Deep Learning Class Project

AGE INVARIANT FACE RECOGNITION

Parth Saboo [2019A4PS0457P]

SIFT

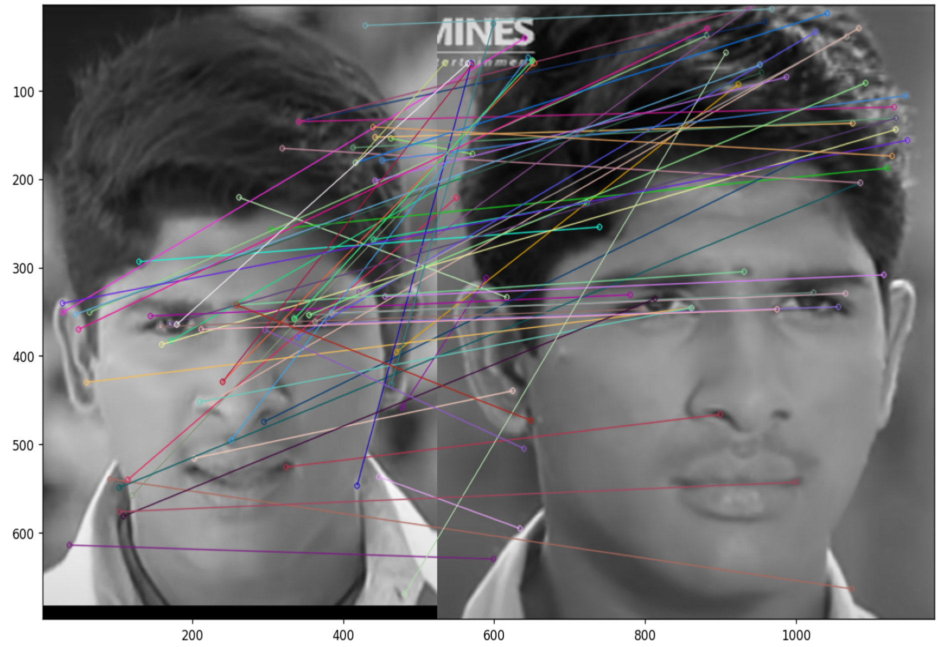
In 2004, **D.Lowe**, University of British Columbia, came up with a new algorithm, Scale Invariant Feature Transform (SIFT) in his paper, **Distinctive Image Features from Scale-Invariant Keypoints**, which extract keypoints and compute its descriptors. *(This paper is easy to understand and considered to be best material available on SIFT. This explanation is just a short summary of this paper)*.

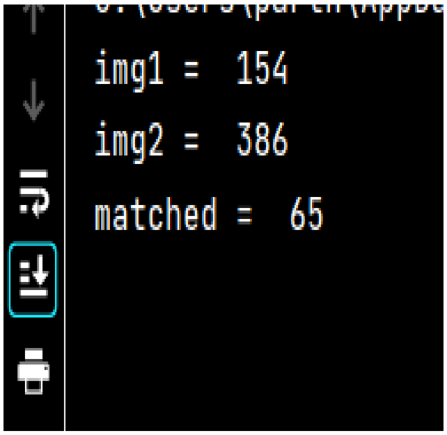
There are mainly four steps involved in SIFT algorithm.

a) **Scale-space Extrema Detection**
b) **Keypoint Localization**
c) **Orientation Assignment**
d) **Keypoint Descriptor**
e) **Keypoint Matching**

We used SIFT module in OpenCV with KNN(k=2) as the matching distance

Results:

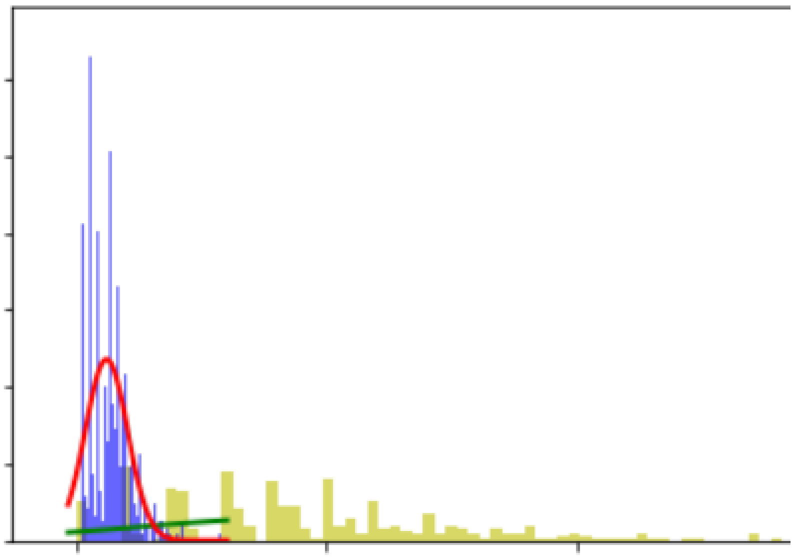




```
img1 =  154
img2 =  386
matched =  65
```

Fig- Keypoints Matching

Imposter vs Genuine Curve

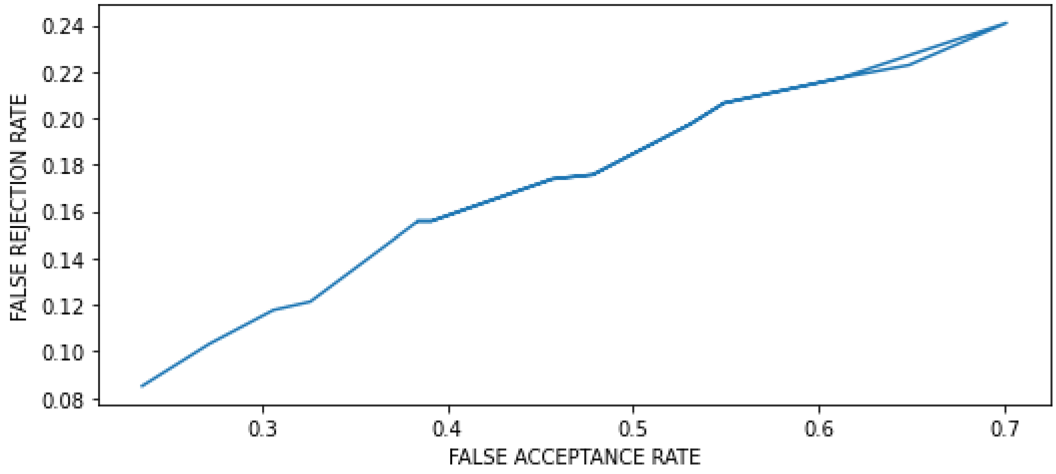


Most of the matches will be imposter as only 50 images should match to a given random image, hence we can see high frequency at the lower side of score due to more imposter matching.

Threshold = 7.6615019 (= equal error rate)

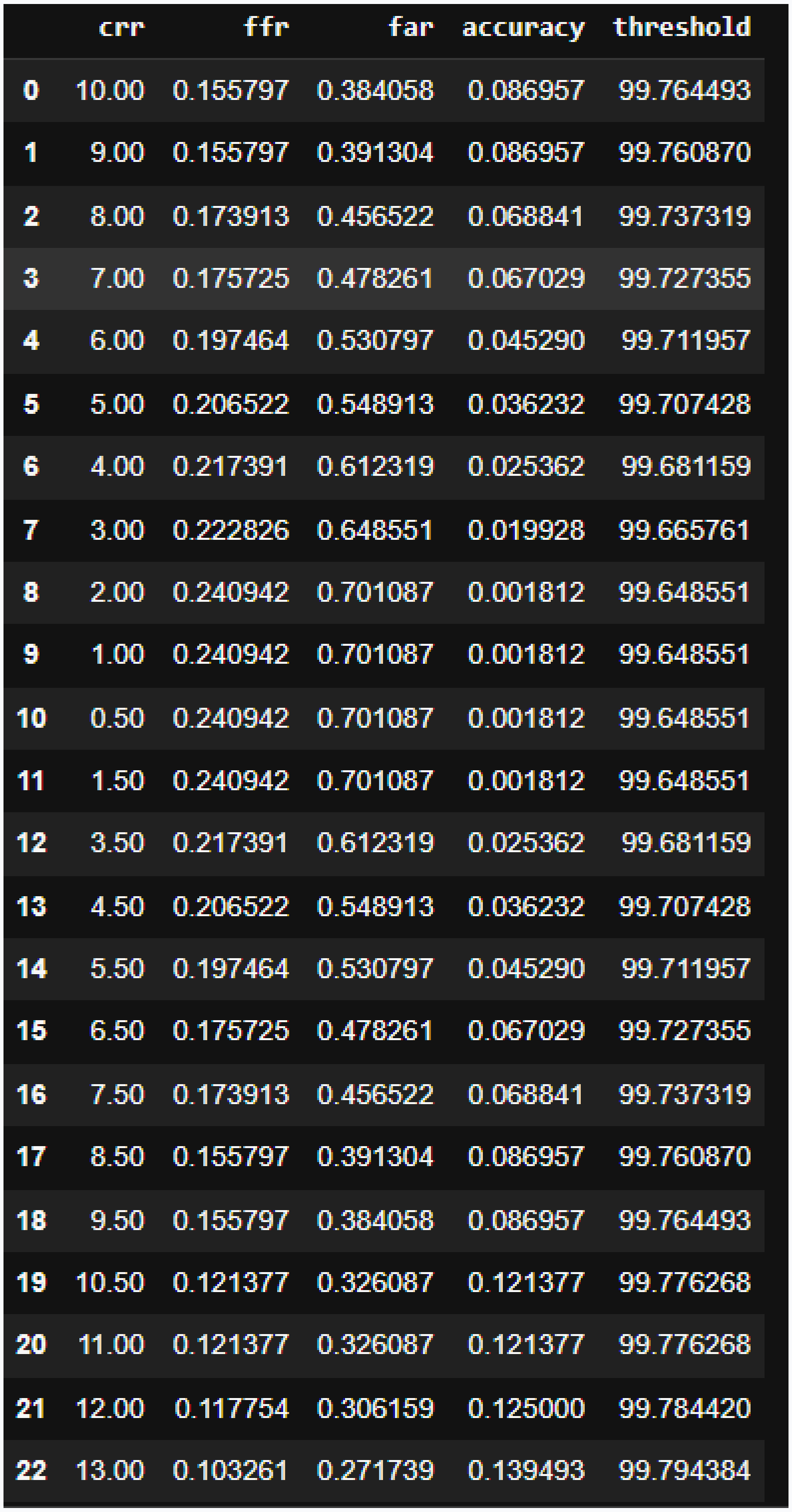
Accuracy = 99.73%
CRR = 0.0688
FAR = 0.456
FFR = 0.1647

ROC Curve



Datapoints for various thresholds

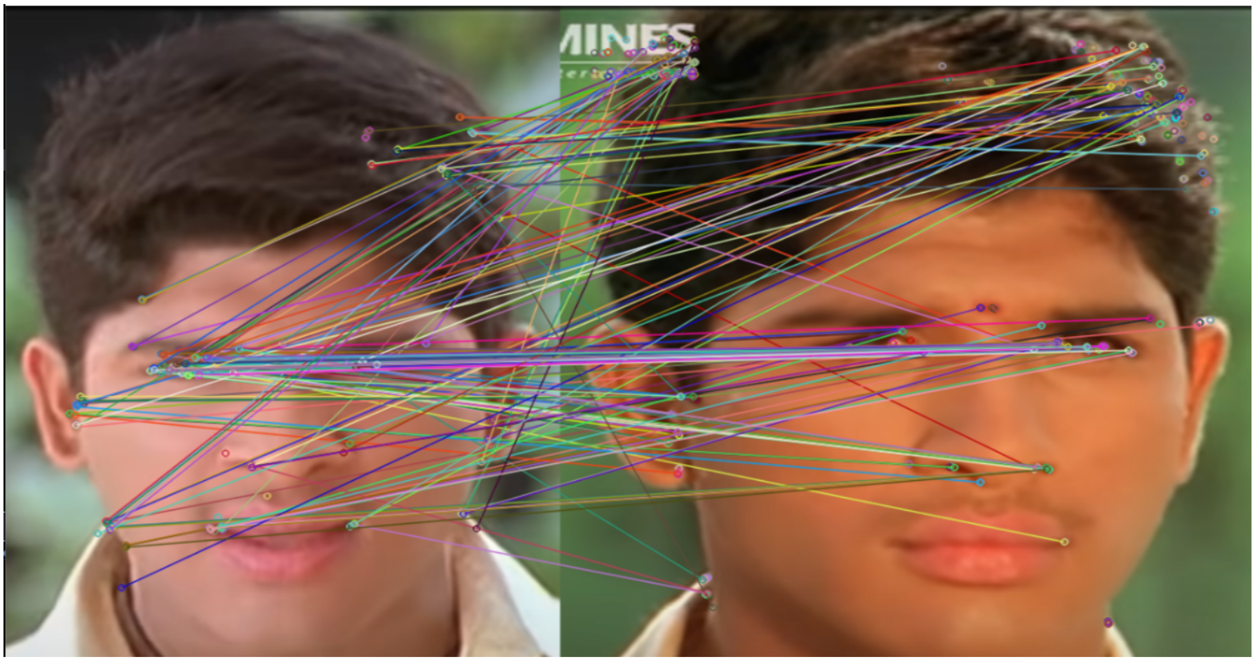
Typo error: [accuracy<-> threshold, crr<-> threshold]

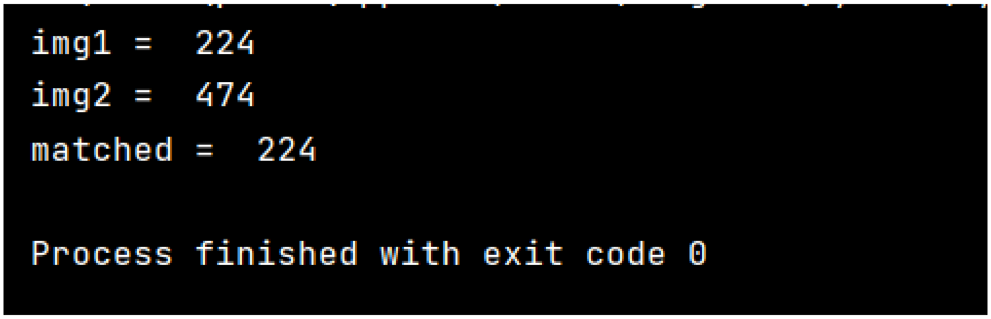


	crr	ffr	far	accuracy	threshold
0	10.00	0.155797	0.384058	0.086957	99.764493
1	9.00	0.155797	0.391304	0.086957	99.760870
2	8.00	0.173913	0.456522	0.068841	99.737319
3	7.00	0.175725	0.478261	0.067029	99.727355
4	6.00	0.197464	0.530797	0.045290	99.711957
5	5.00	0.206522	0.548913	0.036232	99.707428
6	4.00	0.217391	0.612319	0.025362	99.681159
7	3.00	0.222826	0.648551	0.019928	99.665761
8	2.00	0.240942	0.701087	0.001812	99.648551
9	1.00	0.240942	0.701087	0.001812	99.648551
10	0.50	0.240942	0.701087	0.001812	99.648551
11	1.50	0.240942	0.701087	0.001812	99.648551
12	3.50	0.217391	0.612319	0.025362	99.681159
13	4.50	0.206522	0.548913	0.036232	99.707428
14	5.50	0.197464	0.530797	0.045290	99.711957
15	6.50	0.175725	0.478261	0.067029	99.727355
16	7.50	0.173913	0.456522	0.068841	99.737319
17	8.50	0.155797	0.391304	0.086957	99.760870
18	9.50	0.155797	0.384058	0.086957	99.764493
19	10.50	0.121377	0.326087	0.121377	99.776268
20	11.00	0.121377	0.326087	0.121377	99.776268
21	12.00	0.117754	0.306159	0.125000	99.784420
22	13.00	0.103261	0.271739	0.139493	99.794384

ORB (Oriented FAST and Rotated BRIEF)

It is a good alternative to SIFT and SURF in computation cost, matching performance. ORB is basically a fusion of FAST keypoint detector and BRIEF descriptor with many modifications to enhance the performance. First it use FAST to find keypoints, then apply Harris corner measure to find top N points among them. It also use pyramid to produce multiscale-features.

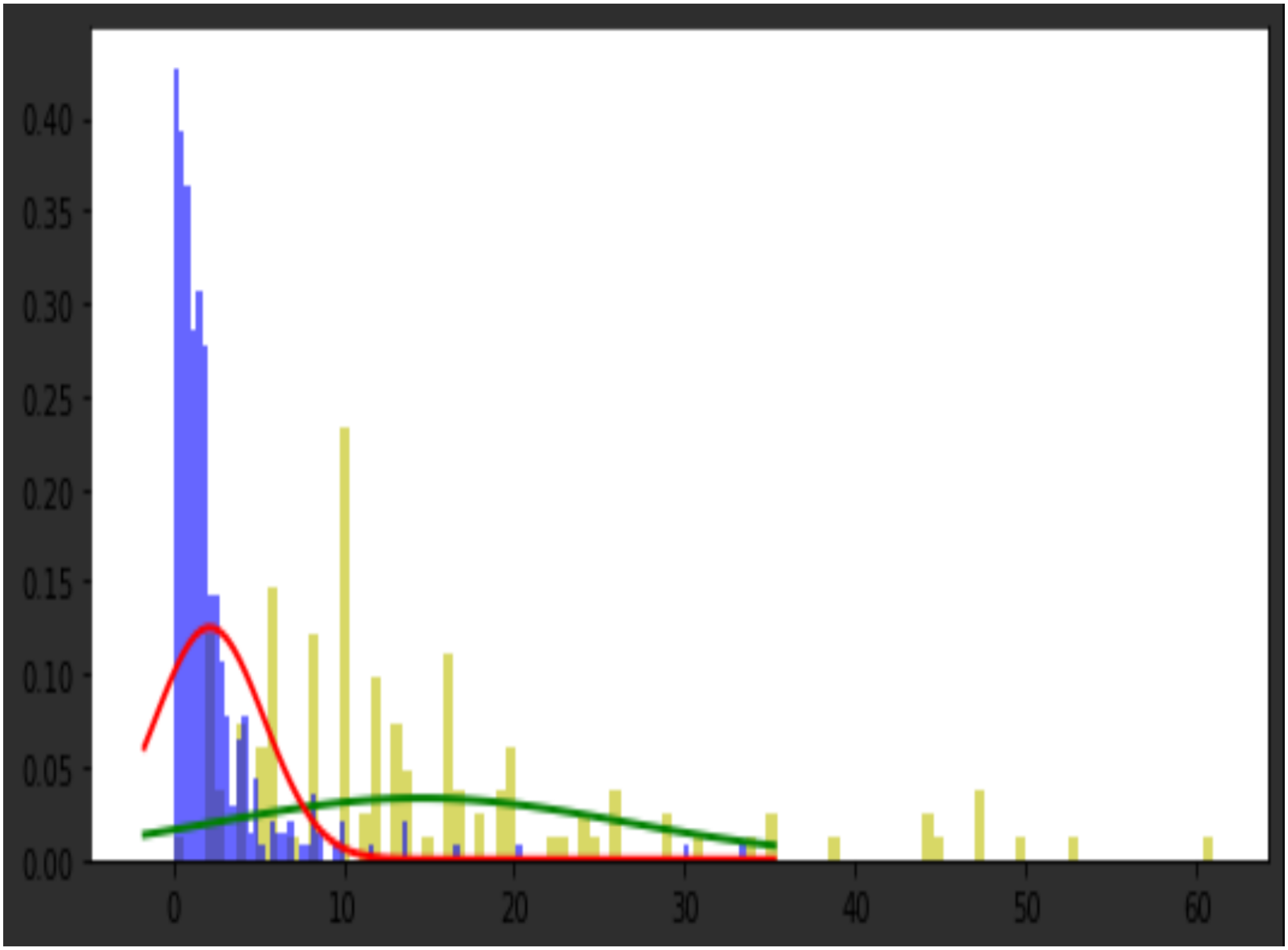




```
img1 =  224
img2 =  474
matched =  224

Process finished with exit code 0
```

Threshold score obtained = 0.3132

Genuine vs Imposter Histogram



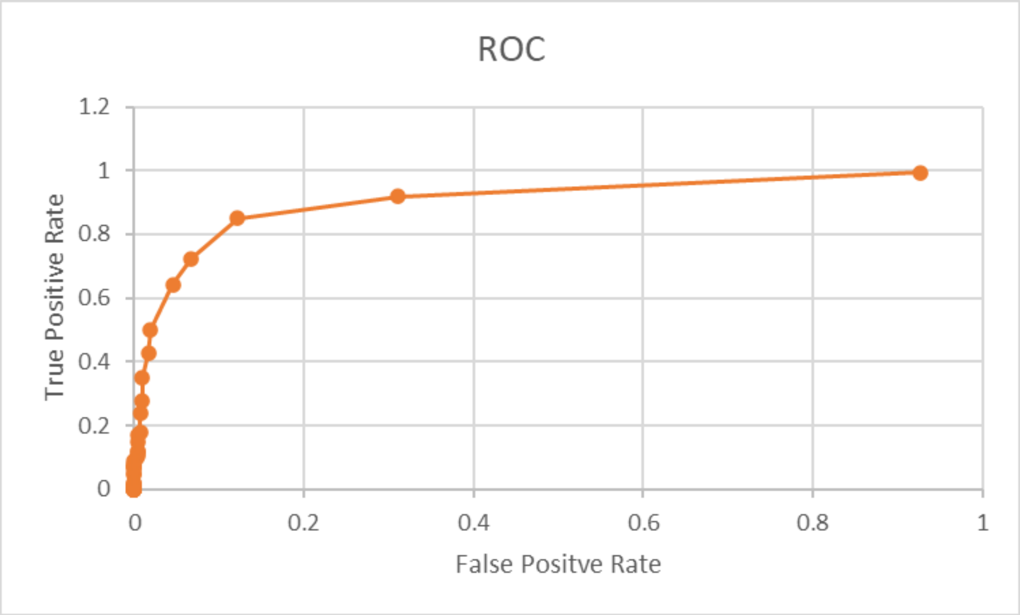
Some data Points from Excel

False Positive Rate	True Positive Rate	AUC
0.992537313	0.925837321	0.020888381
0.917910448	0.311004785	0.015478826
0.850746269	0.122009569	0.005498822
0.723880597	0.066985646	0.006445047

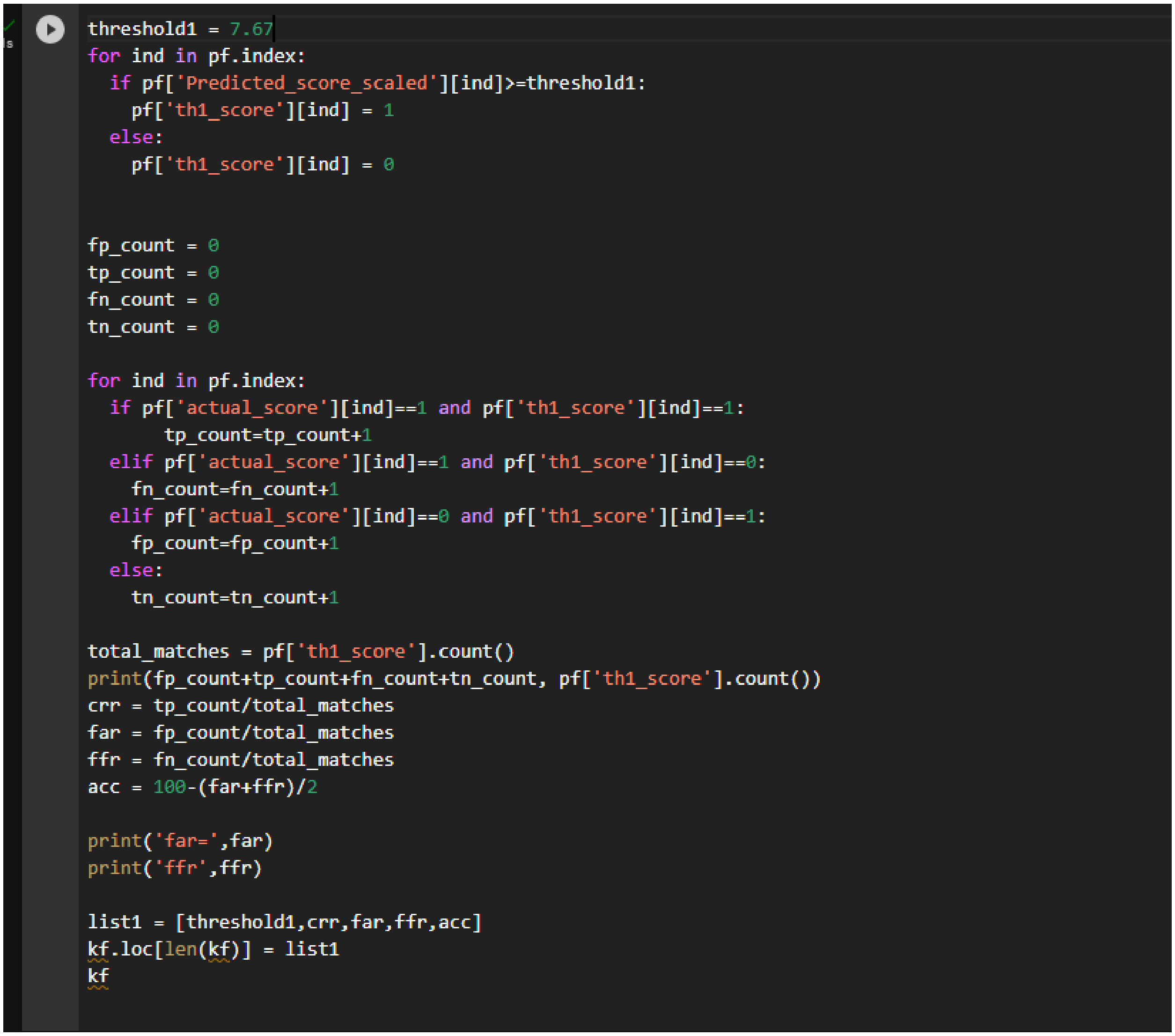
0.641791045	0.045454545	0.001428265
0.5	0.019138756	0.001249732
0.425373134	0.016746411	0.000714133
0.350746269	0.009569378	0.000357066
0.276119403	0.009569378	0.00042848
0.23880597	0.007177033	5.356E-05
0.179104478	0.007177033	0.00010712
0.171641791	0.004784689	0.000142827
0.149253731	0.004784689	0
0.119402985	0.004784689	7.14133E-05
0.119402985	0.004784689	3.57066E-05
0.104477612	0.004784689	1.78533E-05
0.097014925	0.002392344	0
0.089552239	0	0
0.074626866	0	0
0.074626866	0	0
0.067164179	0	0
0.067164179	0	0
0.052238806	0	0
0.044776119	0	0
0.02238806	0	0
0.014925373	0	0
0.014925373	0	0
0.007462687	0	0

0.007462687	0	0
0.007462687	0	0
0.007462687	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
	0	0

ROC curve from orb feature mapping



CODE (to calculate Evaluation Metrics & plot ROC curve)



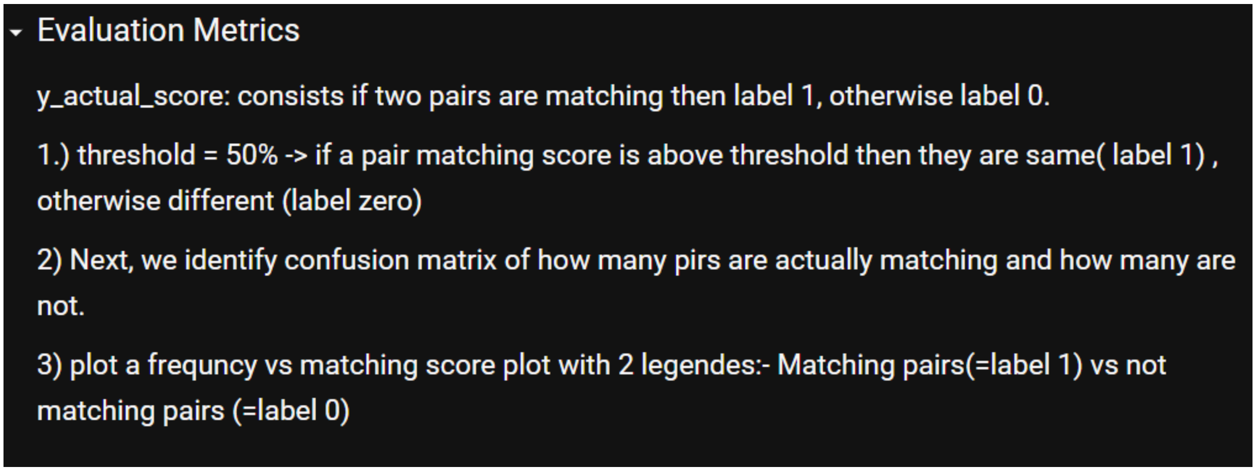
```
threshold1 = 7.67
for ind in pf.index:
  if pf['Predicted_score_scaled'][ind]>=threshold1:
    pf['th1_score'][ind] = 1
  else:
    pf['th1_score'][ind] = 0


fp_count = 0
tp_count = 0
fn_count = 0
tn_count = 0

for ind in pf.index:
  if pf['actual_score'][ind]==1 and pf['th1_score'][ind]==1:
       tp_count=tp_count+1
  elif pf['actual_score'][ind]==1 and pf['th1_score'][ind]==0:
    fn_count=fn_count+1
  elif pf['actual_score'][ind]==0 and pf['th1_score'][ind]==1:
    fp_count=fp_count+1
  else:
    tn_count=tn_count+1

total_matches = pf['th1_score'].count()
print(fp_count+tp_count+fn_count+tn_count, pf['th1_score'].count())
crr = tp_count/total_matches
far = fp_count/total_matches
ffr = fn_count/total_matches
acc = 100-(far+ffr)/2

print('far=',far)
print('ffr',ffr)

list1 = [threshold1,crr,far,ffr,acc]
kf.loc[len(kf)] = list1
kf
```

Deep Learning Class Project

AGE INVARIANT FACE RECOGNITION

Resources

1) SIFT
   https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html
2) ORB
   https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html
3) GENUINE VS IMPOSTOR HISTOGRAM
   https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.hist.html
4) Matplot
   https://matplotlib.org/3.5.1/api/_as_gen/matplotlib.pyplot.plot.html



Evaluation Metrics

y_actual_score: consists if two pairs are matching then label 1, otherwise label 0.

1.) threshold = 50% -> if a pair matching score is above threshold then they are same(label 1) , otherwise different (label zero)

2) Next, we identify confusion matrix of how many pirs are actually matching and how many are not.

3) plot a frequncy vs matching score plot with 2 legendes:- Matching pairs(=label 1) vs not matching pairs (=label 0)