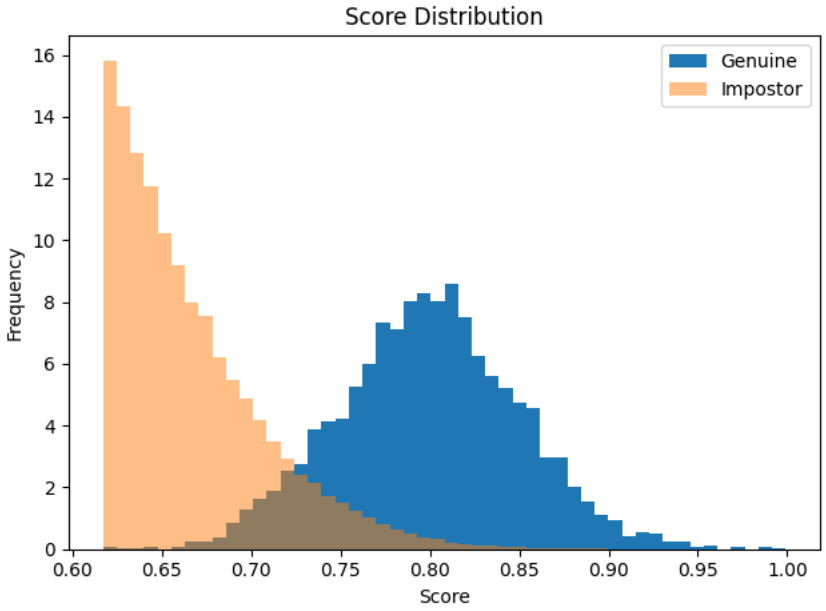
4) I explained my ERR calculation approach in “Introduction” section. ERR point is the point where FRR and FAR are equal. To find these point, I checked for the thresholds where FAR and FRR were closest to each other. For this dataset, there were threshold -75 and -100. So I ran the code between thresholds -75 and -100 with threshold step size 1 (instead of 25). And at threshold -79 I realize they were equal.

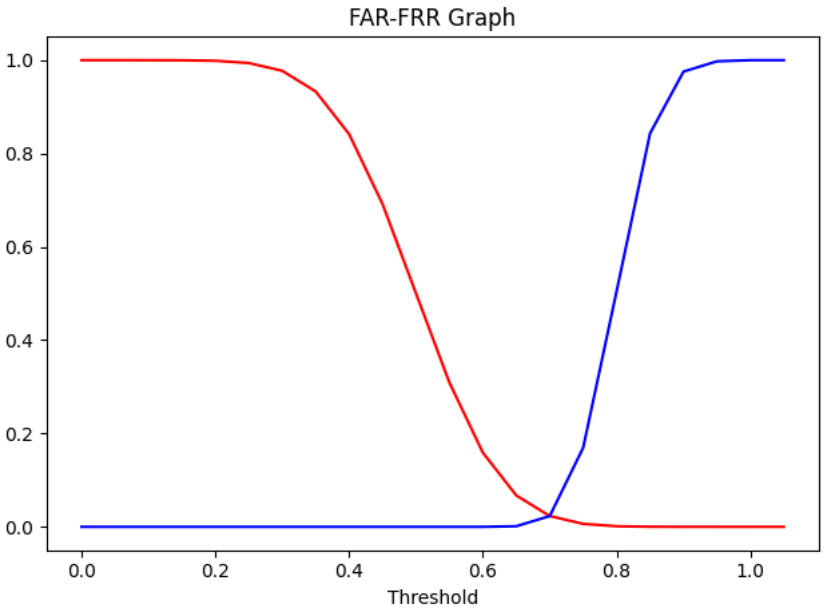
Data 2

1)

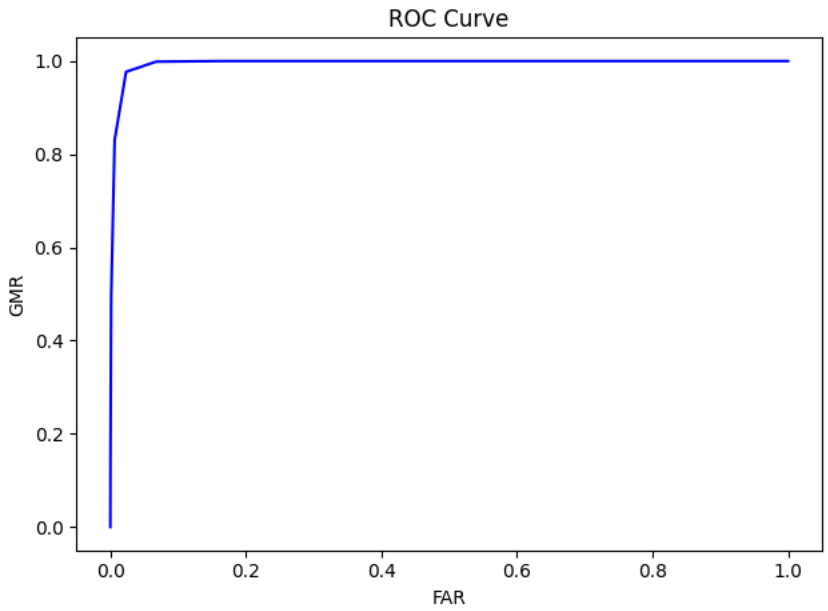
EER	0.023	EER Threshold	0.7
FRR	0.56	at FAR point	0.1%
FRR	0.087	at FAR point	1%
FRR	0.0006	at FAR point	10%

2)





3)



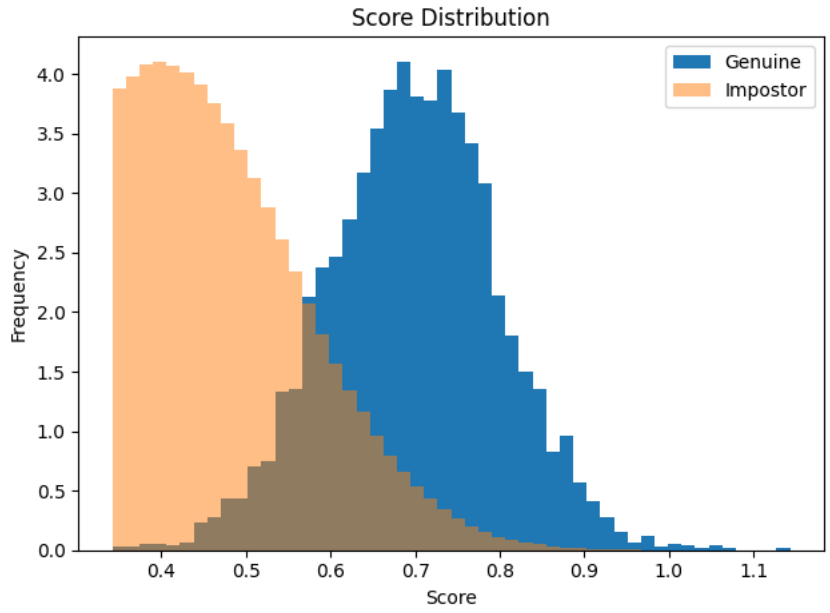
4) I explained my ERR calculation approach in “Introduction” section. ERR point is the point where FRR and FAR are equal. To find these point, I checked for the thresholds where FAR and FRR were closest to each other. For this dataset, there were threshold 0.7 and 0.75. So I ran the code between thresholds 0.7 and 0.75 with threshold step size 0.01 (instead of 0.05). And at threshold 0.7 I realize they were most similar.

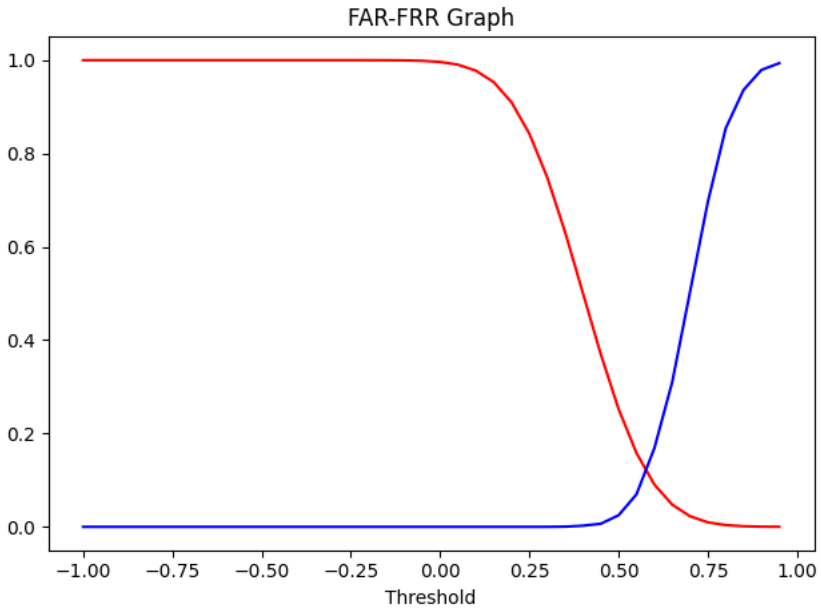
Data 3

1)

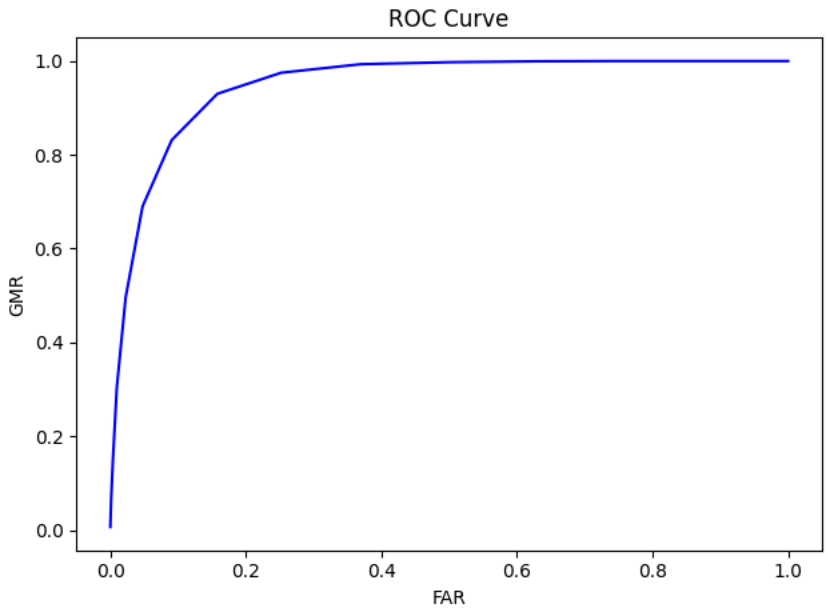
EER	0.1167	EER Threshold	0.578
FRR	0.947	at FAR point	0.1%
FRR	0.691	at FAR point	1%
FRR	0.144	at FAR point	10%

2)





3)



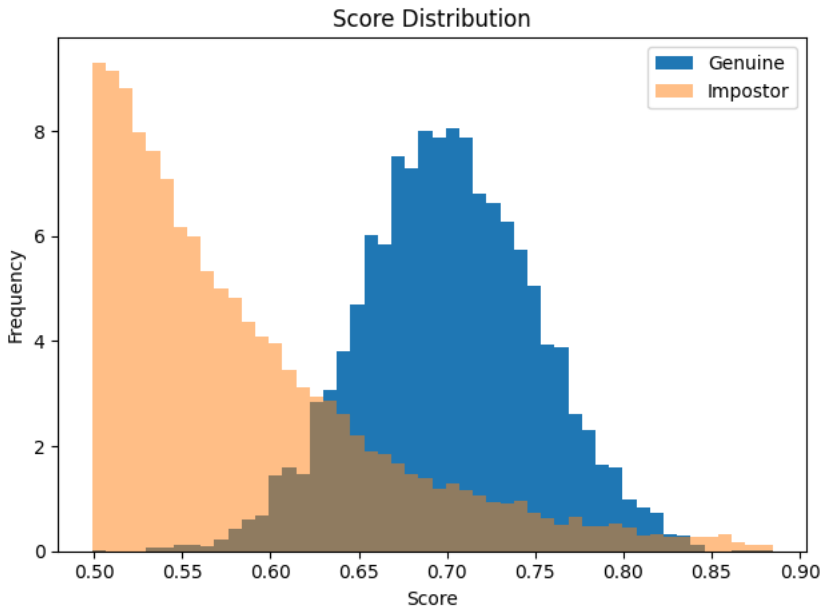
4) I explained my ERR calculation approach in “Introduction” section. ERR point is the point where FRR and FAR are equal. To find these point, I checked for the thresholds where FAR and FRR were closest to each other. For this dataset, there were threshold 0.55 and 0.6. So I ran the code between thresholds 0.55 and 0.6 with threshold step size 0.001 (instead of 0.05). And at threshold 0.578 I realize they were equal.

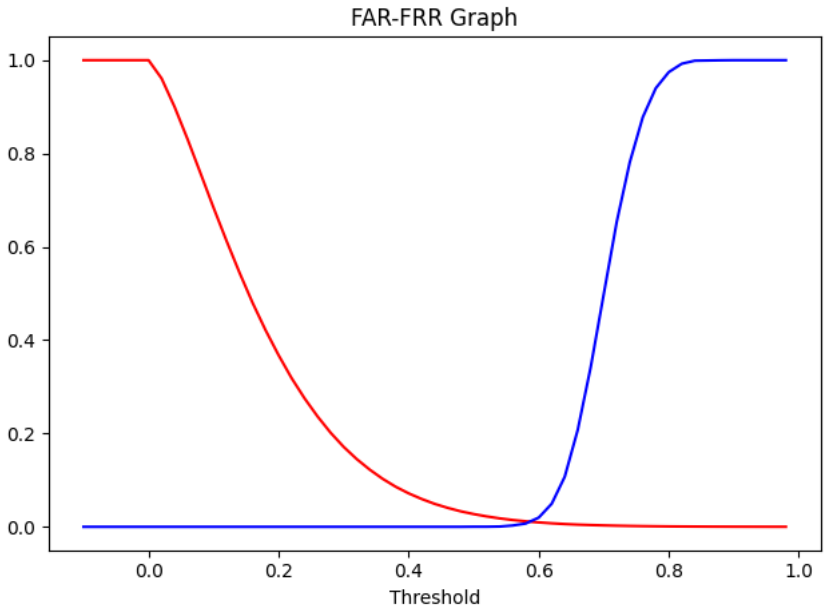
Data 4

1)

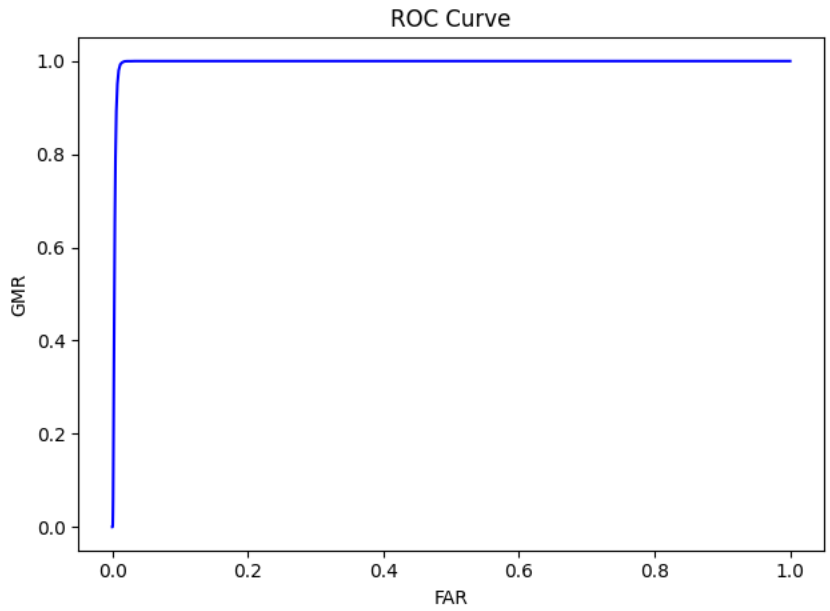
EER	0.011	EER Threshold	0.587
FRR	0.971	at FAR point	0.1%
FRR	0.015	at FAR point	1%
FRR	0.0	at FAR point	10%

2)





3)



4) I explained my ERR calculation approach in “Introduction” section. ERR point is the point where FRR and FAR are equal. To find these point, I checked for the thresholds where FAR and FRR were closest to each other. For this dataset, there were threshold 0.58 and 0.6. So I ran the code between thresholds 0.55 and 0.6 with threshold step size 0.001 (instead of 0.02). And at threshold 0.587 I realize they were equal.