HW2

id:110062209
name:簡晟棋

Function Implement

```
def sample_by_label(data, label, number=10, target_label=0):
    new_data = []
    new_label = []
    for i in [0,1]:
        if i == target_label:
            i_data = data[label == i]
            idx = np.random.choice(
                list(range(len(i_data))), min(number,len(i_data)), replace=False
            )
            new_data.append(i_data[idx])
    return new_data
```

sample_by_label:用於隨機抽取指定 label 的 sample
i_data 為指定 label(normal 為 0，anomaly 為 1)的所有 sample
用 np.random.choice 隨機選擇 min(number,len(i_data))個 sample 的編號
(number 可能超過 len(i_data))

problem 1:

```
#problem 1
def visualization(train_data, train_label,test_data, test_label,category):
    draw_normal_1 = sample_by_label(train_data, train_label, number=10, target_label=0)
    draw_anomaly_1 = sample_by_label(test_data, test_label, number=10, target_label=1)
    fig_1, axs_1 = plt.subplots(2,1)
    plt.suptitle(f'{category} Dataset')
    axs_1[0].set_title("Anomaly Sample")
    for i in draw_anomaly_1[0]:
        axs_1[0].plot(list(range(len(draw_anomaly_1[0][0]))),i,color='red')
    axs_1[1].set_title("Normal Sample")
    for i in draw_normal_1[0]:
        axs_1[1].plot(list(range(len(draw_normal_1[0][0]))),i,color='blue')
    plt.tight_layout()
```

用 sample_by_label 從 train_data 抓出 10 個 normal case，並從 test_data 抓出 10 個 anomaly case

用 plt.plot 作圖

problem 2:

```python
#problem 2
def knn(train_data,test_data,test_label,k):
    k_near_distance_list = []
    all_data = np.concatenate((train_data,test_data))
    distance_mat = pairwise_distances(all_data,metric='euclidean')
    for id,test in enumerate(test_data):
        distance_list = distance_mat[len(train_data)+id,:len(train_data)].copy()
        distance_list.sort()
        k_near_distance_list.append(np.mean(distance_list[:k]))
    return roc_auc_score(test_label,k_near_distance_list)
```

沿用 hw1 的 knn(return roc_auc 結果)

problem 3:

```python
#problem 3
def PCA_(train_data, train_label,test_data, test_label,k = 5,N = 5):
    pca = PCA(n_components=N).fit(train_data)
    train_data_pca = pca.transform(train_data)
    test_data_pca = pca.transform(test_data)
    train_data_r = pca.inverse_transform(train_data_pca)
    test_data_r = pca.inverse_transform(test_data_pca)
    test_dis = []
    for i in range(len(test_data_r)):
        test_dis.append(np.sum((test_data[i]-test_data_r[i])**2)**0.5)
    score =  roc_auc_score(test_label,test_dis)


    draw_normal_3 = sample_by_label(train_data_r, train_label, number=10, target_label=0)
    draw_anomaly_3 = sample_by_label(test_data_r, test_label, number=10, target_label=1)
    fig_3, axs_3 = plt.subplots(2,1)
    plt.suptitle(f'{category}PCA={N}')
    axs_3[0].set_title("Anomaly Sample")
    for i in draw_anomaly_3[0]:
        axs_3[0].plot(list(range(len(draw_anomaly_3[0][0]))),i,color='red')
    axs_3[1].set_title("Normal Sample")
    for i in draw_normal_3[0]:
        axs_3[1].plot(list(range(len(draw_normal_3[0][0]))),i,color='blue')
    plt.tight_layout()
    return score
```

用 train_data 去 fit N 個 component 的 pca

對 train_data 與 test_data 做 pca，結果為 train_data_pca 與 test_data_pca

對 train_data_pca 與 test_data_pca 做 pca.inverse_transform，結果為 train_data_r 與 test_data_r

算出 test_data 與 test_data_r 的 reconstruction error，以此算 roc_auc

用 sample_by_label 從 train_data_r 抓出 10 個 normal case，並從 test_data_r 抓出 10 個 anomaly case

用 plt.plot 作圖

return roc_auc 結果

problem 4:

```python
#problem 4
def discrete_fourier_transform(train_data, train_label,test_data, test_label,k = 5,M=20):

    select_index = []
    tmp = []
    for i in range(int(M/2)):
        select_index.append(i)
    if M%2 == 1:
        select_index.append(int(M/2))
    for i in range(int(M/2)):
        tmp.append(train_data.shape[1]-1-i)
    tmp.reverse()

    select_index.extend(tmp)

    train_data_fft = np.array([fft(row) for row in train_data])
    train_data_fft_select = train_data_fft[:,select_index].copy()
    train_data_fft_magnitude = np.abs(train_data_fft_select)

    test_data_fft = np.array([fft(row) for row in test_data])
    test_data_fft_select = test_data_fft[:,select_index].copy()
    test_data_fft_magnitude = np.abs(test_data_fft_select)

    score =  knn(train_data_fft_magnitude,test_data_fft_magnitude,test_label,k)
    train_data_ifft = np.zeros(train_data.shape, dtype=np.complex128)
    train_data_ifft[:,select_index] = train_data_fft_select

    test_data_ifft = np.zeros(test_data.shape, dtype=np.complex128)
    test_data_ifft[:,select_index] = test_data_fft_select


    train_data_ifft_ = np.array([np.real(ifft(row)) for row in train_data_ifft])
    test_data_ifft_ = np.array([np.real(ifft(row)) for row in test_data_ifft])


    draw_normal_4 = sample_by_label(train_data_ifft_, train_label, number=10, target_label=0)
    draw_anomaly_4 = sample_by_label(test_data_ifft_, test_label, number=10, target_label=1)
    fig_4, axs_4 = plt.subplots(2,1)
    plt.suptitle(f'{category}DFT={M}')
    axs_4[0].set_title("Anomaly Sample")
    for i in draw_anomaly_4[0]:
        axs_4[0].plot(list(range(len(draw_anomaly_4[0][0]))),i,color='red')
    axs_4[1].set_title("Normal Sample")
    for i in draw_normal_4[0]:
        axs_4[1].plot(list(range(len(draw_normal_4[0][0]))),i,color='blue')
    plt.tight_layout()
    return score
```

對 train_data 與 test_data 做 fft，結果為 train_data_fft 與 test_data_fft

train_data_fft_select 與 test_data_fft_select 是 train_data_fft 與 test_data_fft 每個 case 的 lowest M frequency 的項目

train_data_fft_magnitude 與 test_data_fft_magnitude 是 train_data_fft_select 與 test_data_fft_select 的 magnitude(絕對值)

將 train_data_fft_magnitude 與 test_data_fft_magnitude 用 knn 算 roc_auc_score

先使 train_data_ifft 與 test_data_ifft 每個 case 的 lowest M frequency 的項目為 train_data_fft 與 test_data_fft，後面的值補 0，之後對 train_data_ifft 與 test_data_ifft 做 fft

用 sample_by_label 從 train_data_ifft 抓出 10 個 normal case，並從 test_data_ifft 抓出 10 個 anomaly case

用 plt.plot 作圖

return knn 的 roc_auc 結果

problem 5:

```
def haar(data,dir):
    if data.shape[1] == 2:
        left = (data[:,0]+data[:,1])/2
        right = (data[:,1]-data[:,0])/2
        left = np.reshape(left,(data.shape[0],1))
        right = np.reshape(right,(data.shape[0],1))
        return np.concatenate((left,right),axis=1)
    elif dir == "right":
        right = np.zeros((data.shape[0],data.shape[1]//2))
        for i in range(data.shape[1]//2):
            right[:,i]=(data[:,i*2+1]-data[:,i*2])/2
        return right
    else:
        left = np.zeros((data.shape[0],data.shape[1]//2))
        right = haar(data,"right")
        for i in range(data.shape[1]//2):
            left[:,i]=(data[:,i*2+1]+data[:,i*2])/2
        left_new = left.copy()
        left = haar(left_new,"left")
        return np.concatenate((left,right),axis=1)
```

haar:用遞迴式處理 haar wavelet function

data 只有兩項時，回傳($A_{\text{level 1}}$, $D_{\text{level 1}}$)

否則用 dir == “right”算 right = [$D_{\text{level i}}$]，用 dir==”left”算 left = [$A_{\text{level i}}$]，把 left 遞迴下去後把遞迴結果跟 right 合併在一起後 return

```
#problem 5
def discrete_wavelet_transform(train_data, train_label,test_data, test_label,k = 5,S=32):
    level=np.ceil(np.log2(train_data.shape[1]))
    L = int(2**level)
    train_data_haar = np.zeros((train_data.shape[0],L))
    train_data_haar[:,:train_data.shape[1]] = train_data
    train_data_haar = haar(train_data_haar,"left")
    test_data_haar = np.zeros((test_data.shape[0],L))
    test_data_haar[:,:test_data.shape[1]] = test_data
    test_data_haar = haar(test_data_haar,"left")
    score =  knn(train_data_haar[:,:S],test_data_haar[:,:S],test_label,k)
    return score
```

level=np.ceil(np.log2(train_data.shape[1]))為所需 level 數

L = int(2**level)為所需長度

先使 train_data_haar 與 test_data_haar 為 train_data 與 test_data 每個 case 補 0 到所需長度 L，接著用 haar function 算出 train_data_haar 與 test_data_haar 經過 haar wavelet function 處理後的結果

將 train_data_haar 與 test_data_haar 用 knn 算 roc_auc_score

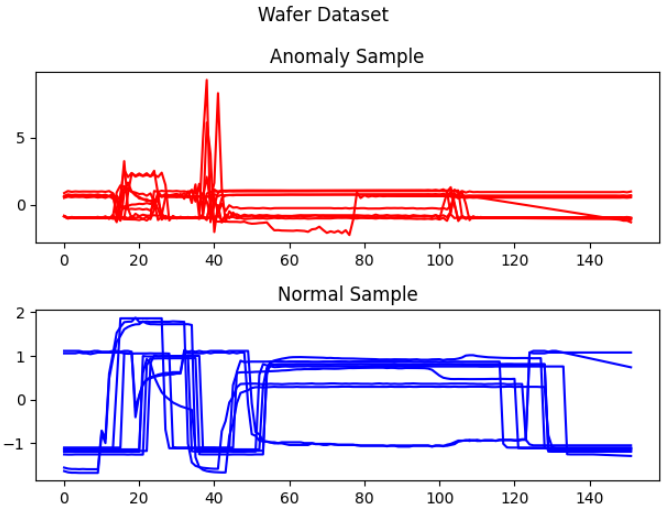
return knn 的 roc_auc 結果

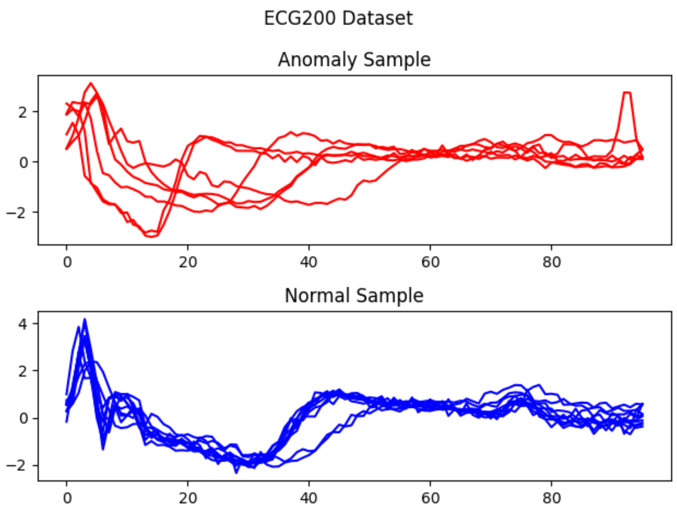
Calculate, Record, Drawing

```python
print("-------problem 1-------")
visualization(train_data, train_label,test_data, test_label,category)

print("-------problem 2-------")
k_ = [2,5,7]
for k in k_:
    print(f"when k={k}, roc_auc_score = {knn(train_data,test_data,test_label,k)}")

print("-------problem 3-------")
N_ = [1,2,5,10]
for N in N_:
    print(f"N={N}, roc_auc_score = {PCA_(train_data, train_label,test_data, test_label,k=k,N=N)}")

print("-------problem 4-------")

M_ = [10,15,20,25]
for k in k_:
    for M in M_:
        print(f"when k={k},M={M}, roc_auc_score = {discrete_fourier_transform(train_data, train_label,test_data, test_label,k = k,M=M)}")

print("-------problem 5-------")
S_ = [8,16,32,64]
for k in k_:
    for S in S_:
        print(f"when k={k},S={S}, roc_auc_score = {discrete_wavelet_transform(train_data, train_label,test_data, test_label,k = k,S=S)}")
plt.show()
```

用 plt.show()把前面的 plt 部分畫出來

problem 1 作圖結果





problem 2 結果:

Wafer:

```
when k=5, roc_auc_score = 0.9884085564820364
```

ECG200:

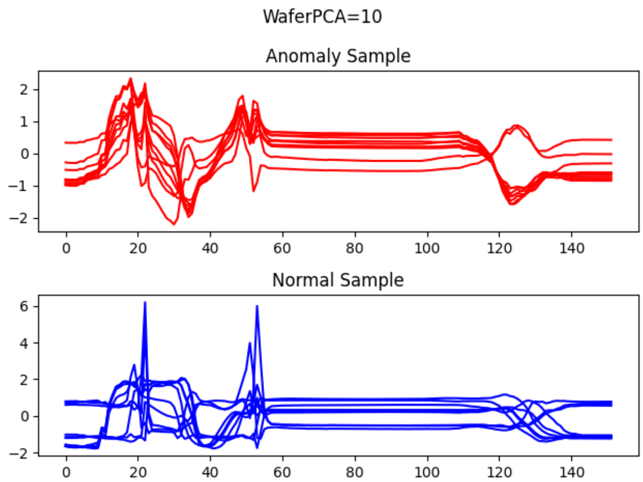
```
when k=5, roc_auc_score = 0.921875
```

problem 3 結果:

分別在 N=1,2,5,10 測試，發現 Wafer 效果在 N=10 效果最好，ECG200 則在 N=5 效果最好

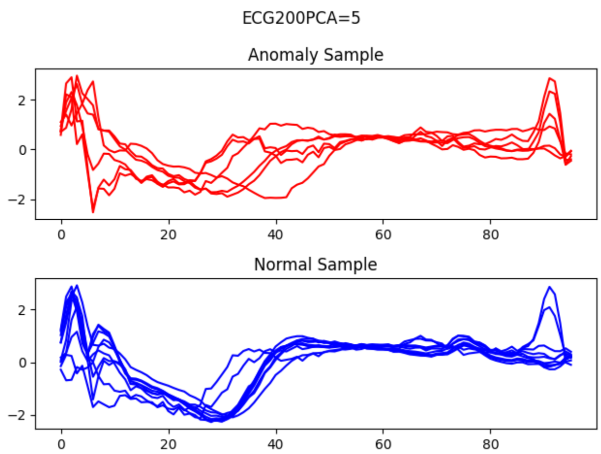
Wafer:

```
-------problem 3-------
N=1, roc_auc_score = 0.9601901455174331
N=2, roc_auc_score = 0.9367787022710868
N=5, roc_auc_score = 0.9552139137071123
N=10, roc_auc_score = 0.9740691385848925
```



ECG200:

```
-------problem 3-------
N=1, roc_auc_score = 0.8177083333333333
N=2, roc_auc_score = 0.9088541666666667
N=5, roc_auc_score = 0.9479166666666667
N=10, roc_auc_score = 0.9036458333333334
```

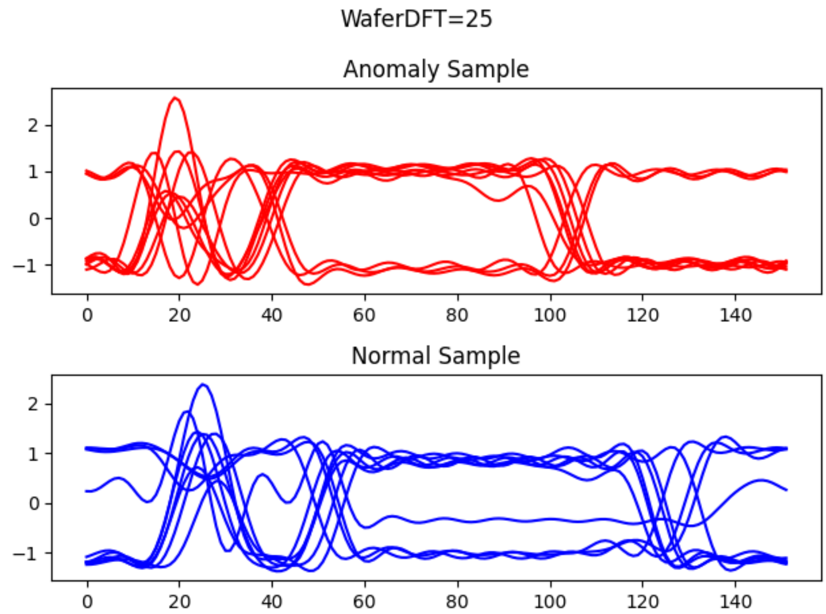


problem 4 結果:

分別在 M=10,15,20,25 測試，發現 Wafer 效果都差不多(M=25 時比其他 M 值好一點點)，ECG200 則在 M=25 效果最好

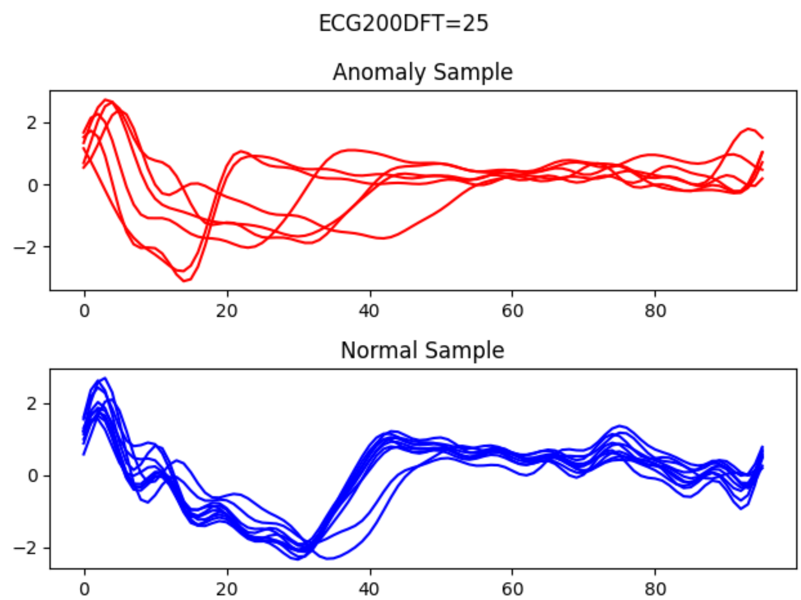
Wafer:

```
when k=5,M=10, roc_auc_score = 0.9942933820389931
when k=5,M=15, roc_auc_score = 0.9962135854473955
when k=5,M=20, roc_auc_score = 0.997174846494693
when k=5,M=25, roc_auc_score = 0.9982318361576588
```



ECG200:

```
when k=5,M=10, roc_auc_score = 0.8229166666666667
when k=5,M=15, roc_auc_score = 0.8541666666666667
when k=5,M=20, roc_auc_score = 0.8541666666666667
when k=5,M=25, roc_auc_score = 0.8567708333333334
```



problem 5 結果:
分別在 S=8,16,32,64 測試，發現 Wafer 效果都差不多(S=8 時比比其他 S 值好一點點)，ECG200 則在 S=32 效果最好
Wafer:

```
when k=5,S=8, roc_auc_score = 0.9985899075539815
when k=5,S=16, roc_auc_score = 0.9976922447565396
when k=5,S=32, roc_auc_score = 0.9979718120631969
when k=5,S=64, roc_auc_score = 0.9973772346752232
```

ECG200:

```
when k=5,S=8, roc_auc_score = 0.8828125
when k=5,S=16, roc_auc_score = 0.9401041666666667
when k=5,S=32, roc_auc_score = 0.9479166666666666
when k=5,S=64, roc_auc_score = 0.9427083333333334
```

Observation

problem 6:

在 Wafer 中，DWT(0.999)> DFT(0.998)>original knn(0.988)> PCA(0.974)
以下是可能原因:
Discrete Wavelet Transform:可以有效降維
Discrete Fourier Transform:資料週期性較佳
original : noise 與維度太多
PCA:變異性差異不大

在 ECG200 中，PCA(0.948) ≈ DWT(0.948) > original knn(0.922) > DFT(0.857)
以下是可能原因:
PCA:可以有效地捕捉資料中的主要變異性
Discrete Wavelet Transform:可以有效降維
original : noise 與維度太多
Discrete Fourier Transform:資料週期性較差

problem 7:

Wafer

original:

```
-------problem 2-------
when k=2, roc_auc_score = 0.9896026798712533
when k=5, roc_auc_score = 0.9884085564820364
when k=7, roc_auc_score = 0.9862919272290276
```

DFT:

```
-------problem 4-------
when k=2,M=10, roc_auc_score = 0.9942195153217128
when k=2,M=15, roc_auc_score = 0.9958035092321803
when k=2,M=20, roc_auc_score = 0.9976607768724964
when k=2,M=25, roc_auc_score = 0.9982722475455879
when k=5,M=10, roc_auc_score = 0.9942933820389931
when k=5,M=15, roc_auc_score = 0.9962135854473955
when k=5,M=20, roc_auc_score = 0.997174846494693
when k=5,M=25, roc_auc_score = 0.9982318361576588
when k=7,M=10, roc_auc_score = 0.9944146162027803
when k=7,M=15, roc_auc_score = 0.996242734645246
when k=7,M=20, roc_auc_score = 0.9969886891175113
when k=7,M=25, roc_auc_score = 0.9981702253531111
```

DWT:

```
-------problem 5-------
when k=2,S=8, roc_auc_score = 0.9973325171557935
when k=2,S=16, roc_auc_score = 0.9979459752741929
when k=2,S=32, roc_auc_score = 0.9977108800870898
when k=2,S=64, roc_auc_score = 0.9972815060595551
when k=5,S=8, roc_auc_score = 0.9985899075539815
when k=5,S=16, roc_auc_score = 0.9976922447565396
when k=5,S=32, roc_auc_score = 0.9979718120631969
when k=5,S=64, roc_auc_score = 0.9973772346752232
when k=7,S=8, roc_auc_score = 0.9986419123728739
when k=7,S=16, roc_auc_score = 0.9972705751103611
when k=7,S=32, roc_auc_score = 0.9977816797953992
when k=7,S=64, roc_auc_score = 0.9968869981659192
```

best combination:

original : k=2

DFT : k=2, M=25

DWT : k=7, S=8

ECG200:

original:

```
-------problem 2-------
when k=2, roc_auc_score = 0.9557291666666666
when k=5, roc_auc_score = 0.921875
when k=7, roc_auc_score = 0.90625
```

DFT:

```
-------problem 4-------
when k=2,M=10, roc_auc_score = 0.8177083333333334
when k=2,M=15, roc_auc_score = 0.8671875
when k=2,M=20, roc_auc_score = 0.8411458333333334
when k=2,M=25, roc_auc_score = 0.8463541666666667
when k=5,M=10, roc_auc_score = 0.8229166666666667
when k=5,M=15, roc_auc_score = 0.8541666666666667
when k=5,M=20, roc_auc_score = 0.8541666666666667
when k=5,M=25, roc_auc_score = 0.8567708333333334
when k=7,M=10, roc_auc_score = 0.8177083333333334
when k=7,M=15, roc_auc_score = 0.8541666666666666
when k=7,M=20, roc_auc_score = 0.8515625
when k=7,M=25, roc_auc_score = 0.8619791666666667
```

DWT:

```
-------problem 5-------
when k=2,S=8, roc_auc_score = 0.9088541666666667
when k=2,S=16, roc_auc_score = 0.953125
when k=2,S=32, roc_auc_score = 0.9635416666666666
when k=2,S=64, roc_auc_score = 0.9635416666666666
when k=5,S=8, roc_auc_score = 0.8828125
when k=5,S=16, roc_auc_score = 0.9401041666666667
when k=5,S=32, roc_auc_score = 0.9479166666666666
when k=5,S=64, roc_auc_score = 0.9427083333333334
when k=7,S=8, roc_auc_score = 0.8697916666666666
when k=7,S=16, roc_auc_score = 0.9088541666666667
when k=7,S=32, roc_auc_score = 0.9088541666666666
when k=7,S=64, roc_auc_score = 0.9244791666666667
```

best combination:

original : k=2

DFT : k=2, M=15

DWT : k=2, S=32