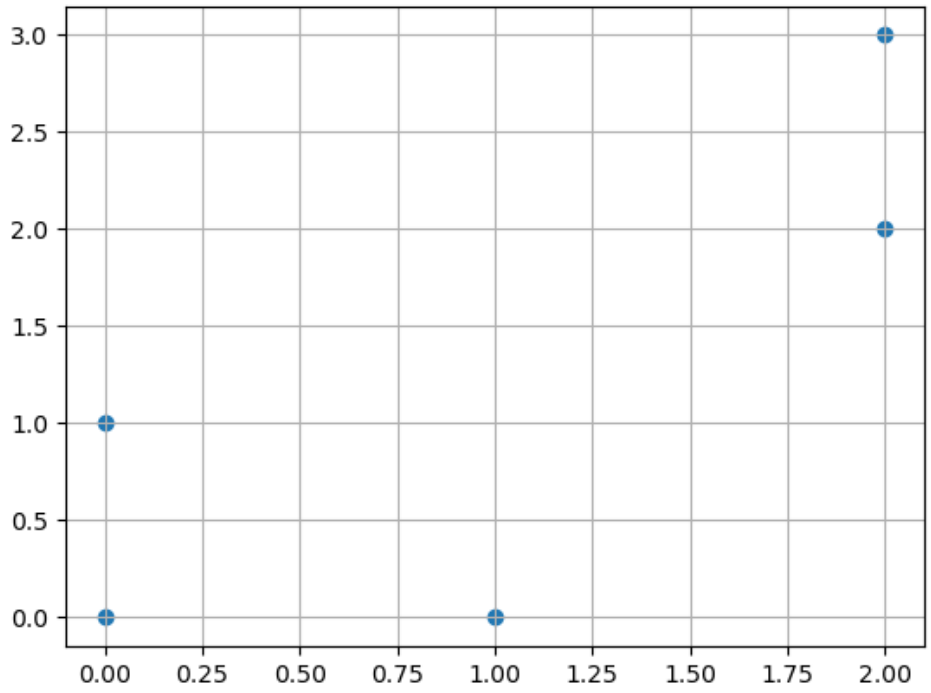
This homework is done by Tianwei Mo.

1.

a)



b) Assume the cluster with center (0, 0) is cluster 1, and the other is cluster 2.
Point 1, 3 is in cluster 1. The other point is in cluster 2.
The new center for cluster 1 is (0, 0.5). the new center for cluster 2 is (1.667, 1.667).

2. The function:

```
def outlier_detect(Xtr, Xts, nc, t):
    km = KMeans(n_clusters=nc)
    km.fit(Xtr)
    center = km.cluster_centers_
    center = center[:, :, None].T
    Xts = Xts.reshape(Xts.shape[0], Xts.shape[1], 1)
    dist = (Xts - center) ** 2
    dist = np.sum(dist, axis=1)
    outlier = dist < t
    outlier = np.sum(outlier, axis=1)
    out_index = np.where(outlier == 0)
    outlier = np.zeros(Xts.shape[0])
    outlier[out_index] = 1
    return outlier
```

3. The function:

```python
import numpy as np
from sklearn.cluster import KMeans
import random

def K_center(Xtr, K):
    center_index = random.sample(range(Xtr.shape[0]), K)
    return Xtr[center_index]

```

4. The preprocessing function:

```python
from sklearn.cluster import KMeans
import numpy as np
from sklearn.linear_model import LinearRegression

def preprocessing(Xtr, ytr, Xts, yts, nc):
    km = KMeans(n_clusters=nc)
    km.fit(Xtr)
    cluster = km.predict(Xtr)

    for c in range(nc):
        Xtr_c = Xtr[cluster == c]
        ytr_c = ytr[cluster == c]
        reg = LinearRegression()
        reg.fit(Xtr_c, ytr_c)
        yhat = reg.predict(Xts)
        mse = np.mean((yts - yhat) ** 2)
```