Biometrics: assignment 1

Ingmar Malfait r0693397

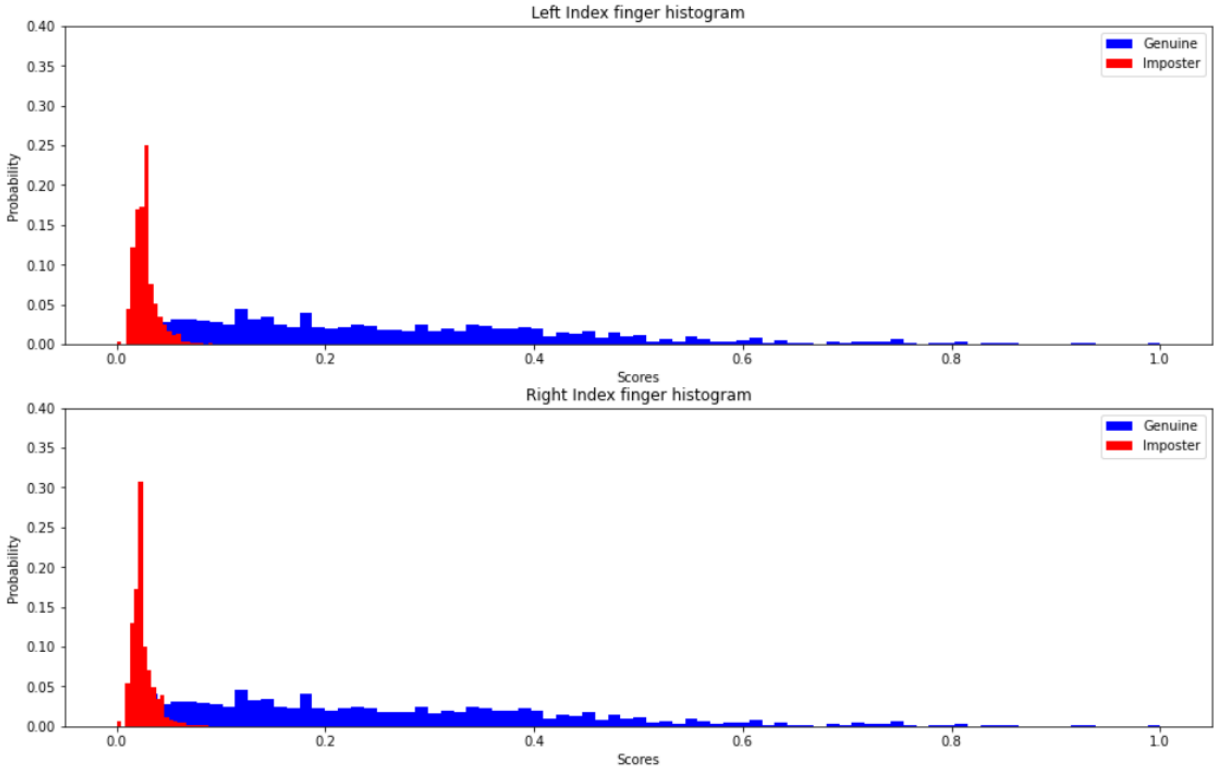
Question 1:

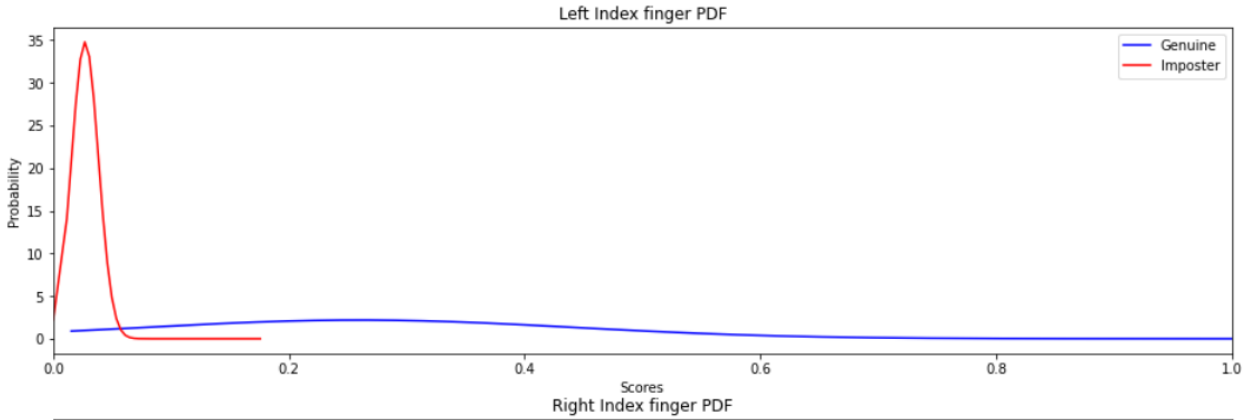
Before answering question one a few things need to be mentioned. Firstly, the mask needs to be applied to the scores using a simple for loop and some numpy arrays. See below snippet.

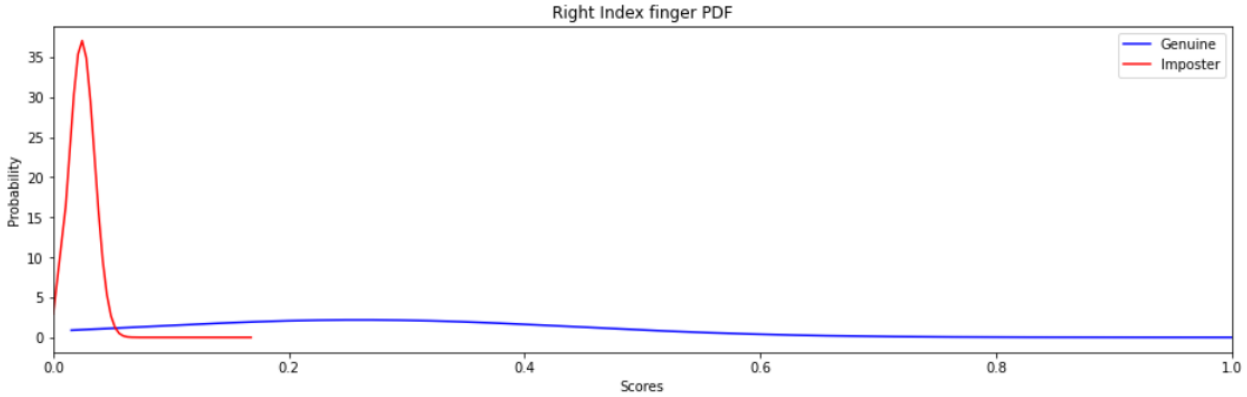
```
def calculateScores(ids, scores):
    imposter_scores = []
    genuine_scores = []
    for i,g_or_not in enumerate(ids):
        if g_or_not == 0:
            imposter_scores.append(scores[i])
        else:
            genuine_scores.append(scores[i])
    genuine_scores = np.sort(genuine_scores)
    imposter_scores = np.sort(imposter_scores)
    return imposter_scores, genuine_scores
```

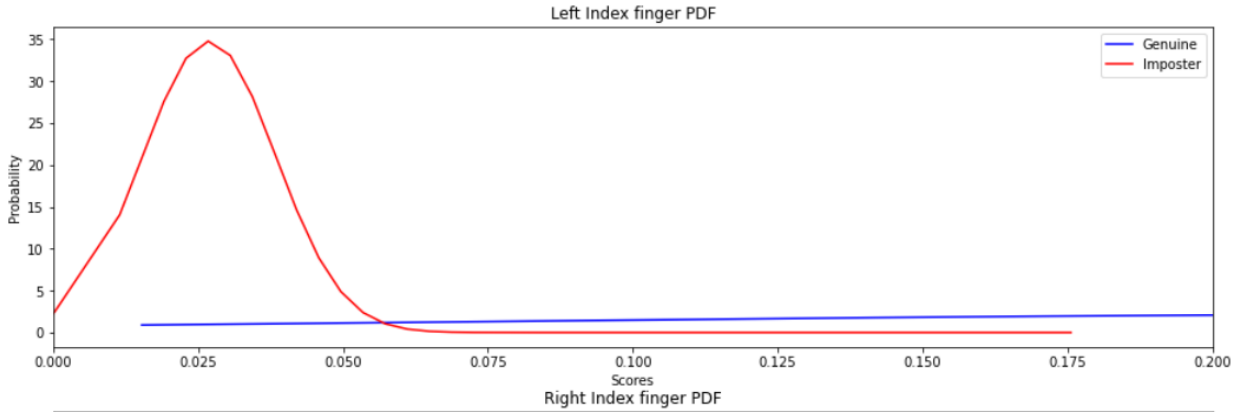
Plots

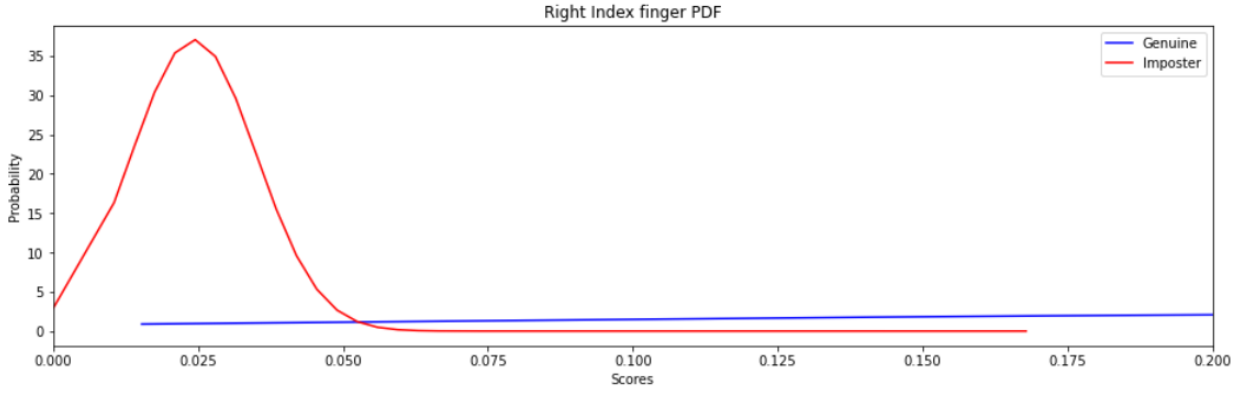
A histogram gives a rough estimate of how the scores are divided so for both left and right index finger a histogram was plotted. Besides this the plot that was asked was the score distribution in the form of a probability distribution function. So also for the two fingers these are plotted. To get the pdf y values scipy.stats.norm.pdf is used.




Left Index finger PDF
Genuine
Imposter
Probability
Scores
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0


Right Index finger PDF
Genuine
Imposter
Probability
Scores
0
5
10
15
20
25
30
35
0.0
0.2
0.4
0.6
0.8
1.0


Left Index finger PDF
Genuine
Imposter
Probability
Scores
0
5
10
15
20
25
30
35
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200


Right Index finger PDF
Genuine
Imposter
Probability
Scores
0
5
10
15
20
25
30
35
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200

From these plots a few things are clear.

1. The genuine and imposter curves do not have much area under the curve overlap. What this means is that not many false negatives nor false positives will be evaluated. In general this is positive. and means **the system works well.**
2. Also the distribution for imposter scores is rather thin. This means that the scores for imposters are small and concentrated. Concentrated scores mean many imposters or other users had the same score and thus there is little difference between fingers and people. In other words **inter user variance is low.** It follows that the identifier is unique, this is not surprising as fingerprints are known to be unique.
3. The distribution for genuine scores is wide. This means that the scores for genuine users are spread out and thus there are big differences between scores of the same user. **intra user variance is high**. It follows that permanence is therefore not great. Which is surprising seeing as permanence for fingerprints is supposed to be good.
4. **Normalisation** with the min max formula seems to **not be all that useful** in this situation. The x axis ranges are already between zero and one. Shifting the axis also gives an inaccurate representation of the data. From the plots as they are it is clear that an imposter has a low score while a genuine user has a high score. Normalisation can shift this perception.
5. When limiting the score range it's also clear that the intersection point is at a score of around 0.05. For both fingers however the left index finger appears to have a higher score for the intersection point.

Question 2:

Before answering the sub questions of question 2, the True Negative, False Negatives, True Positives and False Positives for different thresholds have to be calculated.

The following code snippet is used.

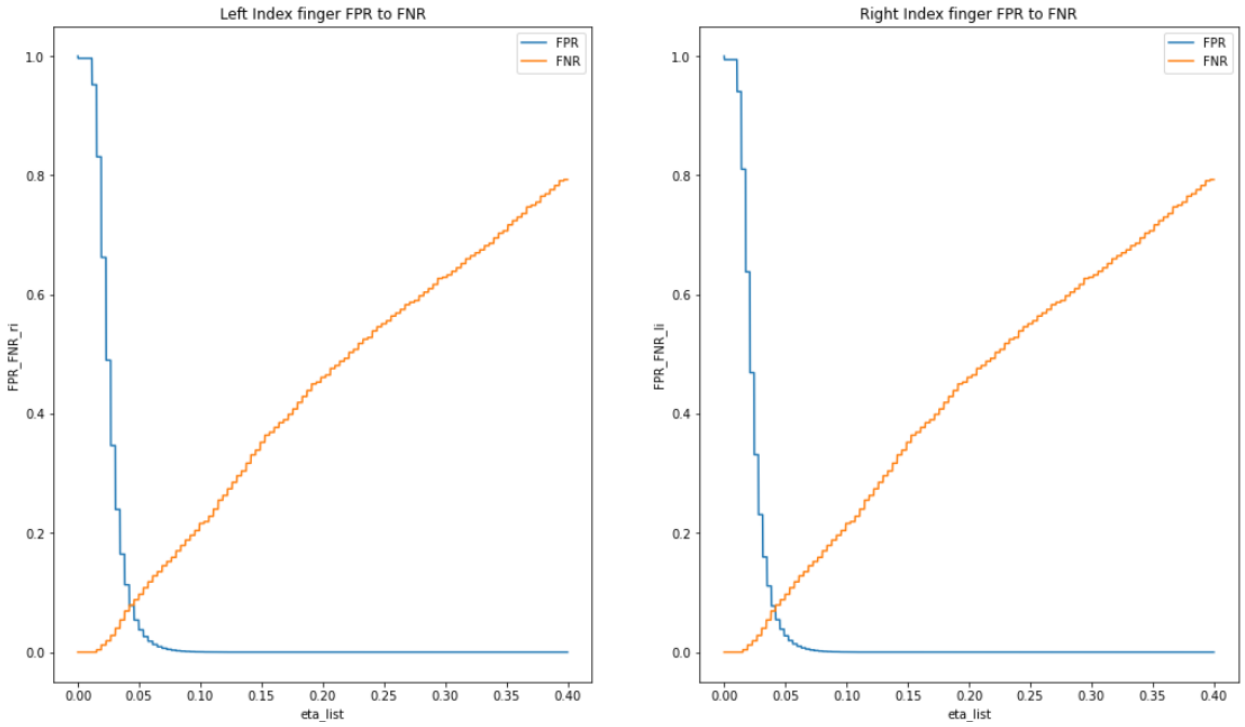
```
def calculateCharacteristics(eta_list, imposter_scores, genuine_scores):
      N = len(imposter_scores)
      P = len(genuine_scores)
      FN_list = np.zeros(len(eta_list))
      TN_list = np.zeros(len(eta_list))
      TP_list = np.zeros(len(eta_list))
      FP_list = np.zeros(len(eta_list))
      for idx, current_eta in tqdm_notebook(enumerate(eta_list)):
      for score in imposter_scores:
            if score>=current_eta:
                  FP_list[idx] += 1 #this is FAR
            else:
                  TN_list[idx] += 1
      for score in genuine_scores:
            if score>=current_eta:
```

```
                TP_list[idx] += 1
            else:
                FN_list[idx] += 1 #this is FRR
    return N,P,FN_list,TN_list,TP_list,FP_list
```

In short a few lists are made for every threshold. Later the values are incremented if their scores are either above or below the threshold.

Then calculate the False Positive Ratio by using element wise division, numpy.true_divide is used here.

FAR and FRR



This plot shows the False Positive Ratio (FPR) or False Acceptance Rate (FAR) compared to the False Negative Ratio (FNR) or False Rejection Ratio (FRR) in function of the threshold. To calculate The intersection a simple for loop will do the trick.

```
average = 0
total = 0
for i,current_eta in enumerate(eta_list):
    if FPR_li[i] + 0.002 > FNR_li[i] and FPR_li[i] - 0.002 < FNR_li[i]:
    average += current_eta
    total += 1
print(average/total)
```

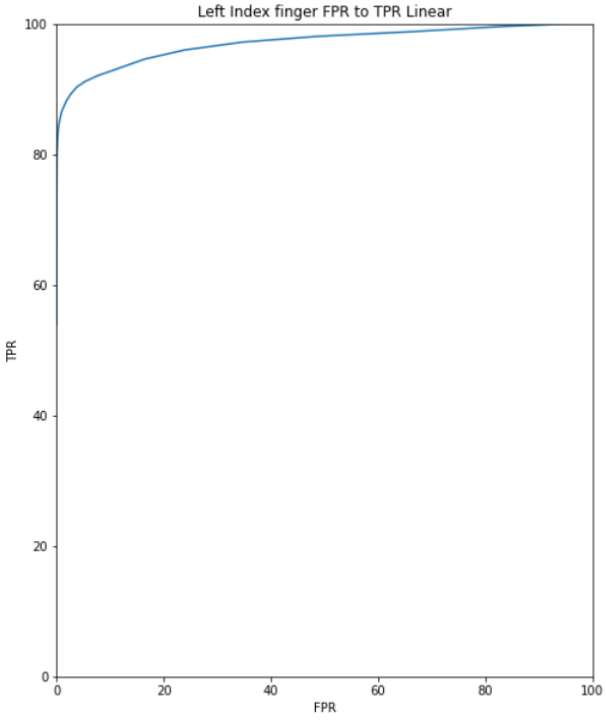
For the left index finger it returns 0.0439 and for the right 0.0402. The question then remains what do these values mean? These eta values indicate when the FAR and FRR are balanced, the Equal Error Rate (EER) point. In a real scenario this is not that useful as every application is different and different

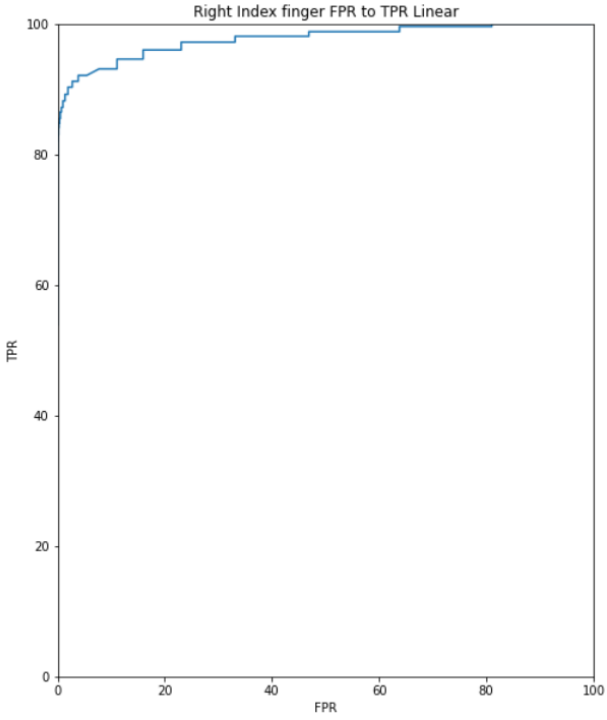
tradeoffs have to be made. e.g. a phone fingerprint sensor has to be convenient rather than secure so the eta can be more to the left (smaller). It is however a great starting point before adjusting the threshold. This just means that the FPR grows and thus more imposters are allowed but less genuine users are rejected. For a nuclear power plant the eta will of course be much bigger for less imposters to be allowed. If the best metrics are wanted then this point would be a good point to put the threshold. The threshold is also close to the intersection point of the genuine and imposter lines. This is no coincidence as going above or under this value the FP and FP change drastically and this is reflected on the plot.

Plotting the FAR and FRR in function of each other also reveals some information.

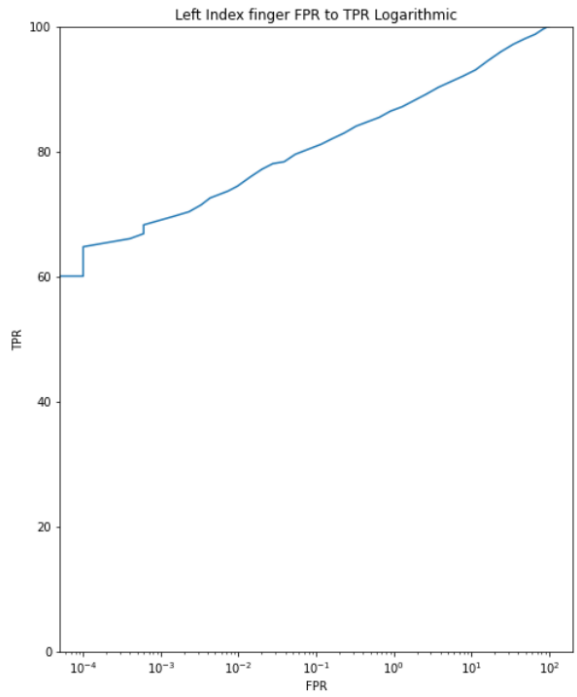
Roc

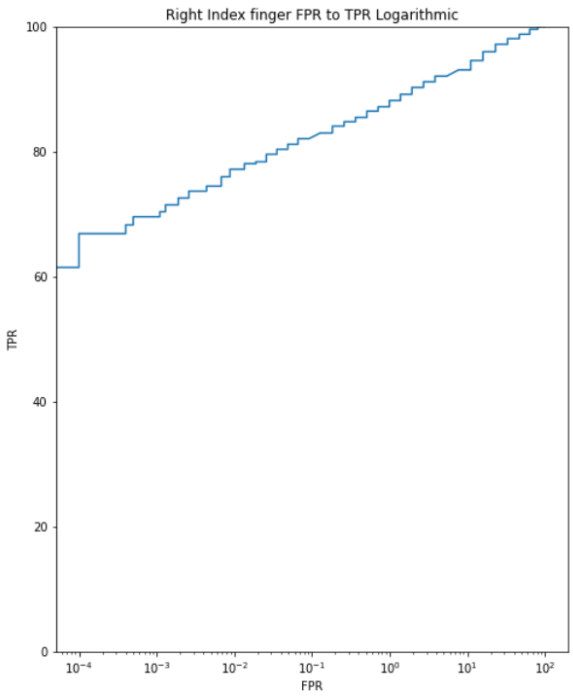
The first plot shows the ROC curve with linear axis. Both right and left index finger show radically lower TPR when the FPR is under 10. So 10 or even 20 can become good benchmarks depending on if you want a more secure system, less false positives and thus under 10 or 20. Or a more convenient system and go above these values even if the tradeoff is many more true positives. In general the curve is very far into the top left corner so it appears to be working well.





The second plot is logarithmic and reveals that for very small false positive ratios True Positives Ratios can drop to as low as 60%. A terrible ratio, so making a system more secure in this case can result in drastic reduction of convenience. Furthermore it is easier to see that the system performs well or not because of the more or less straight line. In general this plot is much more useful seeing as now the 60% TPR, 80% TPR are much easier to see.



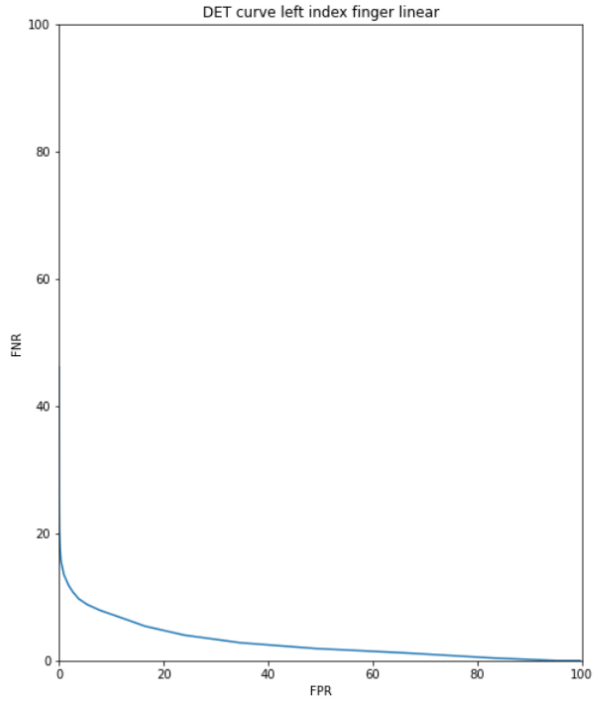


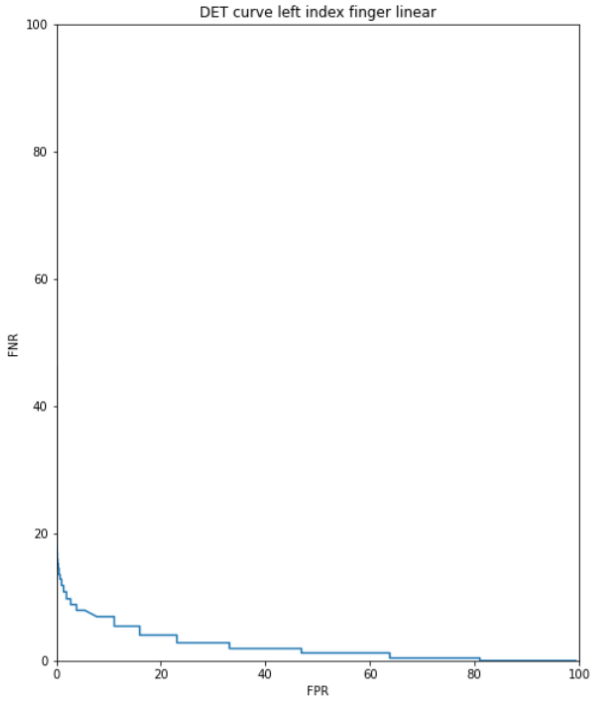
DET

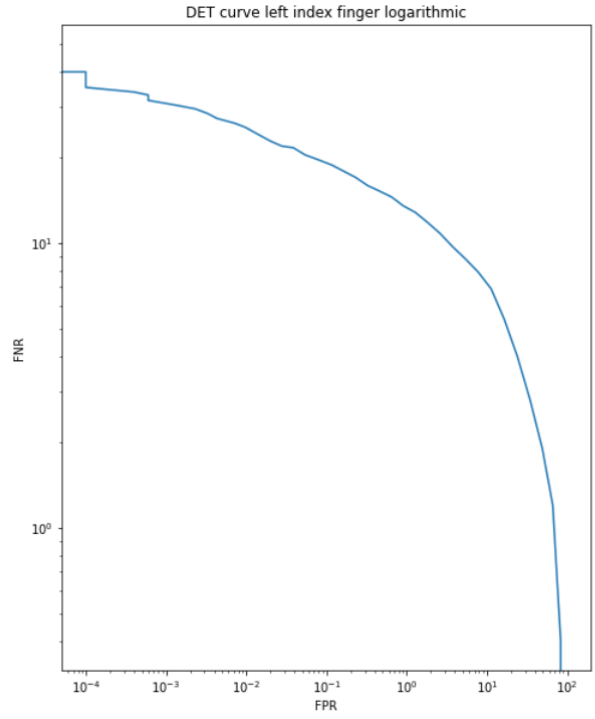
On the linear plot we can clearly see that the system works quite well because the line is close to the ideal curve, being in the left hand bottom corner. Besides this it's also clear that for both index finger linear plots the curve is more up than it is to the left. This means that FPR is bigger compared to FNR which means a more secure system rather than convenient. Because more FNR in general means that more genuine users will be rejected i.e. less convenient.

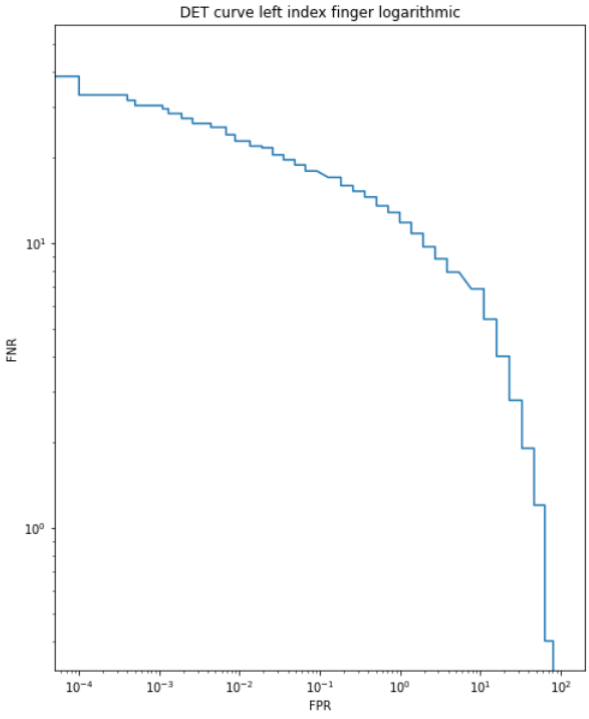
The DET thus makes the convenience vs security tradeoff more clear. And why DET seems preferable.

The logarithmic DET plot also shows a few more things. The most interesting areas for this model are somewhere between 0.1 and 10 FPR as before that range the FNR is very big and after FPR is too big.





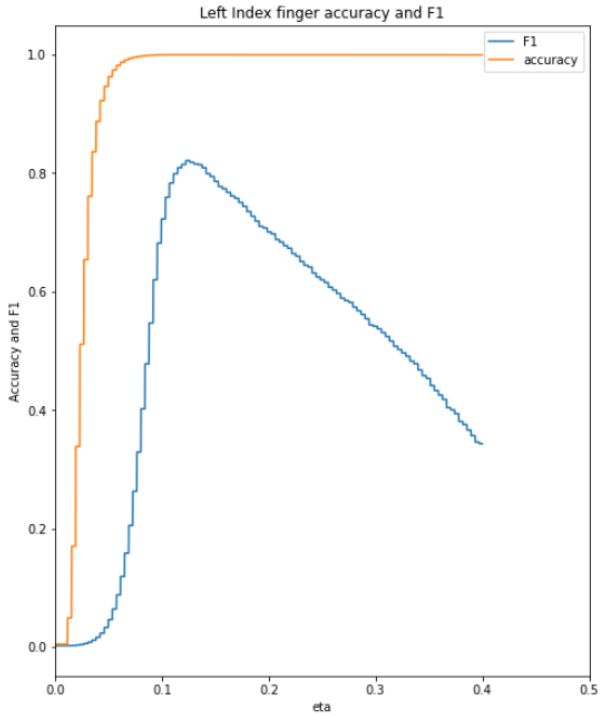


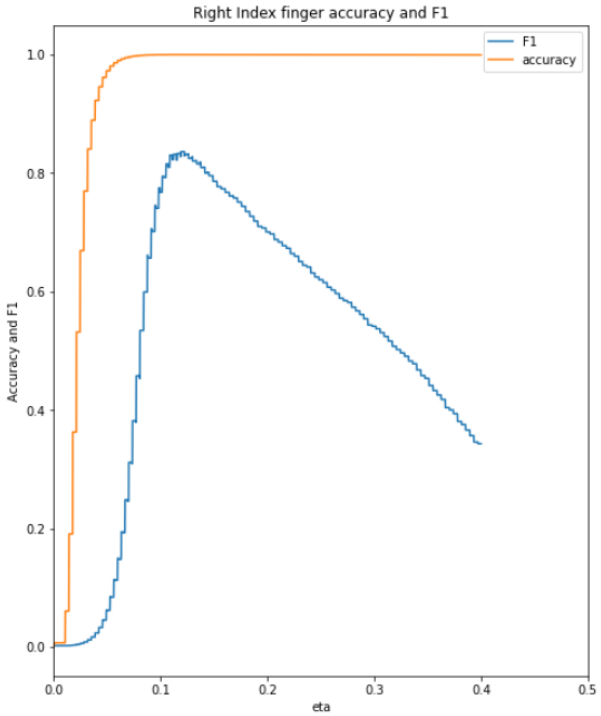


Question 3:

Firstly the plots for F1 and accuracy. What can be observed is that while accuracy just keeps increasing and stagnates, F1 actually goes down after a certain threshold value. As is clear from the plots in

Question 1 and 2 increasing your threshold indefinitely does not result is a better system with better being the tradeoff between security and convenience. F1 on the other hand does take this into account.





The max F1 scores have the following values, threshold values and accuracies.

\	Max value F1	accuracy	threshold
Right index finger	0.8361	0.9997	0.1189
Left index finger	0.8208	0.9997	0.1222

\	Max value accuracy	F1	threshold
Right index finger	0.9997	0.8361	0.1189
Left index finger	0.9997	0.8208	0.1222

At first the values for Accuracy might look surprising but the max value being the same as F1 makes sense. The value for accuracy goes down ever so slightly when F1 goes down because when F1 is at its max recall and precision are balanced. This means that the threshold is balanced between FP and FN. if the threshold increases beyond this point more FN appear. This is not much compared to the amount of FP as there are many more samples for imposters than genuine. This slight change in FP is reflected in the slight decrease in TP, and thus less accuracy even if it is every so slightly. This also confirms again that accuracy is not a good measure as this slight change is barely visible and gives a wrong representation.

This point is also the point at which the tradeoff between FN and FP is the best i.e. the sum is the lowest. However a more secure or more convenient system is achieved by moving the eta to the right or left.

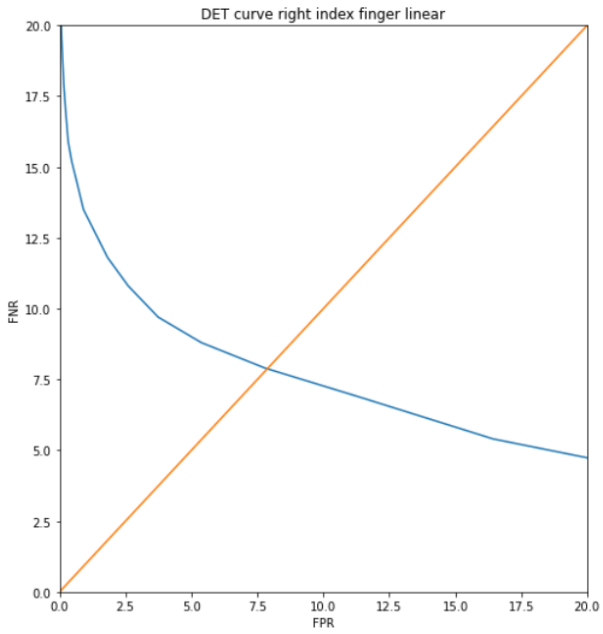
Question 4:

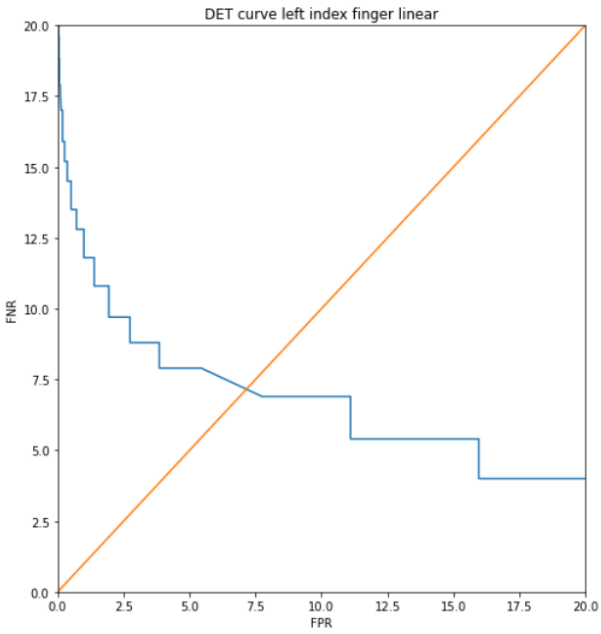
AUC

The AUC for the left index finger and right index finger is 0.971 and 0.983 respectively. While AUC does tell how "good" a system is i.e. how much security and convenience there is in total. It does not tell you if it is more secure rather than convenient. It also doesn't look at FN nor TN.

EER

The plot shows FPR/FNR values close to 7.5 for both fingers.





Also with some approximation logic the calculations reveal a similar value, the table below shows the exact values.

	EER (FPR and FNR)	threshold
left index finger	0.079 (*100)	0.0439
right index finger	0.069 (*100)	0.0400

Generally these values are considered small and thus the system is at a good operating point. Notice that these values are the same as the intersection point values of the FAR FRR curve of question 2. Logical of course as these are the same metrics.

Total classification error

The points at which the sum of FRR and FAR are minimal are shown in the table below.

	min(FRR + FAR) eta score
left index finger	0.0535
right index finger	0.049

These results seem far off the expected results. What is expected as stated in the lab text is the same scores as the EER. There must be an error in the code.

Other optimal performance strategies

One other performance metric mentioned in class was that of d-prime.

Easily implemented with the following snippet:

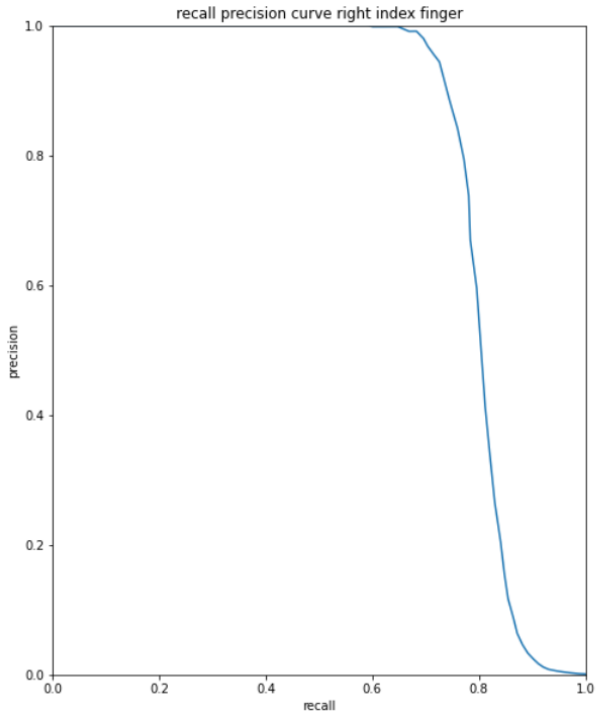
```
def CalculateDPrime(genuine_scores, imposter_scores):
      mu_g = np.average(genuine_scores)
      mu_i = np.average(imposter_scores)
      omega_g = np.std(genuine_scores)
      omega_i = np.std(imposter_scores)
      dPrime = 1.414*(mu_g-mu_i)/ np.sqrt(omega_g**2 + omega_i**2)
      return dPrime
```

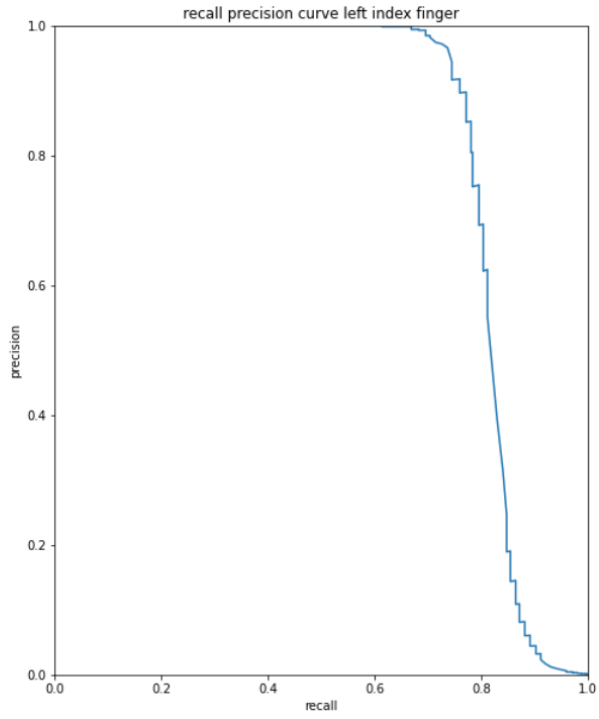
The resulting d-prime values are 1.802 and 1.822 for the left and right index finger respectively. In this way the right index finger performs a bit better overall. These d-prime values indicate the distance between the averages. This means a higher value is more distance and thus less overlap. A value of d == 2 is considered decent to good. The advantage is that it gives a good overall score not based on FNR nor FPR. The disadvantage is that is doesn't give much more info about the system besides the overall performance. It also doesn't take into account the fact that different distributions might be much wider or thinner and this also affects overlap.

Question 5:

Precision recall curve

The precision recall curve indicates that for recall values lower than 0.6 a really high precision is achieved. However, for values higher than 0.85 precision is very low. So in general precision will be much higher than recall. This means that the system is precise i.e. a low amount of FP get trough in general. So the system is secure rather than convenient… again.





AUC

The bigger the area under the curve the bigger the performance overall due to a bigger recall or precision. However this metric is reductionist. It's a good overall performance measure but for specific question such as is the recall high compared to precision it's not suited and thus questions of security and convenience are left unanswered. In general for both left and right index fingers the AUC is large and thus confirms again that the system works well.

	AUC P-R curve
left index finger	0.802
right index finger	0.817

Average precision-score

The average precision score in this case is large. Again this metric is reductionist, not only precision matters as it only takes into account false positives or imposters but not genuine or false negatives. A Average precision-score of 0.5 can mean that the system has great precision from 0 to 0.5 and horrible afterwards or a generally ok precision from 0 to 1. We are left in the dark about this.

	Average Precision-score
left index finger	0.799
right index finger	0.860

Question 6:

To get the CMC curve first the matrix must be taken row per row. Then the score has to be kept track of. When the row is sorted find the first value with the score and use its index as a value for the list that will eventually become the CMC curve. The CMC curve is sorted and iterated over and the positions are counted. The last step is then simply to make it cumulative by adding the last element to the current element.

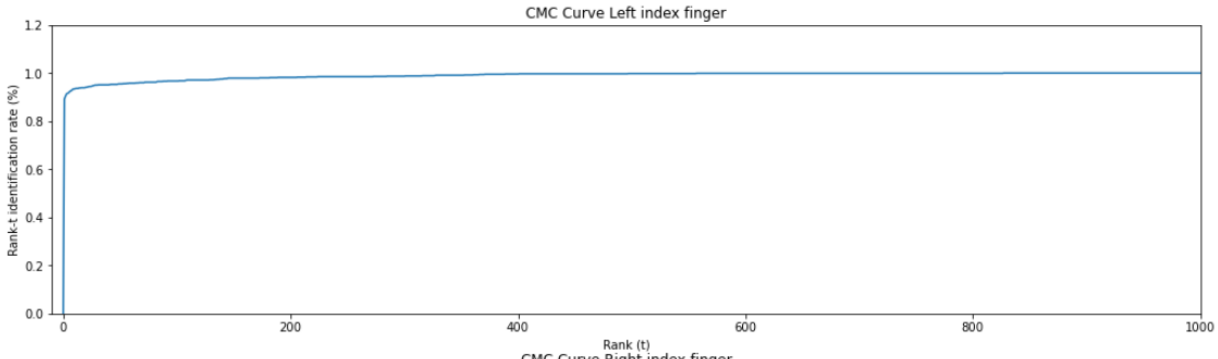
The following code snippet should make this clear:

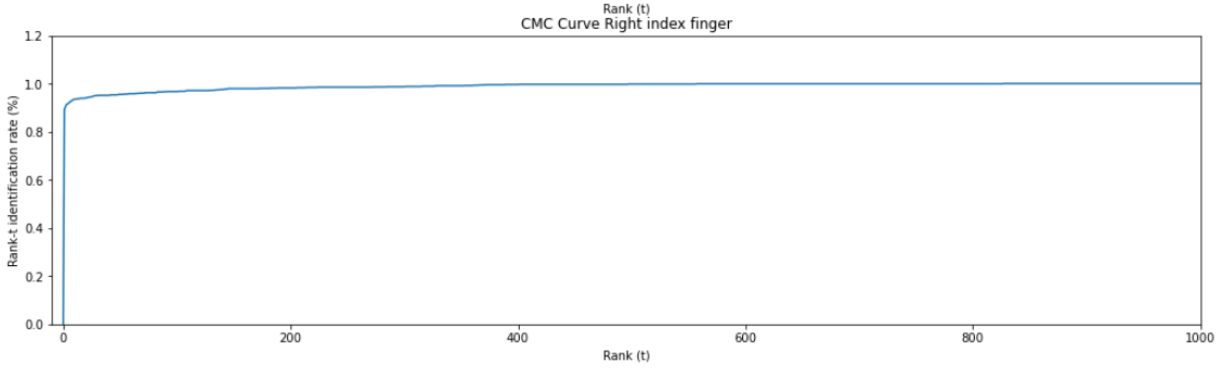
```
# manual without library
def calculateCumSum(matrix):
      pos = np.zeros(len(matrix[0]))
      for i in tqdm_notebook(range(0, len(matrix[0]))):
      temp_row = np.sort(ri_similarity_matrix.values[i])[::-1]
      temp_pos = np.where(temp_row == ri_similarity_matrix.values[i][i])
      pos[i] = temp_pos[0][0]
      pos_sort = np.sort(pos)

      cumsum = np.zeros(len(pos))
      for ind,position in enumerate(pos_sort):
      cumsum[int(position)] += 1

      for ind,sum in enumerate(cumsum):
      if ind>0:
            cumsum[ind] =  cumsum[ind] + cumsum[ind-1]
      return np.insert(np.divide(cumsum,len(pos)), 0, 0)
```

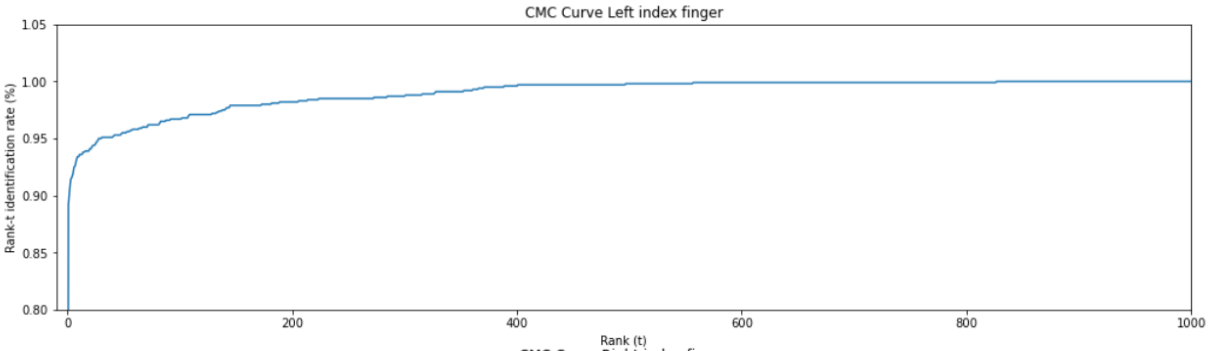
The following plots are acquired. The second pair of plots is simply a zoomed in version of the first.

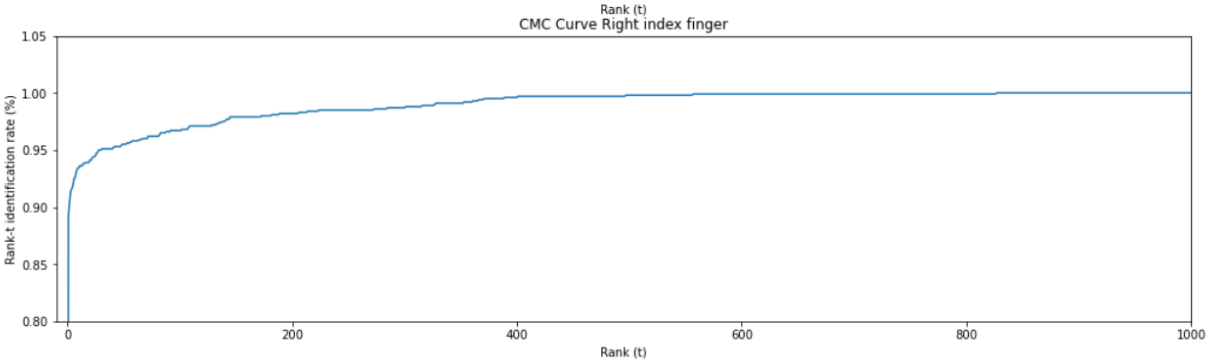
What is clear form these plots is that the system performs very well with a very steep rise initially.

Also the rank-1 recognition rate is good, being 0.892 for both left and right index finger. This means that in 89.2% of the cases the first attempt of a genuine attempt will be a True positive.









Question 7:

In general the right index finger system performs slightly better than its left index finger counterpart. One reason for this might be that a larger part of the population is right handed. So the hardware might be biassed against left index fingers. This seems unlikely, what seems more likely is that people place their right hand better on the sensor. unintentionally this could make the acquired data harder to parse. Especially if the engineers making the software also test mostly on right handed samples.