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 = "../daimonds_HW1_train.csv"
data = pd.read_csv(path).drop(["Unnamed: 0"],axis=1)
data = reorder(data)
data
```

Out[3]:

	carat	x	y	z	price
0	-0.143474	0.168226	0.126436	-0.111555	3002.0
1	0.618050	0.828666	0.841599	0.680407	5845.0
2	0.554590	0.748342	0.658447	0.680407	2968.0
3	-1.010766	-1.152654	-1.112018	-1.172217	800.0
4	-0.841539	-0.920607	-0.867816	-0.861090	579.0
...	...	...	...	...	...
31336	1.168040	1.194585	1.207902	1.104672	5546.0
31337	-0.206935	-0.001347	-0.030551	0.044009	2751.0
31338	-0.503083	-0.394041	-0.370690	-0.450967	1723.0
31339	-0.122321	0.007578	0.021778	0.100578	2906.0
31340	0.469976	0.623394	0.579954	0.708691	7797.0

31341 rows × 5 columns

In [4]:

```
#隨即取出1/10資料
data_random_select_3000 = data.sample(n=3000)
data_random_select_3000=data_random_select_3000.set_index(pd.Index([i for i in range(3000)]
```

In [5]:

```
# data_without_label = data_random_select_3000.drop(["price"],axis=1)
# data_without_label = data_without_label
# data_without_label
data_without_label = data_random_select_3000
data_without_label
```

Out[5]:

	carat	x	y	z	price
0	0.914199	1.025013	0.963700	0.991534	8381.0
1	-0.101167	0.078977	0.126436	0.128862	2500.0
2	-0.904999	-1.009856	-0.955031	-1.002511	649.0
3	-0.524237	-0.411890	-0.449183	-0.422683	1415.0
4	-1.053073	-1.268677	-1.225397	-1.242928	658.0
...	...	...	...	...	...
2995	-0.587697	-0.447590	-0.475348	-0.578246	1806.0
2996	2.564168	2.060297	2.062610	2.066339	17024.0
2997	-0.206935	0.034353	0.047942	-0.168124	2453.0
2998	-0.820385	-0.902758	-0.841651	-0.762095	738.0
2999	-0.185781	-0.037046	-0.004387	0.029867	2326.0

3000 rows × 5 columns

Use kmeans to get the membership function

In [6]:

```
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
```

	predict
0	1
1	0
2	0
3	0
4	0
...	...
2995	0
2996	1
2997	0
2998	0
2999	0

In [7]:

```
cluster1_index = cluster_labels["predict"]==1
cluster2_index = cluster_labels["predict"]==0
```

In [8]:

```
np.array(cluster_labels["predict"]==1, dtype=float).sum()
```

Out[8]:

```
564.0
```

In [9]:

```
cluster2_index
```

Out[9]:

```
0       False
1        True
2        True
3        True
4        True
        ...
2995     True
2996    False
2997     True
2998     True
2999     True
Name: predict, Length: 3000, dtype: bool
```

In [10]:

```
cluster1_data = data_random_select_3000[cluster2_index]
cluster1_data
```

Out[10]:

	carat	x	y	z	price
1	-0.101167	0.078977	0.126436	0.128862	2500.0
2	-0.904999	-1.009856	-0.955031	-1.002511	649.0
3	-0.524237	-0.411890	-0.449183	-0.422683	1415.0
4	-1.053073	-1.268677	-1.225397	-1.242928	658.0
5	0.448823	0.748342	0.771827	0.482416	4166.0
...	...	...	...	...	...
2994	-0.566544	-0.420815	-0.440462	-0.479251	1727.0
2995	-0.587697	-0.447590	-0.475348	-0.578246	1806.0
2997	-0.206935	0.034353	0.047942	-0.168124	2453.0
2998	-0.820385	-0.902758	-0.841651	-0.762095	738.0
2999	-0.185781	-0.037046	-0.004387	0.029867	2326.0

2436 rows × 5 columns

In [11]:

```
cluster2_data = data_random_select_3000[cluster2_index]
cluster2_data
```

Out[11]:

	carat	x	y	z	price
1	-0.101167	0.078977	0.126436	0.128862	2500.0
2	-0.904999	-1.009856	-0.955031	-1.002511	649.0
3	-0.524237	-0.411890	-0.449183	-0.422683	1415.0
4	-1.053073	-1.268677	-1.225397	-1.242928	658.0
5	0.448823	0.748342	0.771827	0.482416	4166.0
...	...	...	...	...	...
2994	-0.566544	-0.420815	-0.440462	-0.479251	1727.0
2995	-0.587697	-0.447590	-0.475348	-0.578246	1806.0
2997	-0.206935	0.034353	0.047942	-0.168124	2453.0
2998	-0.820385	-0.902758	-0.841651	-0.762095	738.0
2999	-0.185781	-0.037046	-0.004387	0.029867	2326.0

2436 rows × 5 columns

In [12]:

```
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 [13]:

```
print(ship)
```

```
{'c1mean': carat      -0.349833
x          -0.323037
y          -0.317133
z          -0.313661
price    2322.174877
dtype: float64, 'c2mean': carat      -0.349833
x          -0.323037
y          -0.317133
z          -0.313661
price    2322.174877
dtype: float64, 'c1_std': carat       0.639762
x           0.769977
y           0.744834
z           0.759905
price    1721.938067
dtype: float64, 'c2_std': carat       0.639762
x           0.769977
y           0.744834
z           0.759905
price    1721.938067
dtype: float64}
```

use anfis

In [14]:

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

test/train

In [15]:

```
path = "../daimonds_HW1_test.csv"
test_data = pd.read_csv(path).drop(["Unnamed: 0"],axis=1)
test = reorder(test_data)
test = shuffle(test)
test_select_1000=np.array(test.sample(1000))
X_test, y_test = test_select_1000[:, 0:4], test_select_1000[:, 4]

train = np.array(data_random_select_3000)
train = shuffle(train)
X, Y = train[:, 0:4], train[:, 4]
```

In [16]:

```
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 [17]:

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

```
current error: 7376993476.197264

C:\Users\40638\OneDrive - 國立陽明交通大學\桌面\machine learning\kmeans-with-a
nfis\anfis\anfis.py:104: RuntimeWarning: divide by zero encountered in doubl
e_scalars
  eta = k / np.abs(np.sum(t))

current error: 7373471993.118223
```

Out[17]:

```
array([[1936.82274612],
       [5425.77583784],
       [9524.23256913],
       ...,
       [ 966.5107976 ],
       [5769.77537967],
       [6066.64598929]])
```

In [18]:

```
def prediction(anfis, X):
    y_pred = np.squeeze(predict(anf, X), axis=1)
    return y_pred
```

In [19]:

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

In [24]:

```
from sklearn.metrics import mean_absolute_percentage_error
print("MAPE_train:",mean_absolute_percentage_error(Y, y_train_pred))
print("MAPE_test:",mean_absolute_percentage_error(y_test, y_test_pred))
```

```
MAPE_train: 0.318197946109387
MAPE_test: 0.3110758851443322
```

In [21]:

```
from sklearn.metrics import mean_squared_error
print("MSE_train:",mean_squared_error(Y, y_train_pred))
print("MSE:_test:",mean_squared_error(y_test, y_test_pred))
```

```
MSE_train: 2457815.5201271228
MSE:_test: 2326047.943400685
```

In [22]:

```
def rmse(y_true, y_pred):
    return np.sqrt(mean_squared_error(y_true, y_pred))
print("RMSE_train:",rmse(Y, y_train_pred))
print("RMSE_test:",rmse(y_test, y_test_pred))
```

```
RMSE_train: 1567.7421727207325
RMSE_test: 1525.1386636633029
```

In [23]:

```
Y_std=Y.std()
y_test_std=y_test.std()
def nrmse(y_true, y_pred,std):
    return (mean_squared_error(y_true, y_pred))/std
print("NRMSE_train:",nrmse(Y, y_train_pred,Y_std))
print("NRMSE_test:",nrmse(y_test, y_test_pred,y_test_std))
```

```
NRMSE_train: 623.7395161331065
NRMSE_test: 591.295034313826
```