In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import cluster, metrics
from sklearn.cluster import KMeans
from sklearn.utils import shuffle
from pandas.plotting import scatter_matrix
import matplotlib.pyplot as plt
```

In [2]:

```
def reorder(x):
    col = x.columns.tolist()
    col = col[1:] + [col[0]]
    reorder_data = x[col]
    return reorder_data
```

In [3]:

```
path = "../bankruptcy_HW1_train.csv"
data = pd.read_csv(path).drop(["Unnamed: 0"],axis=1)
data = reorder(data)
data
```

Out[3]:

	Net worth/Assets	Debt ratio %	Per Share Net profit before tax (Yuan ¥)	ROA(C) before interest and depreciation before interest	Bankrupt?
0	0.888200	0.111800	0.184454	0.500951	0
1	0.847689	0.152311	0.166755	0.479647	0
2	0.966895	0.033105	0.181969	0.516502	0
3	0.906310	0.093690	0.167960	0.446302	0
4	0.882396	0.117604	0.179935	0.575245	0
...	...	...	...	...	...
9257	0.824272	0.175728	0.161288	0.467220	1
9258	0.786659	0.213341	0.174660	0.494583	1
9259	0.806011	0.193989	0.193313	0.494676	1
9260	0.872011	0.127989	0.155082	0.417124	1
9261	0.837775	0.162225	0.166181	0.463788	1

9262 rows × 5 columns

In [4]:

```
data_without_label = data.drop(["Bankrupt?"],axis=1)
data_without_label = data_without_label
data_without_label
```

Out[4]:

	Net worth/Assets	Debt ratio %	Per Share Net profit before tax (Yuan ¥)	ROA(C) before interest and depreciation before interest
0	0.888200	0.111800	0.184454	0.500951
1	0.847689	0.152311	0.166755	0.479647
2	0.966895	0.033105	0.181969	0.516502
3	0.906310	0.093690	0.167960	0.446302
4	0.882396	0.117604	0.179935	0.575245
...	...	...	...	...
9257	0.824272	0.175728	0.161288	0.467220
9258	0.786659	0.213341	0.174660	0.494583
9259	0.806011	0.193989	0.193313	0.494676
9260	0.872011	0.127989	0.155082	0.417124
9261	0.837775	0.162225	0.166181	0.463788

9262 rows × 4 columns

Use kmeans to get the membership function

In [5]:

```
kmeans_fit = cluster.KMeans(n_clusters = 2 ).fit(data_without_label)
cluster_labels = kmeans_fit.labels_
cluster_labels = pd.DataFrame(cluster_labels, columns = ['predict'])
cluster_labels
```

Out[5]:

	predict
0	1
1	0
2	1
3	1
4	1
...	...
9257	0
9258	0
9259	0
9260	0
9261	0

9262 rows × 1 columns

In [6]:

```
cluster1_index = cluster_labels["predict"]==1
cluster2_index = cluster_labels["predict"]==0
cluster1_data = data[cluster1_index]
cluster2_data = data[cluster2_index]

# cluster1_data = data[data["Bankrupt?"]==1]
# cluster2_data = data[data["Bankrupt?"]==0]
```

In [7]:

```
print(np.array(cluster1_data['Bankrupt?']==1, dtype=float).sum())
print(np.array(cluster1_data['Bankrupt?']==0, dtype=float).sum())
```

```
398.0
3721.0
```

In [8]:

```
cluster1_mean = cluster1_data.mean()
cluster2_mean = cluster2_data.mean()
cluster1_std = cluster1_data.std()
cluster2_std = cluster2_data.std()


ship = {"c1mean":cluster1_mean, "c2mean":cluster2_mean, "c1_std":cluster1_std, "c2_std":clu
```

In [9]:

```
# np.save("membership.npy", membership)
# x = np.load("./membership.npy", allow_pickle=True)
print(ship)
```

```
{'c1mean':  Net worth/Assets                                          0.904
852
 Debt ratio %                                              0.095552
 Per Share Net profit before tax (Yuan ¥)                  0.187574
 ROA(C) before interest and depreciation before interest   0.513694
Bankrupt?                                                  0.096625
dtype: float64, 'c2mean':  Net worth/Assets
0.808247
 Debt ratio %                                              0.191845
 Per Share Net profit before tax (Yuan ¥)                  0.150364
 ROA(C) before interest and depreciation before interest   0.425470
Bankrupt?                                                  0.823060
dtype: float64, 'c1_std':  Net worth/Assets
0.039682
 Debt ratio %                                              0.039850
 Per Share Net profit before tax (Yuan ¥)                  0.030294
 ROA(C) before interest and depreciation before interest   0.057172
Bankrupt?                                                  0.295483
dtype: float64, 'c2_std':  Net worth/Assets
0.045154
 Debt ratio %                                              0.045134
 Per Share Net profit before tax (Yuan ¥)                  0.024471
 ROA(C) before interest and depreciation before interest   0.068658
Bankrupt?                                                  0.381655
dtype: float64}
```

use anfis

In [10]:

```
import anfis
from anfis import predict
import membership.mfDerivs
import membership.membershipfunction
from sklearn.metrics import classification_report
```

test/train

In [11]:

```
test_data = pd.read_csv("../bankruptcy_HW1_test.csv").drop(["Unnamed: 0"],axis=1)
test = reorder(test_data)
test = np.array(shuffle(test))
X_test, y_test = test[:, 0:4], test[:, 4]

train = np.array(data)
train = shuffle(train)
X, Y = train[:, 0:4], train[:, 4]
```

In [12]:

```
mf = [
        [['gaussmf',{'mean':ship["c1mean"][0],'sigma':ship["c1_std"][0]}],['gaussmf',{'mean
        [['gaussmf',{'mean':ship["c1mean"][1],'sigma':ship["c1_std"][1]}],['gaussmf',{'mean
        [['gaussmf',{'mean':ship["c1mean"][2],'sigma':ship["c1_std"][2]}],['gaussmf',{'mean
        [['gaussmf',{'mean':ship["c1mean"][3],'sigma':ship["c1_std"][3]}],['gaussmf',{'mean
```

In [13]:

```
EPOCH = 3
mfc = membership.membershipfunction.MemFuncs(mf)
anf = anfis.ANFIS(X, Y, mfc)
anf.trainHybridJangOffLine(epochs=EPOCH)
```

```
current error: 816.810652999799
current error: 815.2312171667468
```

Out[13]:

```
array([[ 0.97197622],
       [ 0.99855612],
       [ 0.90909873],
       ...,
       [-0.09342046],
       [ 0.01243939],
       [-0.08286484]])
```

In [14]:

```
def prediction(anfis, X):
    y_pred = np.squeeze(predict(anf, X), axis=1)
    y_pred[y_pred >= 0.5] = 1
    y_pred[y_pred < 0.5] = 0
    return y_pred
```

In [15]:

```
y_test_pred = prediction(anfis, X_test)
y_train_pred = prediction(anfis, X)
```

In [17]:

```
from sklearn.metrics import accuracy_score
print("Accuracy_train:",accuracy_score(Y, y_train_pred))
print(classification_report(Y, y_train_pred))
print("Accuracy_train:",accuracy_score(y_test, y_test_pred))
print(classification_report(y_test, y_test_pred))
```

```
Accuracy_train: 0.8689267976678903
              precision    recall  f1-score   support

         0.0       0.86      0.88      0.87      4631
         1.0       0.88      0.86      0.87      4631

    accuracy                           0.87      9262
   macro avg       0.87      0.87      0.87      9262
weighted avg       0.87      0.87      0.87      9262

Accuracy_train: 0.8719452590420332
              precision    recall  f1-score   support

         0.0       0.99      0.87      0.93      1968
         1.0       0.21      0.83      0.33        78

    accuracy                           0.87      2046
   macro avg       0.60      0.85      0.63      2046
weighted avg       0.96      0.87      0.91      2046
```