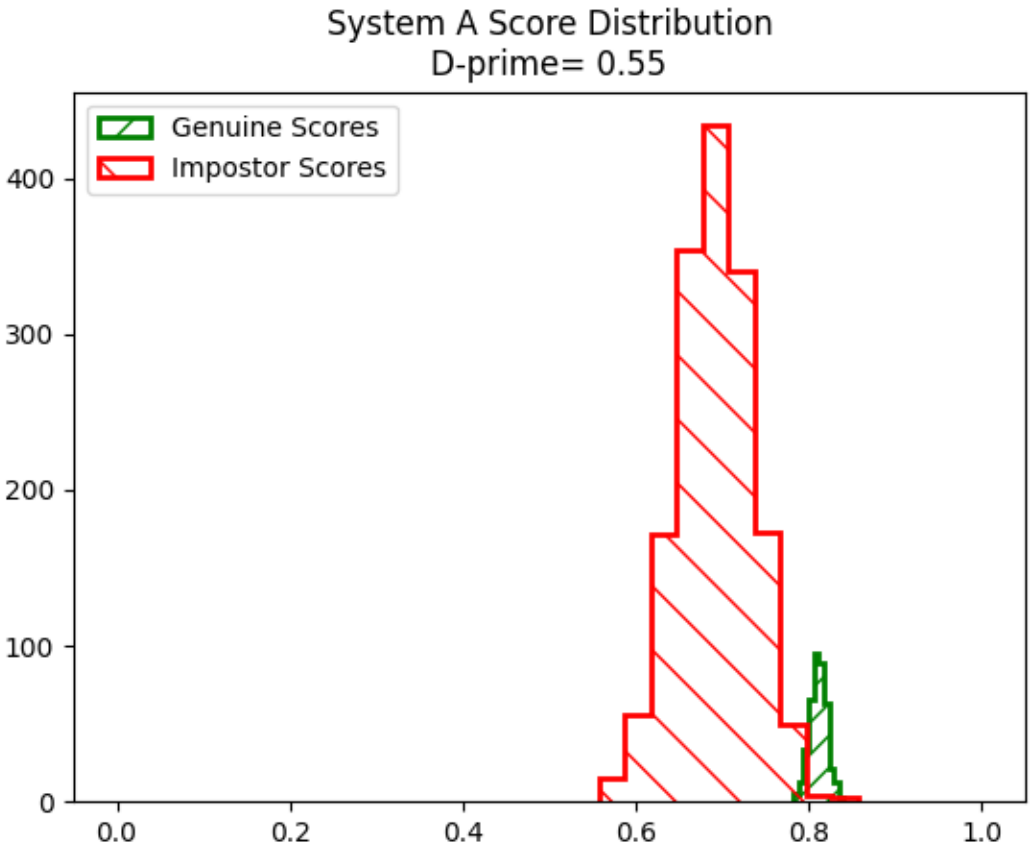
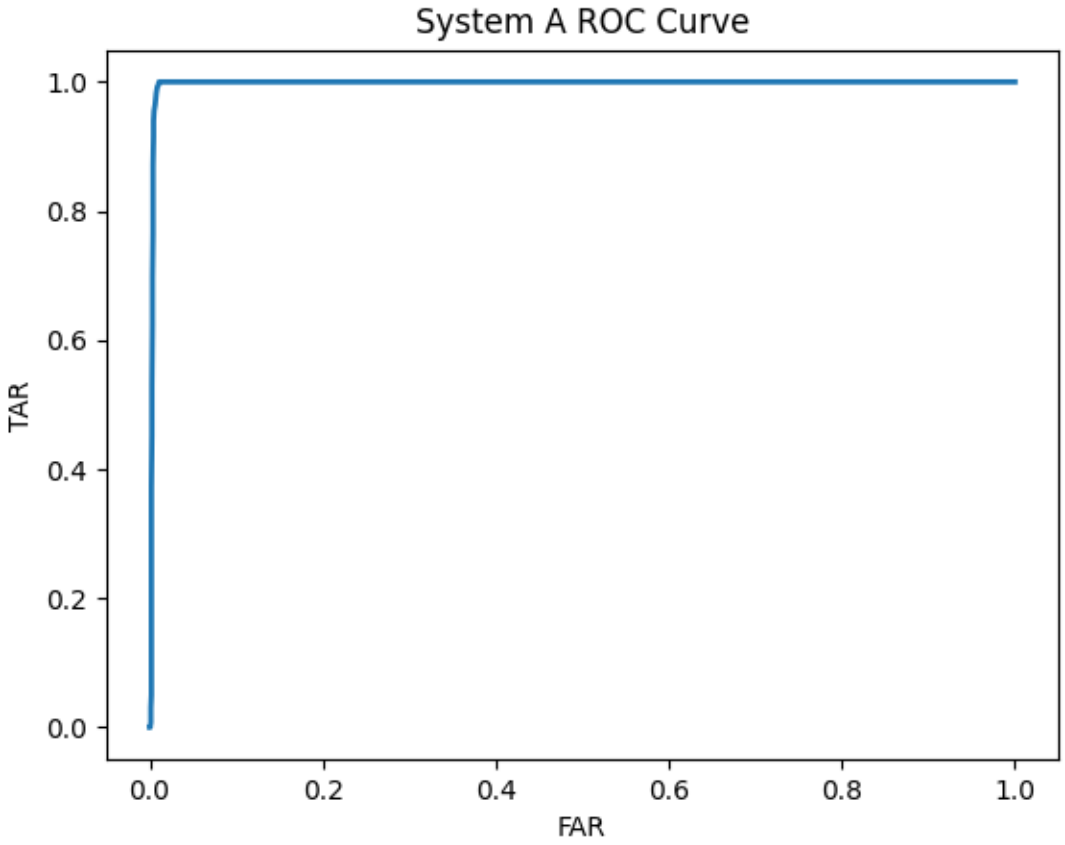
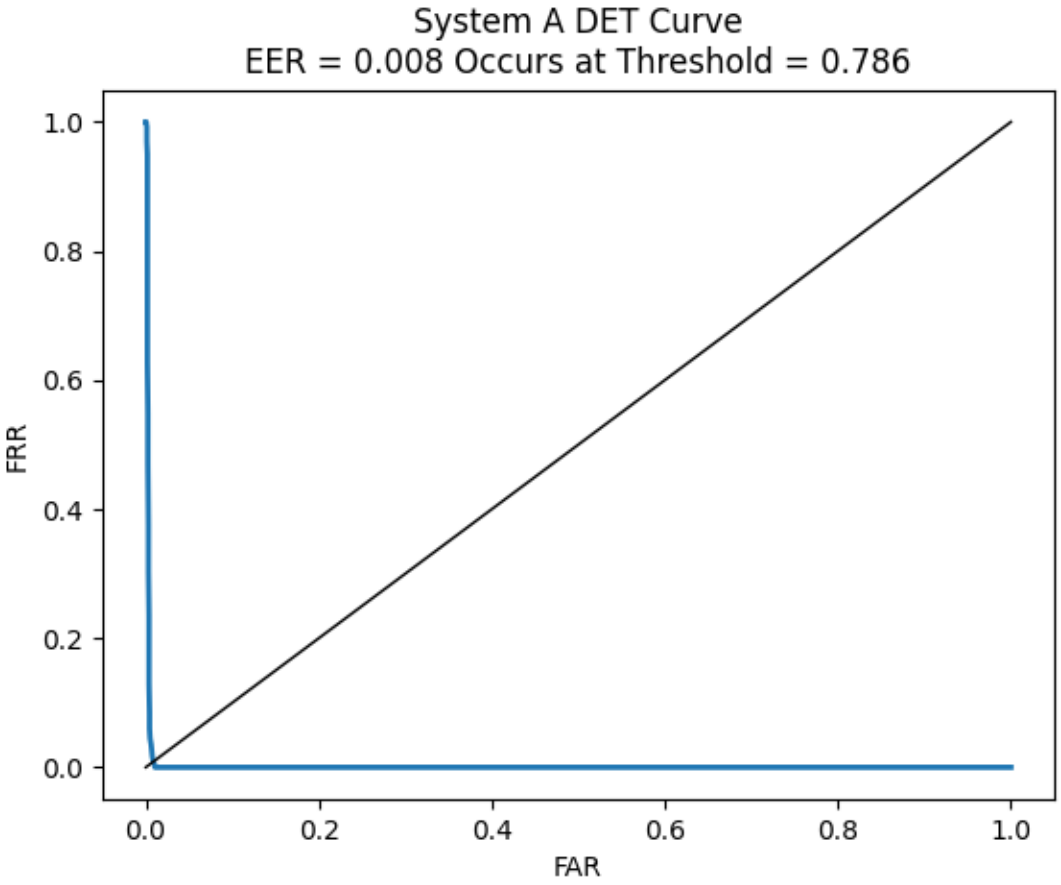
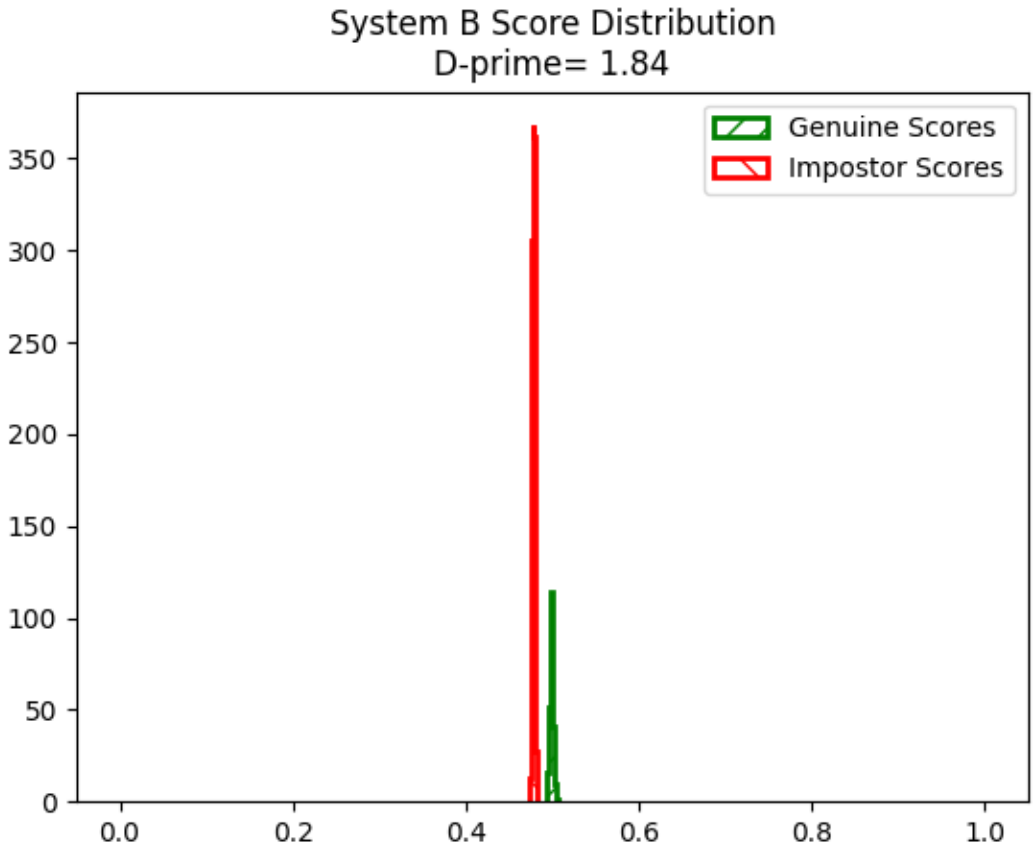
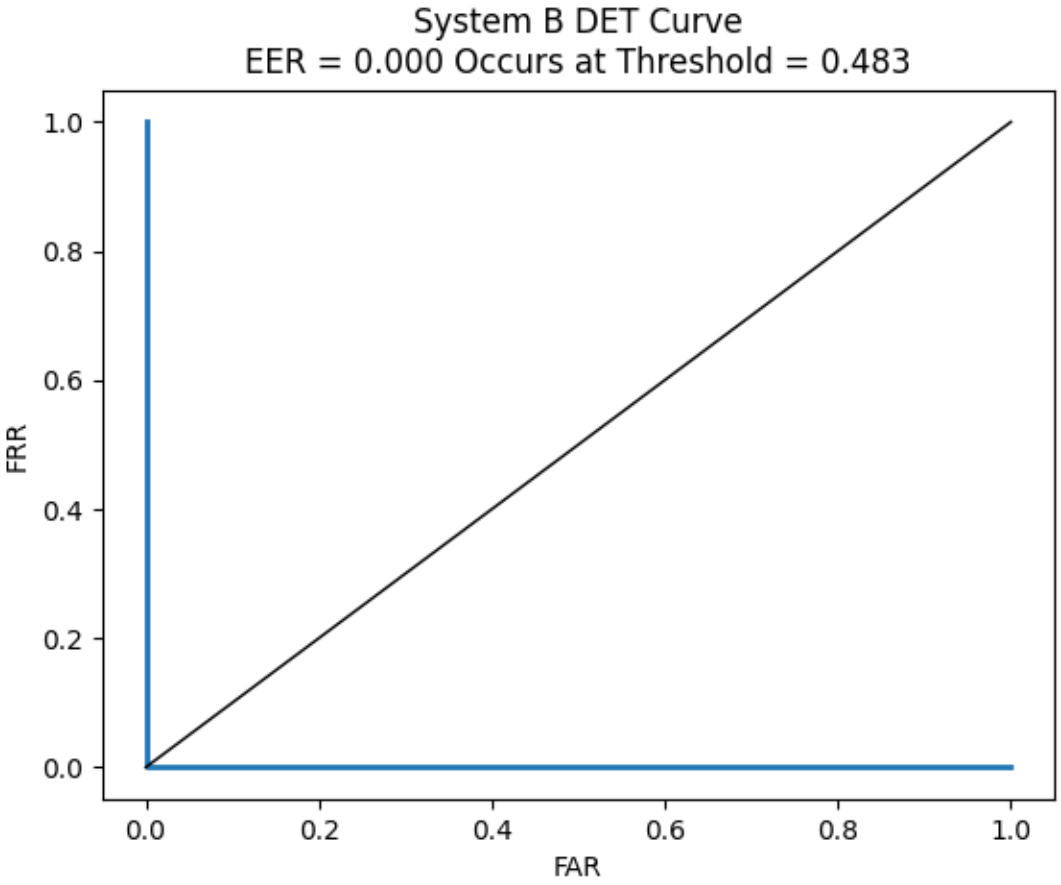
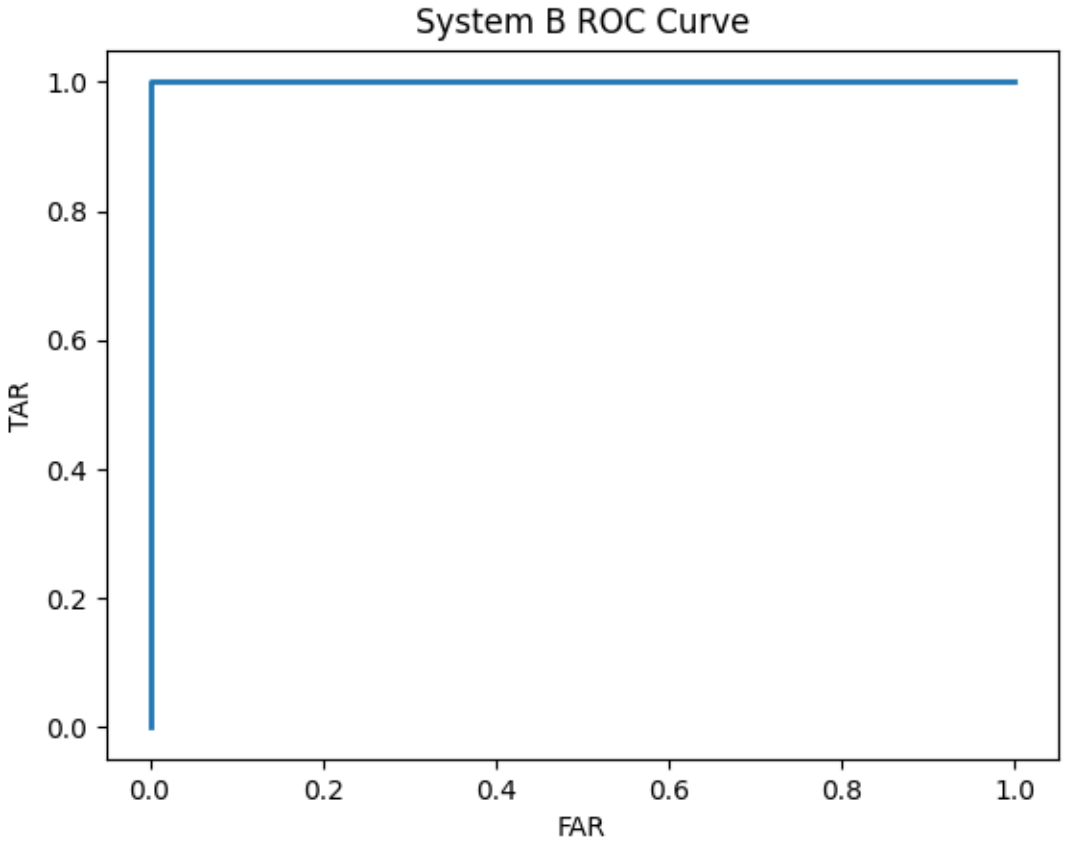

System A Score Distribution
D-prime= 0.55
Genuine Scores
Impostor Scores
400
300
200
100
0
0.0
0.2
0.4
0.6
0.8
1.0

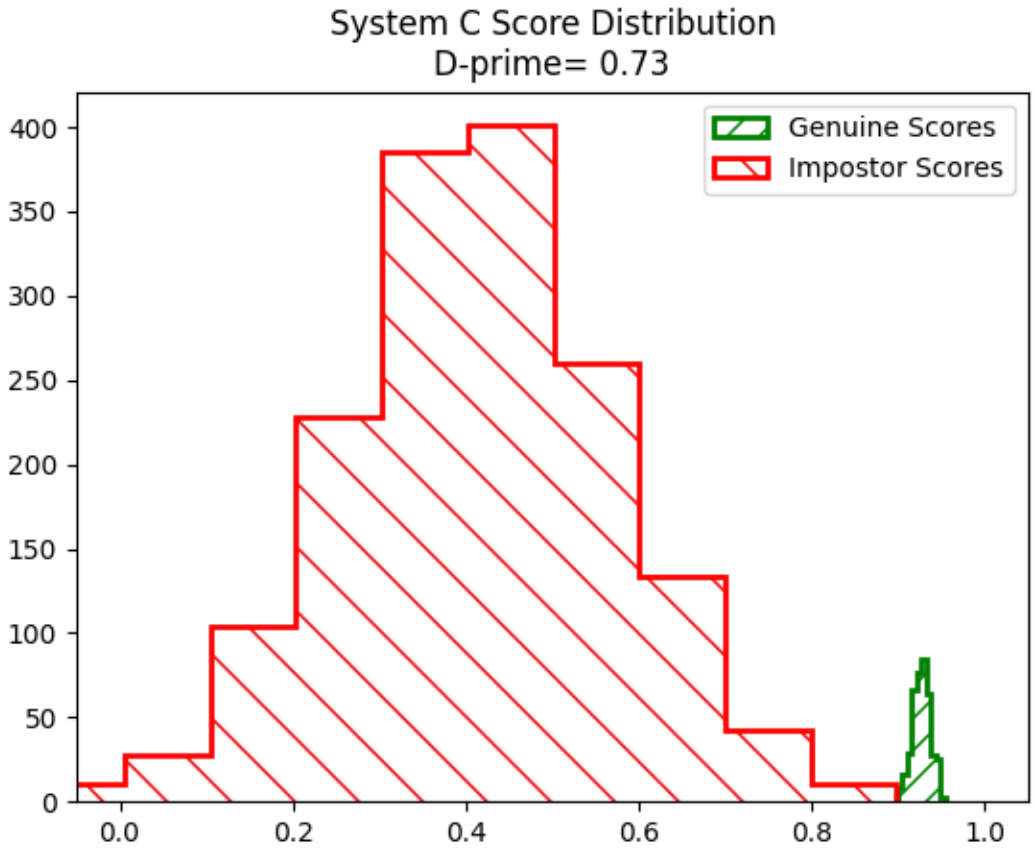

System A ROC Curve
TAR
FAR
0.0
0.2
0.4
0.6
0.8
1.0

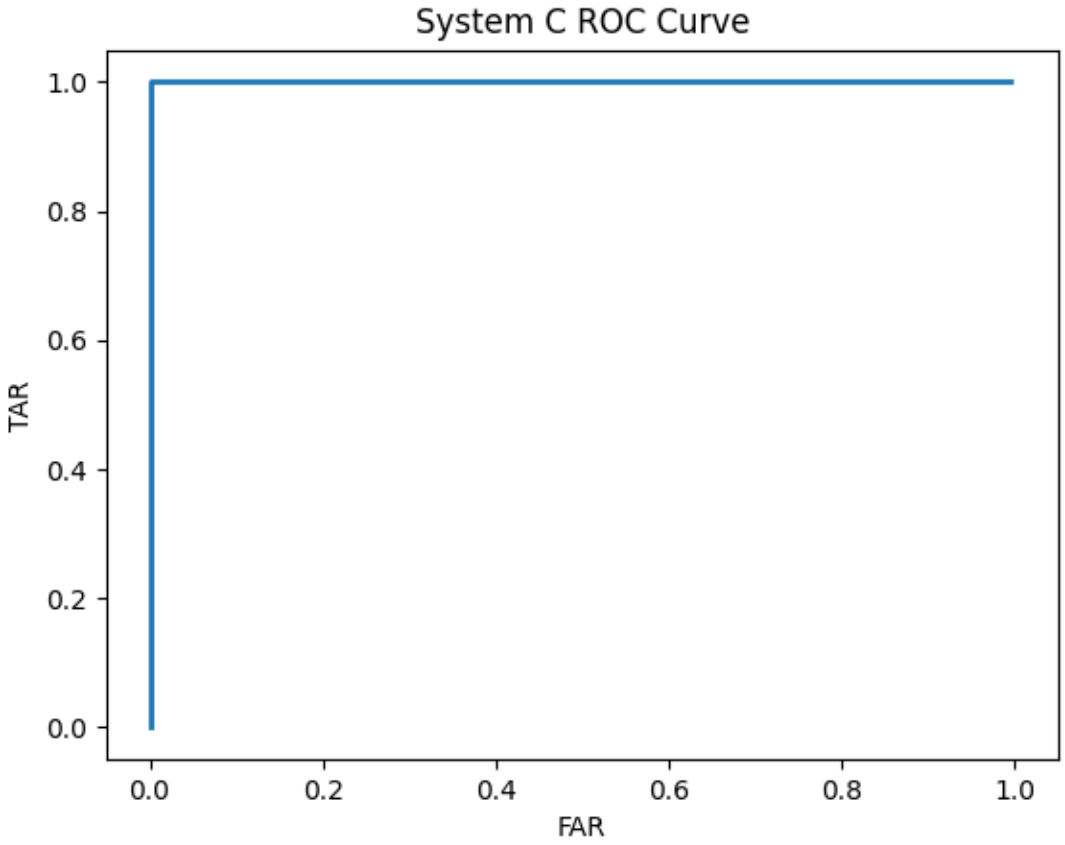

System A DET Curve
EER = 0.008 Occurs at Threshold = 0.786
FRR
FAR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

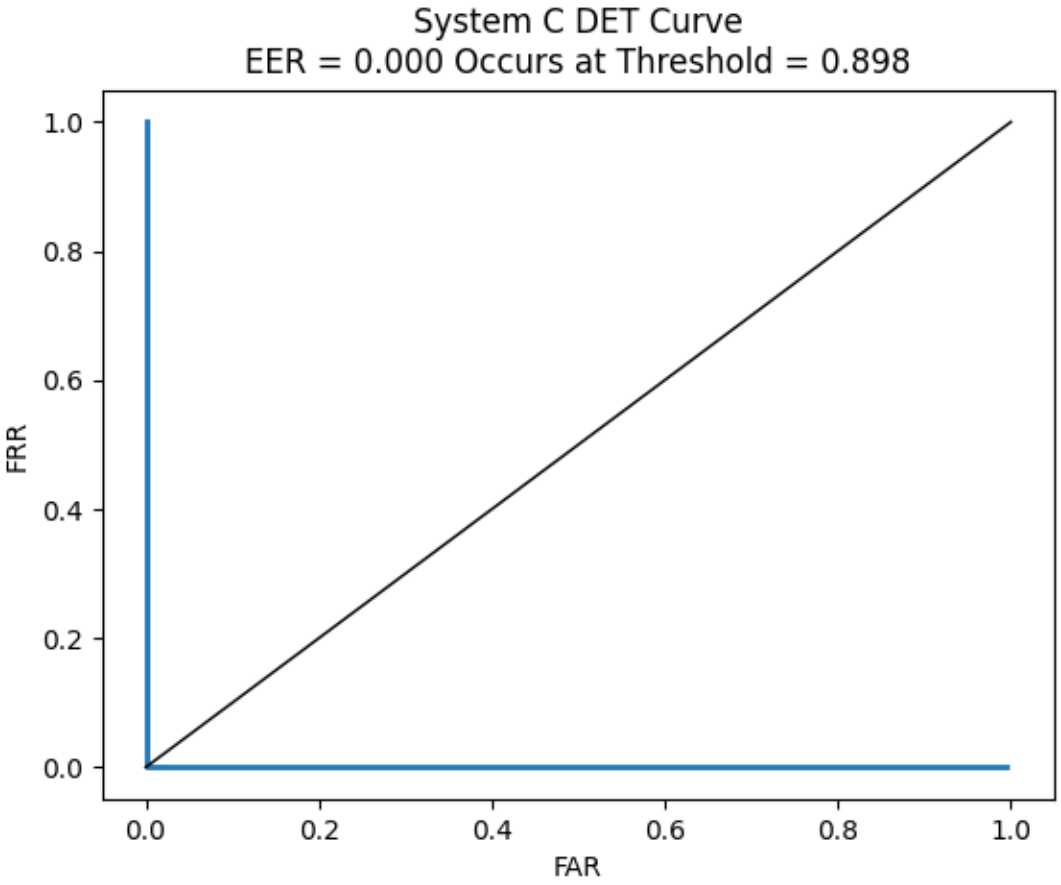

System B Score Distribution
D-prime= 1.84
Genuine Scores
Impostor Scores
350
300
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0


System B DET Curve
EER = 0.000 Occurs at Threshold = 0.483
FRR
FAR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


System B ROC Curve
TAR
FAR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


System C Score Distribution
D-prime= 0.73
Genuine Scores
Impostor Scores
400
350
300
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0


System C ROC Curve
TAR
FAR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


System C DET Curve
EER = 0.000 Occurs at Threshold = 0.898
FRR
FAR
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
import numpy as np
import matplotlib.pyplot as plt

############ functions
#################################################################

def getSpread(arr):
    return np.max(arr) - np.min(arr)

def getMean(arr):
    return np.mean(arr)

def dprime(gen_scores, imp_scores):
    sqrtTwo = np.sqrt(2)
    gen_mean = getMean(gen_scores)
    imp_mean = getMean(imp_scores)
    gen_spread = getSpread(gen_scores)
    imp_spread = getSpread(imp_scores)
    x = sqrtTwo * np.abs(gen_mean - imp_mean)
    y = np.sqrt(gen_spread**2 + imp_spread**2)
    return x / y

def plot_scoreDist(gen_scores, imp_scores, plot_title):
    plt.figure()
    plt.hist(gen_scores, color='green', lw=2, histtype='step', hatch='//',
label='Genuine Scores')
    plt.hist(imp_scores, color='red', lw=2, histtype='step', hatch='\\',
label='Impostor Scores')
    plt.xlim([-0.05,1.05])
    plt.legend(loc='best')
    dp = dprime(gen_scores, imp_scores)
    plt.title(plot_title + ' Score Distribution\nD-prime= %.2f' % dp)
    plt.savefig(plot_title + '_scoreDist.png')
    plt.show(block= False)
    return

def get_EER(far, frr):
    eer = 0
    diff = float('inf')
    bestErrIndex = 0
    for i in range(0,len(far)):
        if abs(far[i] - frr[i]) < diff:
            diff = abs(far[i] - frr[i])
            eer = (far[i] + frr[i]) / 2
            bestErrIndex = i
```

```
    return eer, bestErrIndex

def plot_det(far, frr, plot_title,thresholds):
    eer, eerThresholdIndex = get_EER(far, frr)
    print(far[eerThresholdIndex], frr[eerThresholdIndex])
    plt.figure()
    plt.plot(far, frr, lw=2)
    plt.plot([0,1], [0,1], lw=1, color='black')
    plt.xlim([-0.05,1.05])
    plt.ylim([-0.05,1.05])
    plt.xlabel('FAR')
    plt.ylabel('FRR')
    plt.title(plot_title + ' DET Curve\nEER = %.3f' % eer + ' Occurs at Threshold
= %.3f' % thresholds[eerThresholdIndex])
    plt.savefig(plot_title + '_det.png')
    plt.show(block= False)
    return

def plot_roc(far, tpr, plot_title):
    plt.figure()
    plt.plot(far, tpr, lw=2)
    plt.xlim([-0.05,1.05])
    plt.ylim([-0.05,1.05])
    plt.xlabel('FAR')
    plt.ylabel('TAR')
    plt.title(plot_title + ' ROC Curve')
    plt.savefig(plot_title + '_roc.png')
    plt.show(block= False)
    return

# Function to compute TPR, FAR, FRR
def compute_rates(gen_scores, imp_scores, num_thresholds):
    thresholds = np.linspace(0,1,num_thresholds)
    far = []
    frr = []
    tpr = []

    for t in thresholds:
        tp, fp, tn, fn = 0, 0, 0, 0
        for g_s in gen_scores:
            if g_s >= t:
                tp += 1
            else:
                fn += 1
        for i_s in imp_scores:
```

```
            if i_s >= t:
                fp += 1
            else:
                tn += 1
        far.append(fp/(fp+tn))
        frr.append(fn/(fn+tp))
        tpr.append(tp/(tp+fn))

    return far, frr, tpr, thresholds

def getSystemLetter(n):
    return chr(65 + n)

############ main code
######################################################################

np.random.seed(1)

gen_means = [ 0.8100, 0.4990, 0.9260] # fill these in
gen_stds = [ 0.0100, 0.0020, 0.0098] # fill these in
imp_means = [ 0.6900, 0.4780, 0.4120] # fill these in
imp_stds = [ 0.0420, 0.0014, 0.1550] # fill these in
threshold_count = 500

for i in range(len(gen_means)):
    gen_scores = np.random.normal(gen_means[i],gen_stds[i],400)
    imp_scores = np.random.normal(imp_means[i],imp_stds[i],1600)
    far, frr, tpr, th = compute_rates(gen_scores,imp_scores,threshold_count)
    plot_title = 'System ' + getSystemLetter(i)
    plot_scoreDist(gen_scores, imp_scores, plot_title)
    plot_roc(far, tpr, plot_title)
    plot_det(far, frr, plot_title, th)
plt.show()
```

7.

I would say that system B is the best performing. While both B and C have an EER of 0.000 System B has a bit more of a gap between the imposter and genuine scores which could lead to better performance if this system was updated with new template information during use. System B also has a much higher D Prime value which shows that there is a larger gap between the genuine scores and imposter scores.

8.

Either system B or C may be best for detecting false positive but more so B. This is because both of these systems have a threshold value that will ensure that no false positives occur. However, B has a much higher D prime value.

9.

This could be either C or B, but more so B, may be best for recognizing genuine subjects as genuine. This is because both systems have a wide enough gap between the genuine and imposter scores to ensure that neither of them are incorrectly classifying a template. However, the D prime value of B is much higher than that of system C.

10.

System A is least likely to classify a true negative as such. This can be seen by the fact that the EER is a non-zero number, while the other two have an EER of zero. This is most likely due to the fact that this system has an overlap between the imposter and genuine scores and a very low d prime value when compared to the others.

Also, completely unrelated side comment in reference to System A's Imposter values…. Nice.