Unsupervised Learning

Clustering-K Means

dataroadmap

مدرس: مونا حاتمی

Unsupervised Learning

یادگیری بدون نظارت

در یادگیری بدون نظارت نیاز به آموزش دادن مدل وجود ندارد و ما اجازه میدیم مدل به خودی خود رابطه بین دیتاهای موجود رو کشف کند.

پس درالگوریتم های یادگیری نیازی به دیتاست ترین (train dataset) نداریم یا به عبارتی دیگر نیازی به لیبل کردن دیتا (y), نداریم.

dataroadmap

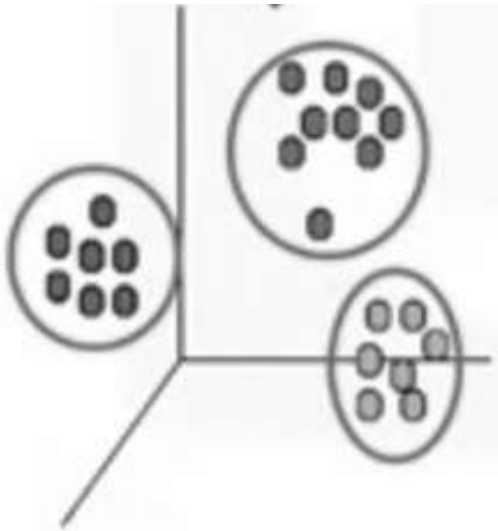
Customer Segmentation

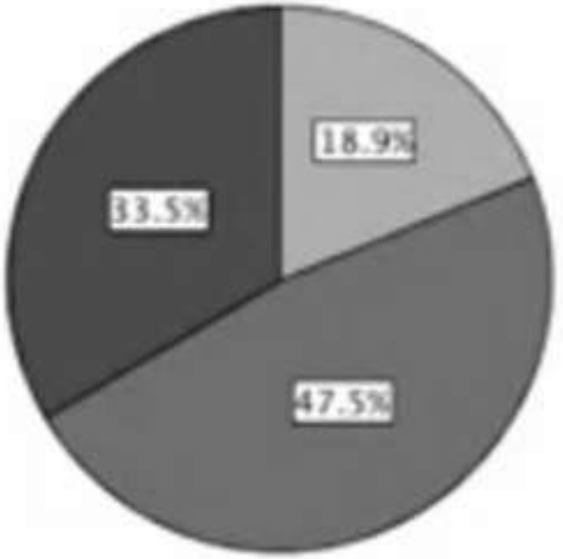
جداسازی مشتری

Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Address	DebtIncomeRatio	Defaulted
1	41	2	6	19	0.124	1.073	NBA001	6.3	0
2	47	1	26	100	4.582	8.218	NBA021	12.8	0
3	33	2	10	57	6.111	5.802	NBA013	20.9	1
4	29	2	4	19	0.681	0.516	NBA009	6.3	0
5	47	1	31	253	9.308	8.908	NBA008	7.2	0
6	40	1	23	81	0.998	7.831	NBA016	10.9	1
7	38	2	4	56	0.442	0.454	NBA013	1.6	0
8	42	3	0	64	0.279	3.945	NBA009	6.6	0
9	26	1	5	18	0.575	2.215	NBA006	15.5	1

Clustering

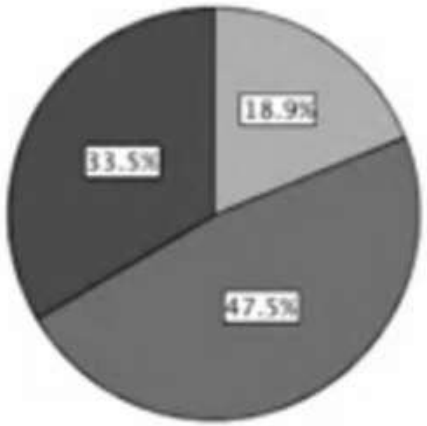
خوشه بندی





Clustering

خوشه بندی



Cluster
cluster-1
cluster-2
cluster-3

Cluster	Segment Name
cluster-1	AFFULUENT AND MIDDLE AGED
cluster-2	YOUNG EDUCATED AND MIDDLE INCOME
cluster-3	YOUNG AND LOW INCOME

Clustering

خوشه بندی

Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Address	DebtIncomeRatio	Defaulted
1	41	2	6	19	0.124	1.073	NBA001	6.3	0
2	47	1	26	100	4.582	8.218	NBA021	12.8	0
3	33	2	10	57	6.111	5.802	NBA013	20.9	1
4	29	2	4	19	0.681	0.516	NBA009	6.3	0
5	47	1	31	253	9.308	8.908	NBA008	7.2	0
6	40	1	23	81	0.998	7.831	NBA016	10.9	1
7	38	2	4	56	0.442	0.454	NBA013	1.6	0
8	42	3	0	64	0.279	3.945	NBA009	6.6	0
9	26	1	5	18	0.575	2.215	NBA006	15.5	1

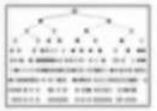
Customer ID	Segment
1	YOUNG AND LOW INCOME
2	AFFULUENT AND MIDDLE AGED
3	AFFULUENT AND MIDDLE AGED
4	YOUNG AND LOW INCOME
5	AFFULUENT AND MIDDLE AGED
6	AFFULUENT AND MIDDLE AGED
7	YOUNG AND LOW INCOME
8	YOUNG AND LOW INCOME
9	AFFULUENT AND MIDDLE AGED

Clustering vs Classification

Labeled dataset

Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Address	DebtIncomeRatio	Defaulted
1	41	2	6	19	0.124	1.073	NBA001	6.3	0
2	47	1	26	100	4.582	8.218	NBA021	12.8	0
3	33	2	10	57	6.111	5.802	NBA013	20.9	1
4	29	2	4	19	0.681	0.516	NBA009	6.3	0
5	47	1	31	253	9.308	8.908	NBA008	7.2	0
6	40	1	23	81	0.998	7.831	NBA016	10.9	1
7	38	2	4	56	0.442	0.454	NBA013	1.6	0
8	42	3	0	64	0.279	3.945	NBA009	6.6	0
9	26	1	5	18	0.575	2.215	NBA006	15.5	1
10	47	3	23	115	0.653	3.947	NBA011	4	0

Modeling



Decision Tree

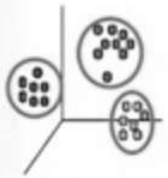
Prediction

Unlabeled dataset

Customer Id	Age	Edu	Years Employed	Income	Card Debt	Other Debt	Address	DebtIncomeRatio	Defaulted
1	41	2	6	19	0.124	1.073	NBA001	6.3	0
2	47	1	26	100	4.582	8.218	NBA021	12.8	0
3	33	2	10	57	6.111	5.802	NBA013	20.9	1
4	29	2	4	19	0.681	0.516	NBA009	6.3	0
5	47	1	31	253	9.308	8.908	NBA008	7.2	0
6	40	1	23	81	0.998	7.831	NBA016	10.9	1
7	38	2	4	56	0.442	0.454	NBA013	1.6	0
8	42	3	0	64	0.279	3.945	NBA009	6.6	0
9	26	1	5	18	0.575	2.215	NBA006	15.5	1

Modeling

Segmentation



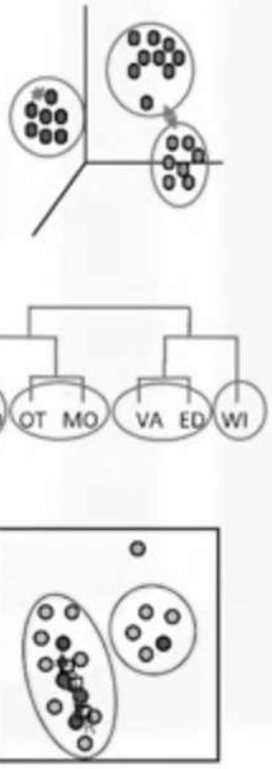
K-Means

Clustering Applications

- RETAIL/MARKETING:
  - Identifying buying patterns of customers
  - Recommending new books or movies to new customers
- BANKING:
  - Fraud detection in credit card use
  - Identifying clusters of customers (e.g., loyal)
- INSURANCE:
  - Fraud detection in claims analysis
  - Insurance risk of customers

Clustering Algorithms

- Partitioned-based Clustering
  - Relatively efficient
  - E.g. k-Means, k-Median, Fuzzy c-Means
- Hierarchical Clustering
  - Produces trees of clusters
  - E.g. Agglomerative, Divisive
- Density-based Clustering
  - Produces arbitrary shaped clusters
  - E.g. DBSCAN



Distance Formula

محاسبه فاصله دو نقط

1-dimensional similarity/distance

Customer 1
Age
54

Customer 2
Age
50

$$\text{Dis}\ (x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

$$\text{Dis}\ (x_1, x_2) = \sqrt{(54 - 50)^2} = 4$$

Distance Formula

Multi-dimentional similarity/distance

Customer 1		
Age	Income	education
54	190	3

Customer 2		
Age	Income	education
50	200	8

$$\text{Dis}\,(x_1, x_2) = \sqrt{\sum_{i=0}^{n}(x_{1i} - x_{2i})^2}$$

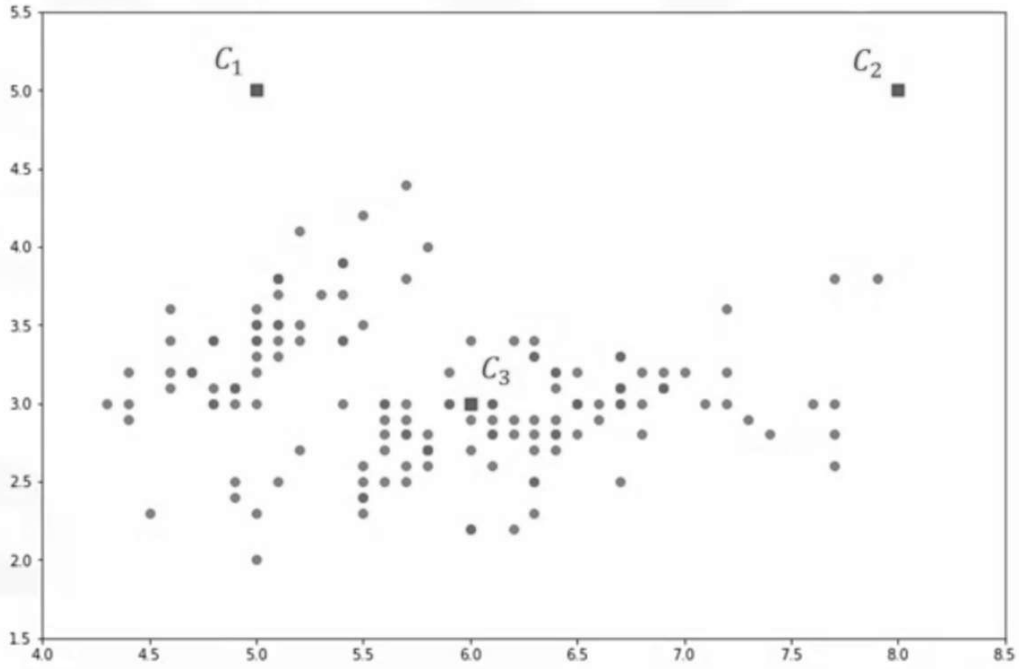
$$= \sqrt{(54-50)^2+(190-200)^2+(3-8)^2} = 11.87$$

k_Mean

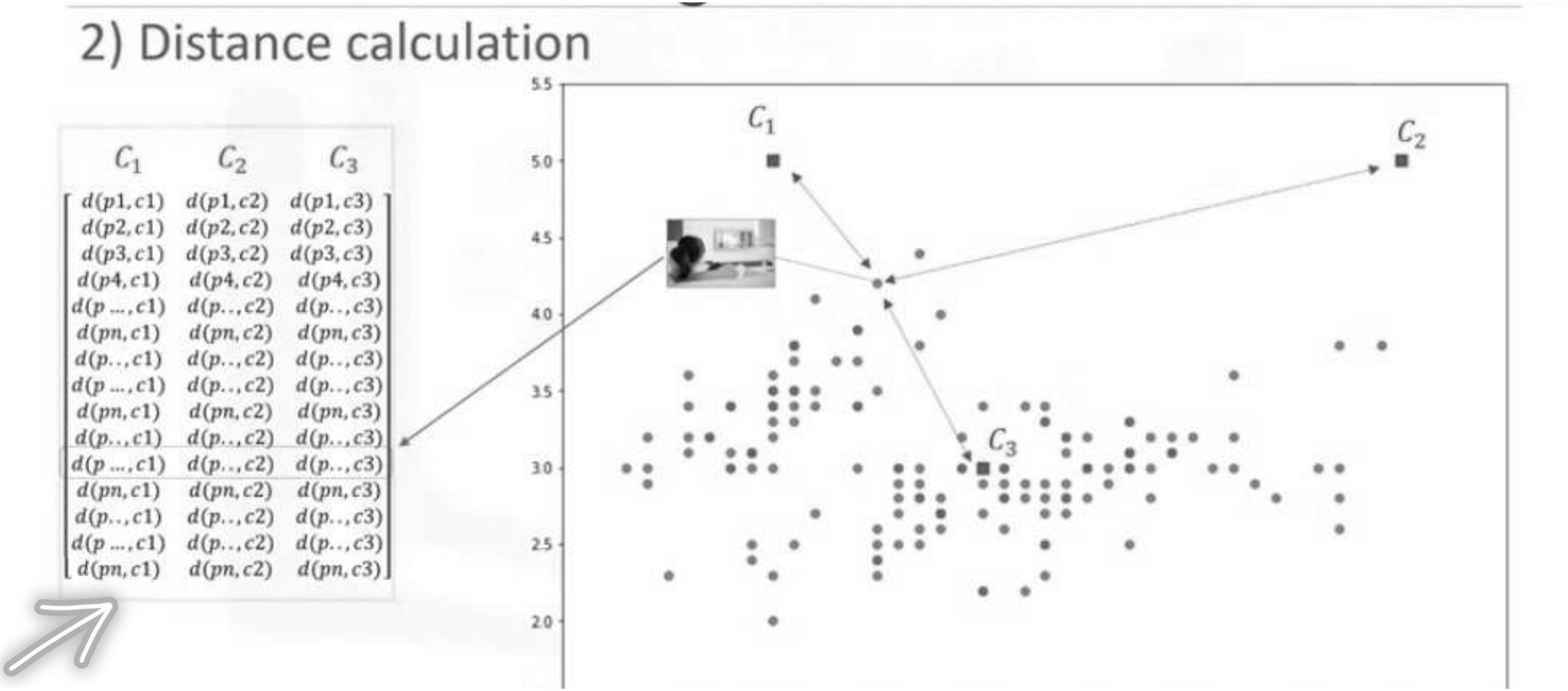
k-Means clustering – initialize k

1) Initialize k=3 centroids randomly

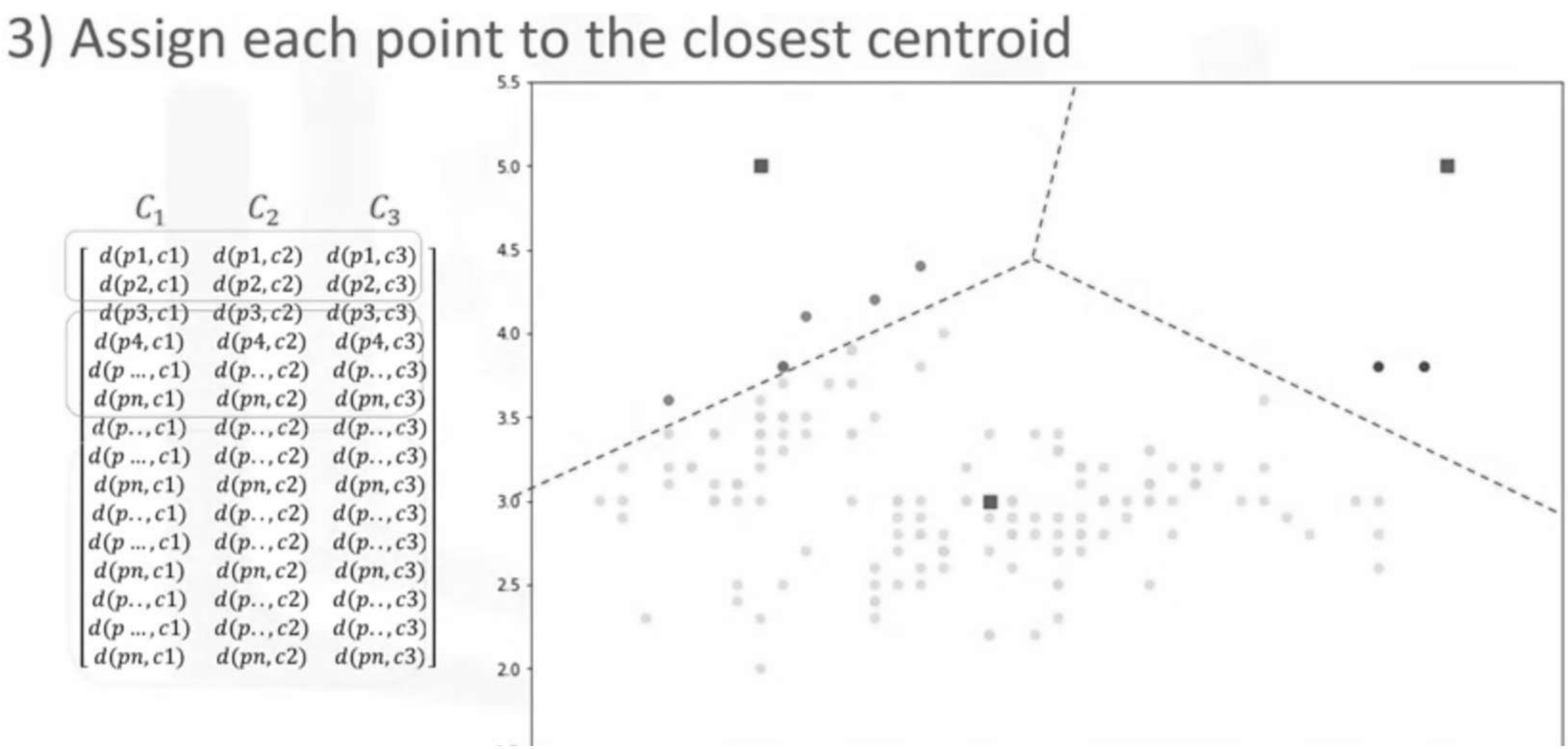
C_1 = **[8., 5.]**
C_2 = **[5., 5.]**
C_3 = **[6., 3.]**



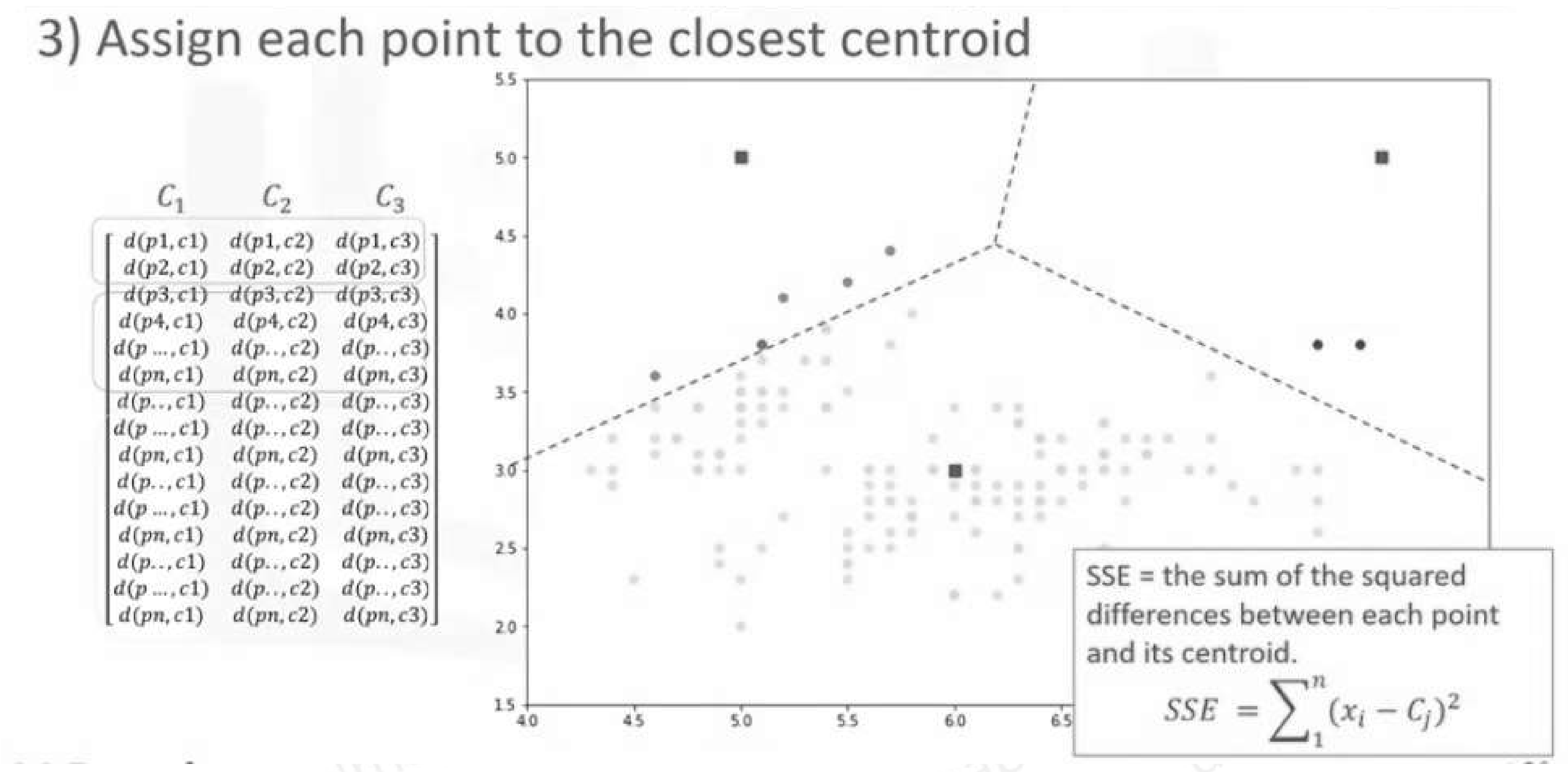
K- Means clustering- calculate the distance



K- Means clustering- assign to centroid

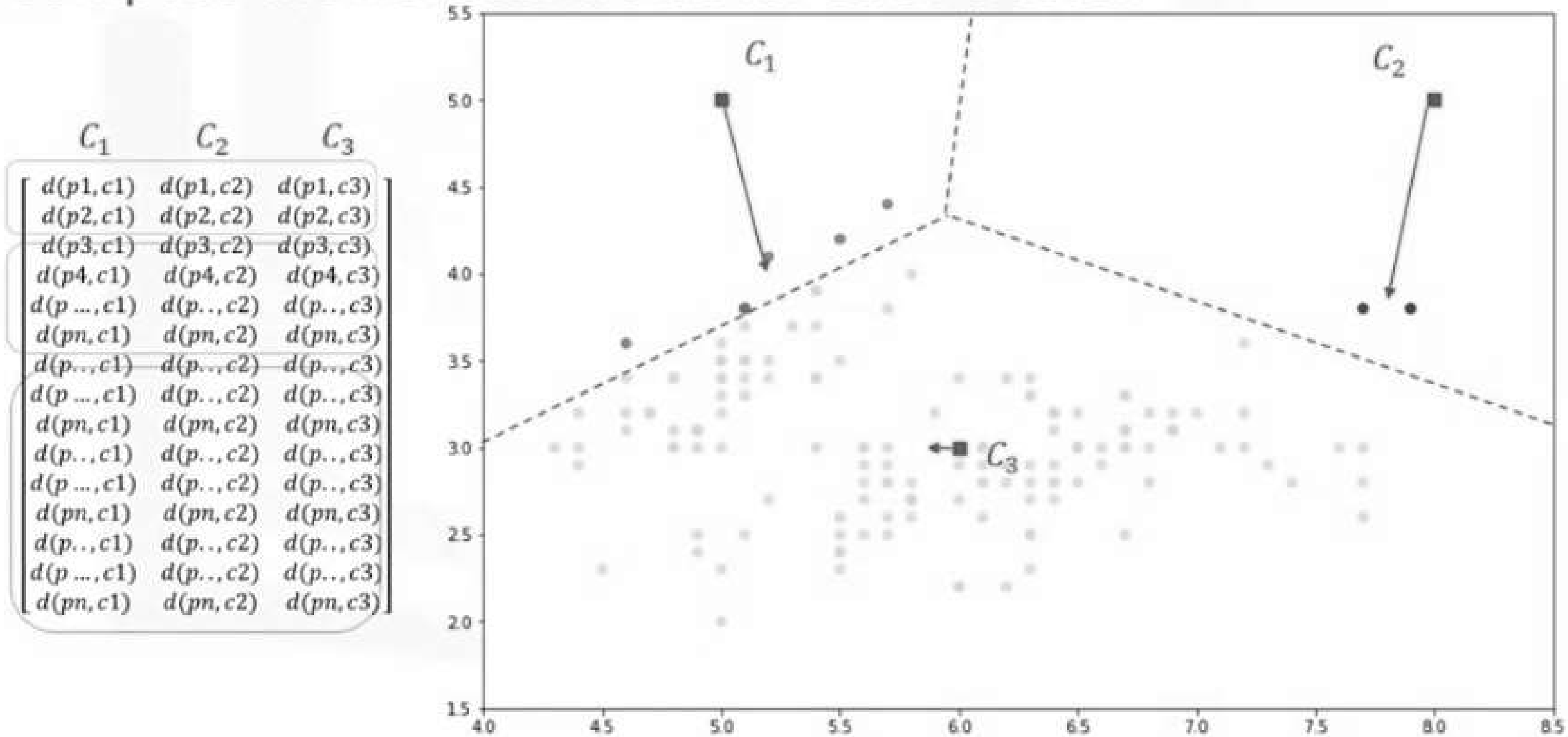


K- Means clustering- assign to centroid

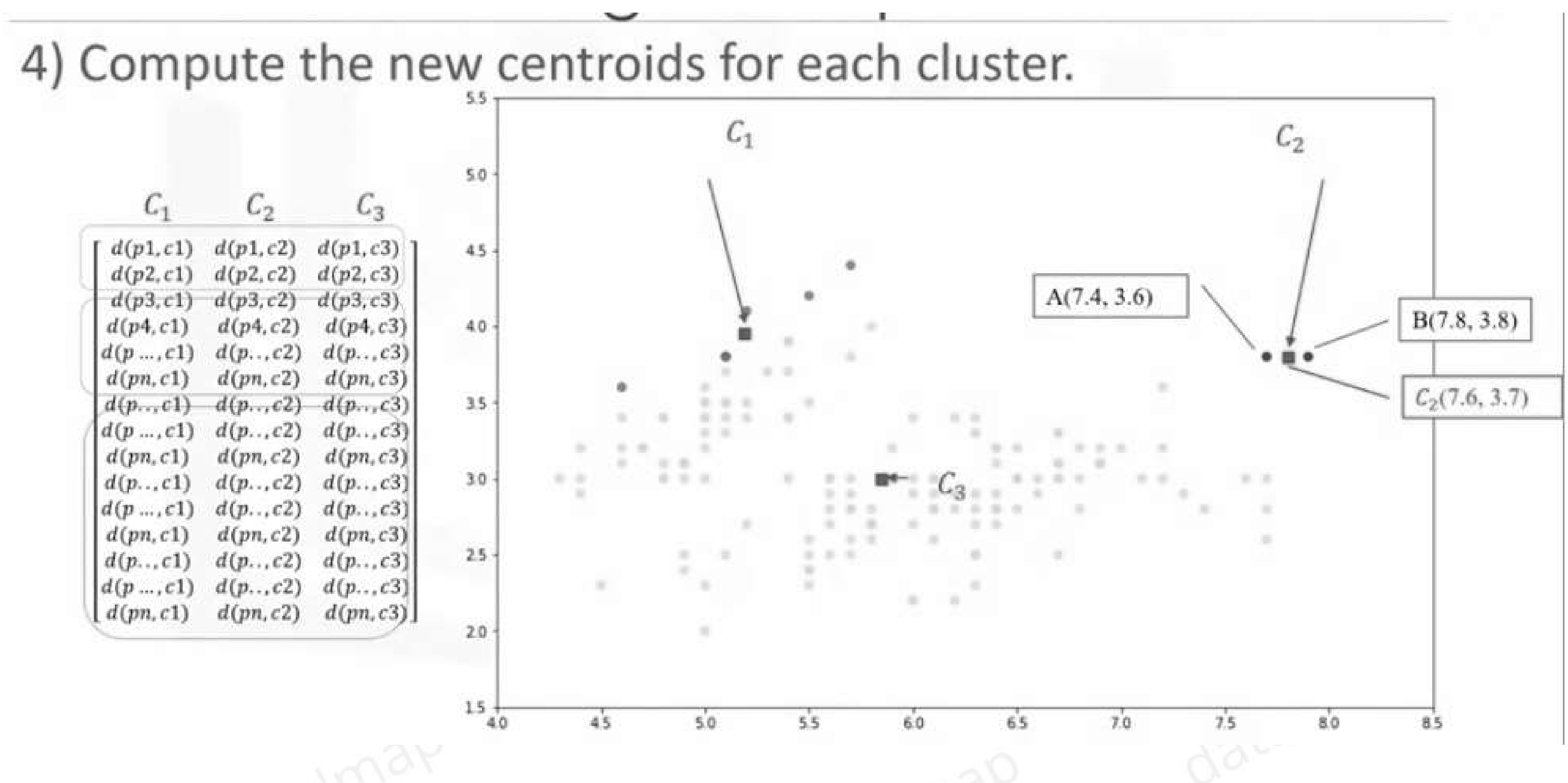


K- Means clustering- new centroids

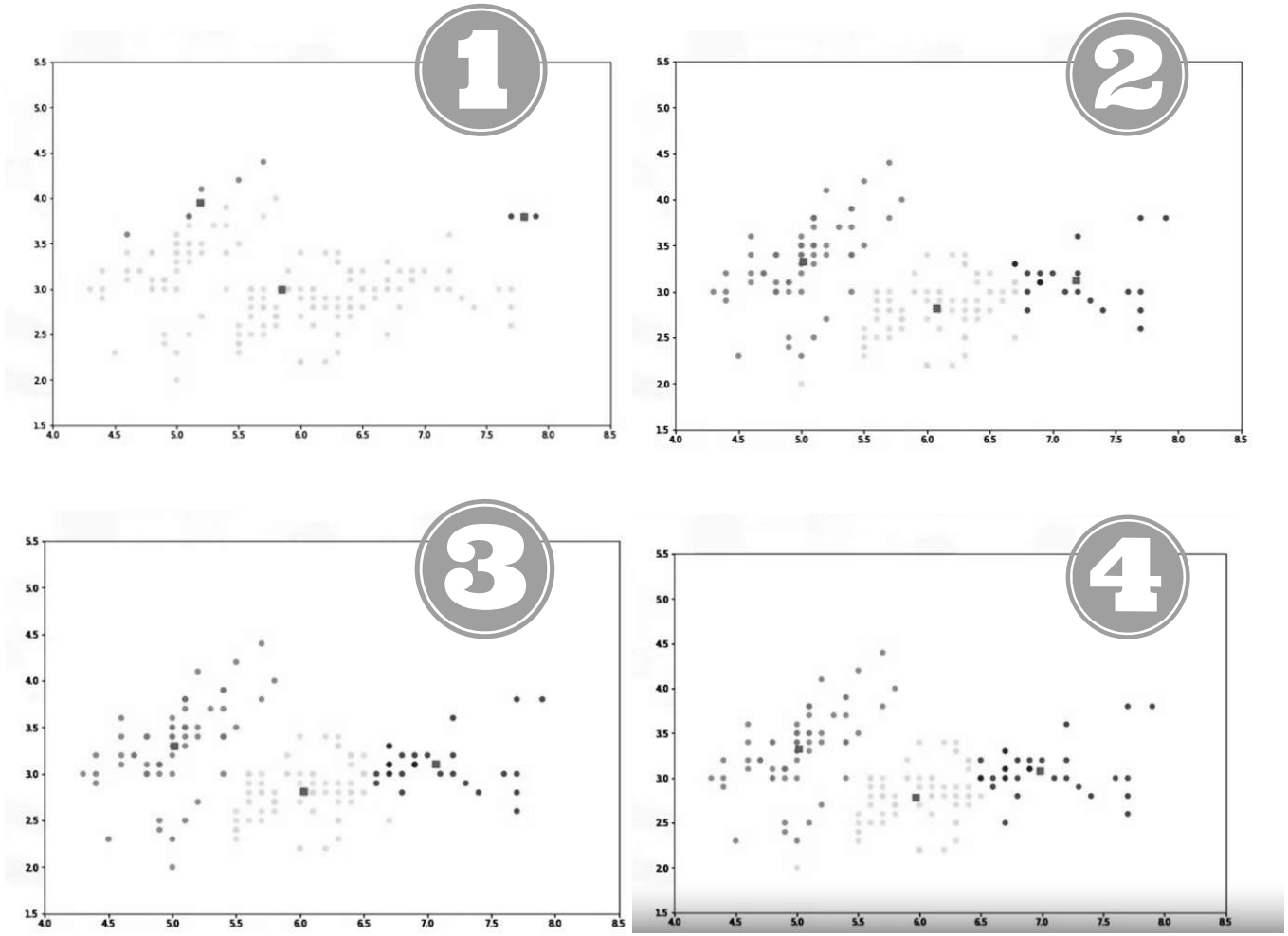
4) Compute the new centroids for each cluster.



K- Means clustering- new centroids



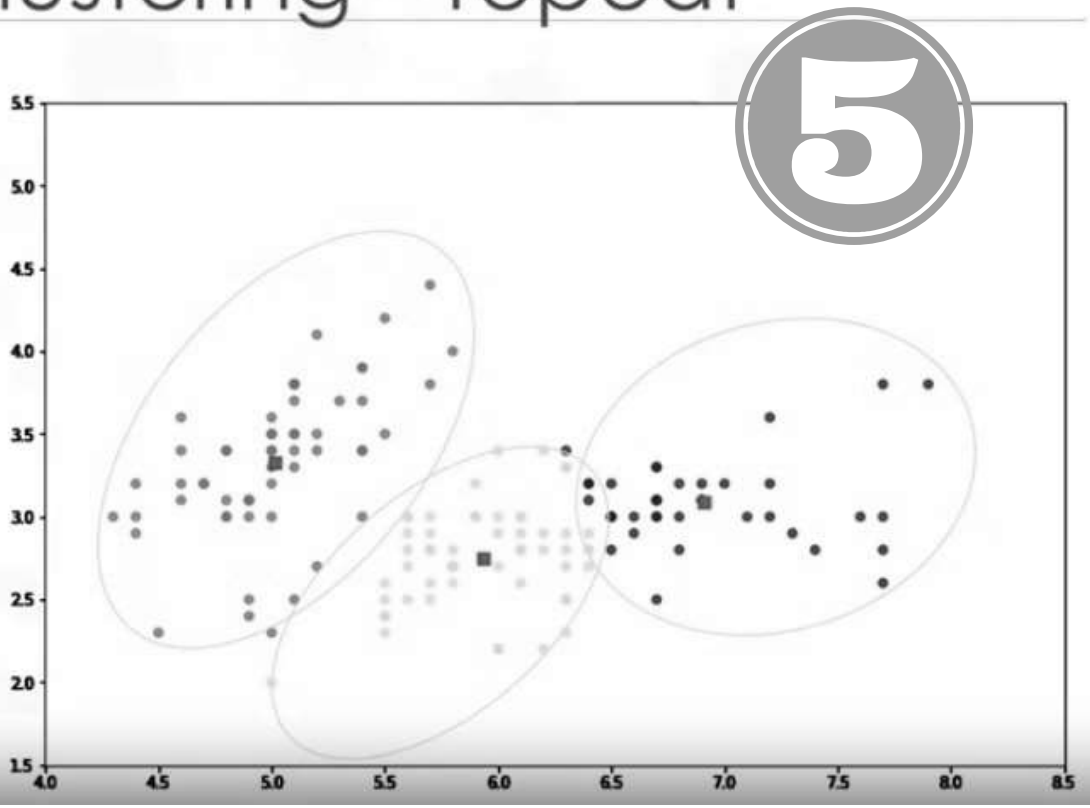
k_Mean



k_Mean

k-Means clustering – repeat

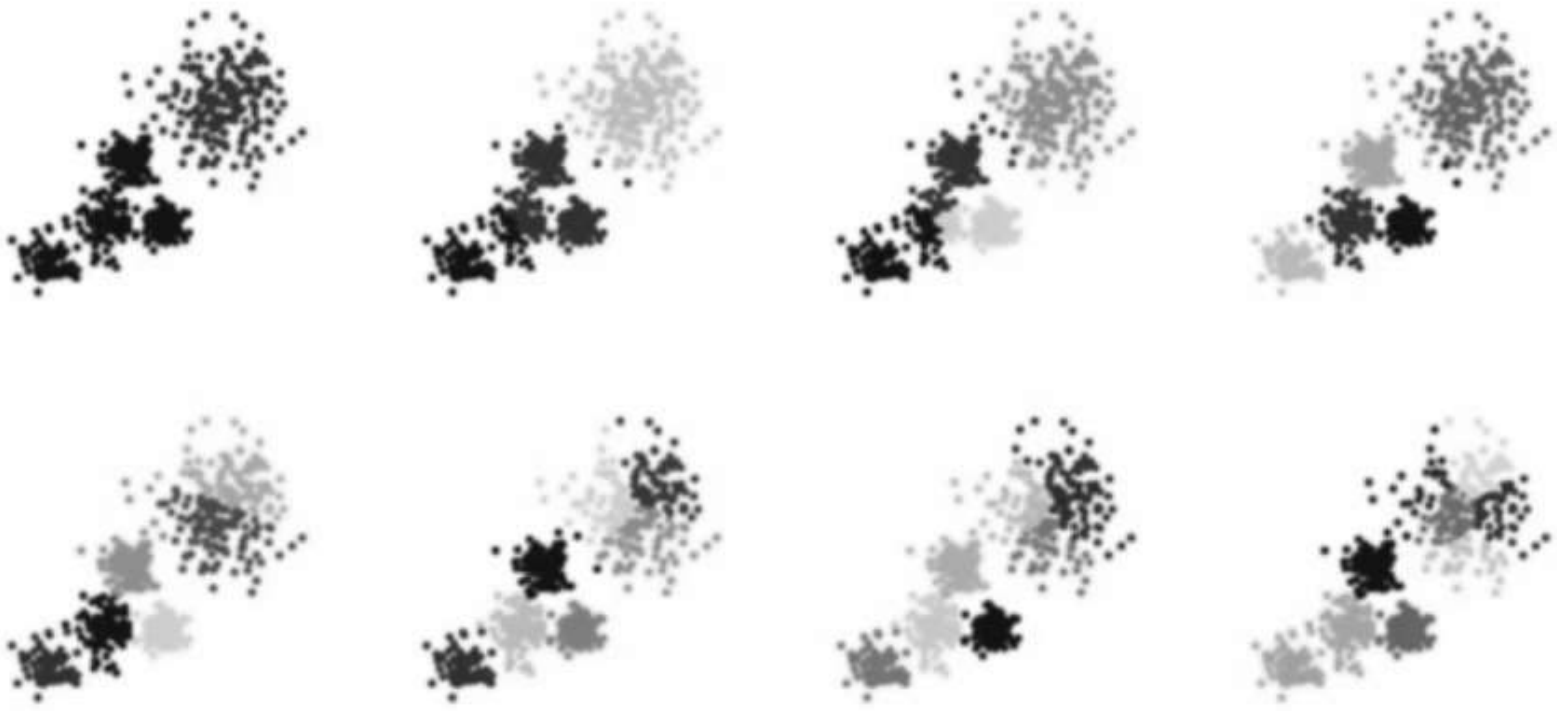
5) Repeat until there are no more changes.



SSE = the sum of the squared differences between each point and its centroid.

$$SSE = \sum_{1}^{n} (x_i - C_j)^2$$

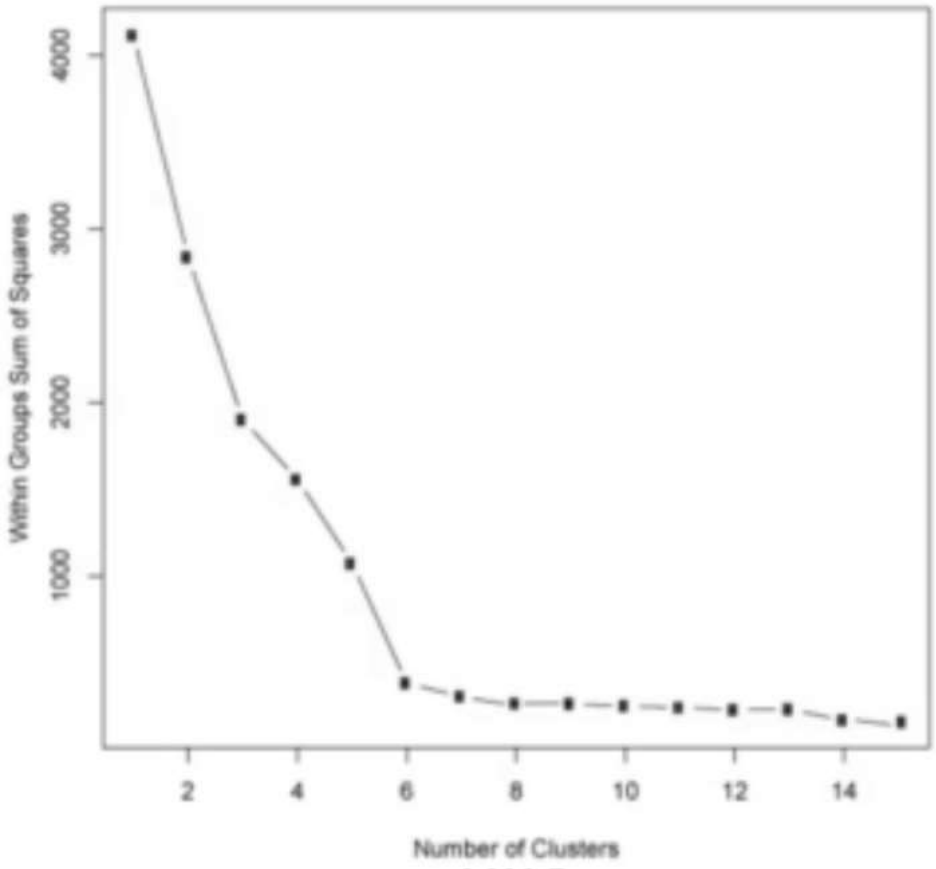
Choosing a K Value



Elbow Method

SSE = the sum of the squared differences between each point and its centroid.

$$SSE = \sum_{1}^{n} (x_i - C_j)^2$$



Input Libraries

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

```
df=pd.read_csv('data_for_kmean.csv')
```

dataroadmap

Open dataframe

```
: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Feature 1  200 non-null    float64
 1   Feature 2  200 non-null    float64
dtypes: float64(2)
memory usage: 3.2 KB
```

```
: df
```

t[4]:

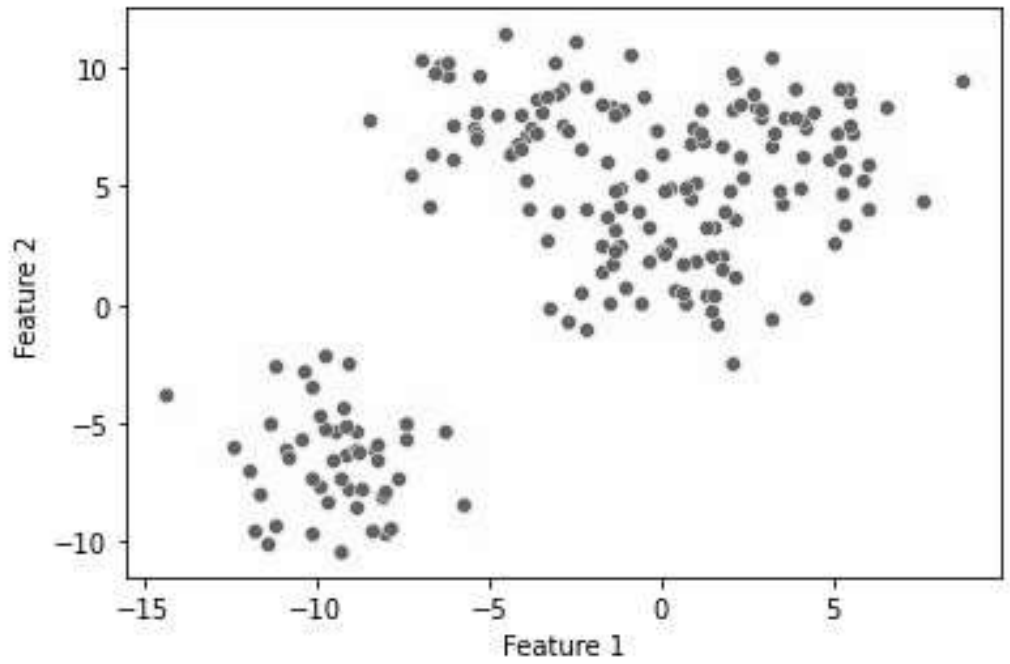
	Feature 1	Feature 2
0	-6.428841	10.141117
1	5.868679	5.201104
2	-0.376109	3.264279
3	2.166792	9.563005

dataroadmap

Scatter Plot

```
sns.scatterplot(x=df["Feature 1"], y=df["Feature 2"])
```

```
5]: <AxesSubplot:xlabel='Feature 1', ylabel='Feature 2'>
```



dataroadmap
dataroadmap
dataroadmap

Unsupervised Learning

```
from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=4)
```

```
kmeans.fit(df)
```

]:

KMeans

KMeans(n_clusters=4)

dataroadmap

Unsupervised Learning

```
kmeans.labels_
```

```
array([2, 0, 3, 0, 0, 1, 0, 3, 0, 3, 2, 3, 0, 0, 2, 3, 0, 3, 1, 2, 1, 3,
       3, 1, 2, 1, 1, 3, 0, 0, 2, 1, 0, 3, 3, 2, 1, 1, 1, 3, 1, 2, 2, 2,
       3, 0, 2, 3, 1, 3, 3, 2, 0, 3, 1, 2, 3, 3, 2, 0, 1, 0, 1, 2, 0, 3,
       1, 0, 0, 1, 0, 3, 1, 3, 1, 0, 0, 3, 2, 3, 3, 1, 0, 1, 3, 3, 3, 2,
       3, 1, 1, 1, 1, 3, 3, 1, 0, 2, 1, 0, 3, 1, 3, 3, 0, 3, 1, 0, 1, 1,
       0, 2, 2, 0, 1, 0, 2, 2, 0, 2, 3, 2, 3, 2, 3, 0, 2, 3, 1, 2, 2, 2,
       3, 1, 1, 2, 0, 2, 0, 3, 1, 0, 1, 2, 2, 0, 3, 1, 2, 2, 2, 2, 3, 0,
       3, 2, 0, 0, 0, 3, 0, 3, 3, 2, 1, 2, 3, 0, 2, 3, 0, 3, 2, 0, 3, 2,
       0, 0, 1, 0, 2, 1, 1, 2, 1, 1, 1, 1, 1, 3, 1, 0, 0, 2, 1, 3, 0, 0,
       1, 3])
```

```
kmeans.cluster_centers_
```

```
array([[ 3.71749226,  7.01388735],
       [-9.46941837, -6.56081545],
       [-4.13591321,  7.95389851],
       [-0.0123077 ,  2.13407664]])
```

dataroadmap

Unsupervised Learning

```
df_k=df
```

```
df_k['Lable(K=4)']=kmeans.labels_
```

```
df_k
```

:

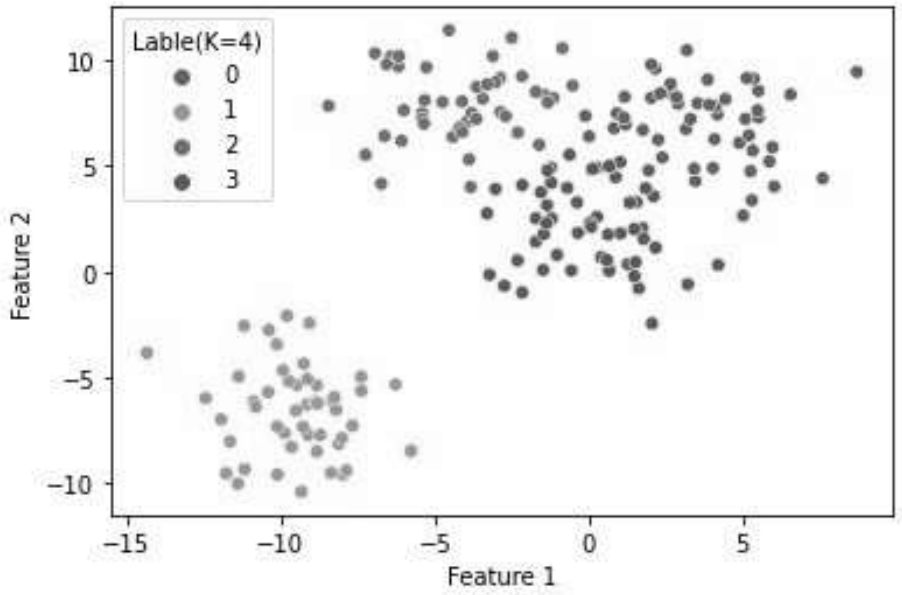
	Feature 1	Feature 2	Lable(K=4)
0	-6.428841	10.141117	2
1	5.868679	5.201104	0
2	-0.376109	3.264279	3
3	2.166792	9.563005	0
4	5.095086	7.207527	0

dataroadmap

Scatter Plot

```
sns.scatterplot(data=df_k, x='Feature 1',y='Feature 2',hue='Lable(K=4)',palette='tab10')
```

```
]: <AxesSubplot:xlabel='Feature 1', ylabel='Feature 2'>
```



KMean for k=2

```
kmeans1 = KMeans(n_clusters=2)
```

```
kmeans1.fit(df)
```

[17]:

```
KMeans(n_clusters=2)
```

```
df_k['Lable(K=2)']=kmeans1.labels_
```

```
df_k
```

[19]:

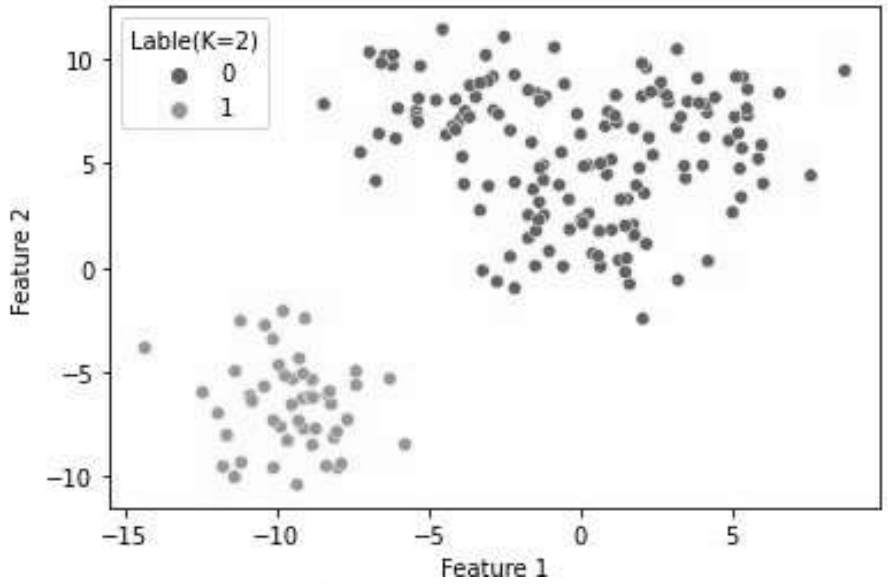
	Feature 1	Feature 2	Lable(K=4)	Lable(K=2)
0	-6.428841	10.141117	2	0
1	5.868679	5.201104	3	0

dataroadmap

Scatter Plot

```
sns.scatterplot(data=df_k, x='Feature 1',y='Feature 2',hue='Lable(K=2)',palette='tab10')
```

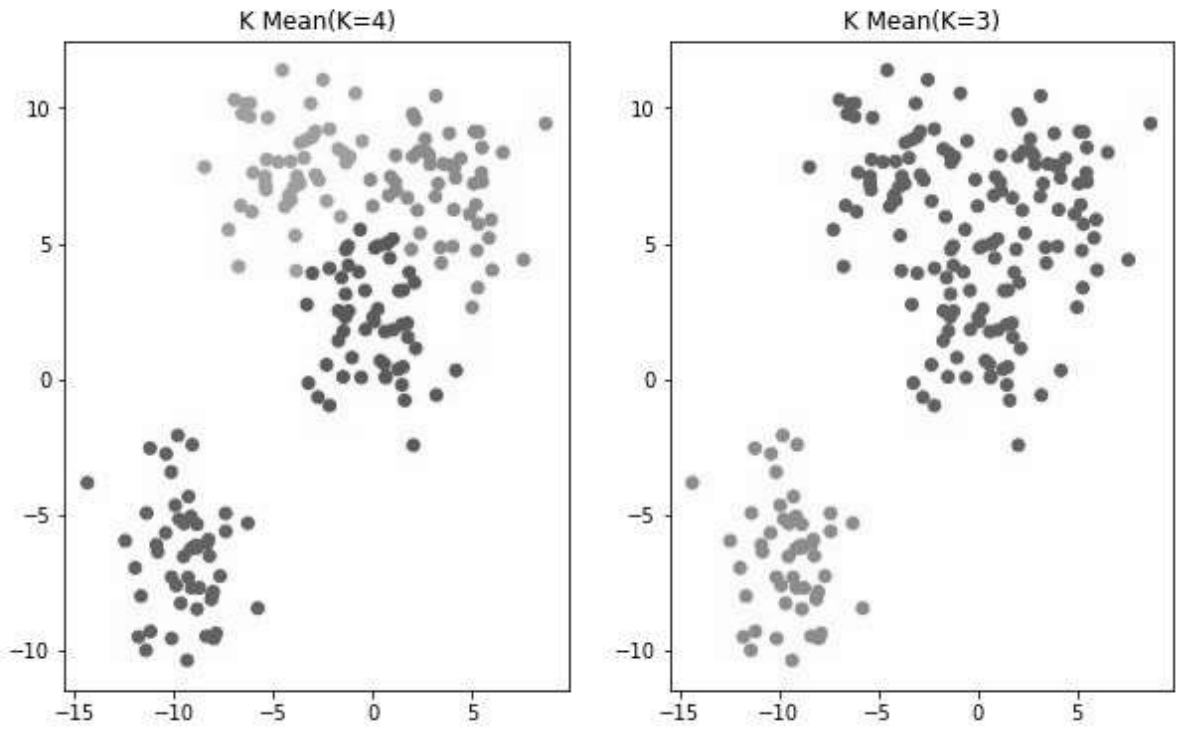
```
20]: <AxesSubplot:xlabel='Feature 1', ylabel='Feature 2'>
```



Scatter Plots

```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(10,6))
ax1.set_title('K Mean(K=4)')
ax1.scatter(x=df_k['Feature 1'],y=df_k['Feature 2'],c=df_k['Lable(K=4)'], cmap='tab10')
ax2.set_title('K Mean(K=3)')
ax2.scatter(x=df_k['Feature 1'],y=df_k['Feature 2'],c=df_k['Lable(K=2)'],cmap='tab10')
```

```
<matplotlib.collections.PathCollection at 0x1cc75a17a90>
```



Create dataset

```
from sklearn.datasets import make_blobs
```

```
# Create Data
data = make_blobs(n_samples=200, n_features=2,
                  centers=4, cluster_std=1.8,random_state=101)
```

```
data
```

```
25]: (array([[-6.42884095e+00,  1.01411174e+01],
             [ 5.86867888e+00,  5.20110356e+00],
             [-3.76109375e-01,  3.26427943e+00],
             [ 2.16679181e+00,  9.56300522e+00],
             [ 5.09508570e+00,  7.20752718e+00],
             [-1.08788882e+01, -6.11318040e+00],
```

Create dataset

```
type(data)
```

```
4]: tuple
```

```
len(data)
```

```
7]: 2
```

dataroadmap

Create dataset

```
data[0]
```

```
array([[-6.42884095e+00,  1.01411174e+01],
       [ 5.86867888e+00,  5.20110356e+00],
       [-3.76109375e-01,  3.26427943e+00],
```

```
type(data[0])
```

```
numpy.ndarray
```

```
len(data[0])
```

```
200
```

dataroadmap

Create dataset

```
]:  data[1]
```

```
[36]: array([3, 2, 0, 2, 2, 1, 2, 0, 2, 0, 3, 0, 2, 2, 3, 0, 2, 0, 1, 3, 1, 0,
       0, 1, 3, 1, 1, 0, 2, 2, 3, 1, 2, 0, 0, 3, 1, 1, 1, 2, 1, 3, 3, 3,
       0, 3, 3, 0, 1, 2, 0, 3, 2, 0, 1, 3, 0, 0, 3, 2, 1, 2, 1, 3, 2, 0,
       1, 2, 2, 1, 2, 0, 1, 3, 1, 2, 2, 0, 3, 0, 0, 1, 2, 1, 0, 0, 0, 3,
       2, 1, 1, 1, 1, 3, 0, 1, 2, 3, 1, 2, 0, 1, 0, 0, 2, 0, 1, 2, 1, 1,
       0, 3, 3, 2, 1, 2, 3, 3, 2, 3, 0, 3, 0, 3, 0, 2, 3, 0, 1, 3, 3, 3,
       0, 1, 1, 3, 2, 3, 2, 0, 1, 2, 1, 3, 3, 2, 0, 1, 3, 3, 3, 3, 0, 2,
       0, 3, 2, 2, 2, 0, 2, 0, 0, 3, 1, 3, 0, 2, 3, 0, 2, 0, 3, 3, 0, 3,
       2, 2, 1, 2, 3, 1, 1, 3, 1, 1, 1, 1, 1, 0, 1, 2, 2, 3, 1, 0, 2, 2,
       1, 0])
```

```
]:  data[0][1]
```

```
[37]: array([5.86867888, 5.20110356])
```

Create dataset

```
3]: data[0][:,0]

ut[33]: array([-6.42884095e+00,  5.86867888e+00, -3.76109375e-01,  2.16679181e+00,
                5.09508570e+00, -1.08788882e+01,  2.03405554e+00, -1.71798771e+00,
                1.16911341e+00, -1.35185444e+00, -6.18548214e+00, -1.19856602e+00,
                2.90296863e+00,  2.39250023e+00, -5.27545147e+00, -5.66814687e-01,
                5.97336628e+00, -2.31355268e+00, -1.01344756e+01, -4.54082629e+00,
               -1.04155833e+01,  6.64796693e-01,  2.11460477e+00, -1.11790221e+01,
               -6.63698251e+00, -7.67422005e+00, -7.98668260e+00,  1.27983684e+00,
                3.54480244e+00,  4.03940181e+00, -2.88118898e+00, -9.11009911e+00,
                5.26001172e+00,  2.05859724e+00, -1.71289834e+00, -5.40562319e+00,
               -1.11995123e+01, -1.13753641e+01, -1.17821836e+01,  1.74815503e+00,
               -9.00392334e+00, -2.86564584e+00, -1.42742293e+00, -3.10933432e+00,
                2.71130095e-01,  8.21556561e-01, -4.11495481e+00,  1.55414928e+00,
               -1.16546211e+01, -1.22009637e+00,  3.22017630e+00, -5.40452892e+00,
                6.02795351e+00,  4.02600451e-01, -7.38985009e+00, -1.60537707e+00,
                8.72770362e-01,  1.03445241e+00, -3.88943018e+00,  3.16835529e+00,
               -8.07309689e+00,  9.16131646e-01, -7.39648298e+00, -1.71632701e+00,
                2.71396283e+00, -2.16570885e+00, -1.19474369e+01,  4.89539219e+00,
                2.86177832e+00, -9.15392597e+00,  2.03477094e+00,  7.56601080e-01,
               -8.84039494e+00, -3.02650610e+00, -8.88037875e+00,  5.20737777e+00,

4]: data[0][:,1]

ut[34]: array([ 10.14111739,   5.20110356,   3.26427943,   9.56300522,
                 7.20752718,  -6.1131804 ,   9.76664755,   1.4140114 ,
                 8.24556988,   3.13245345,   9.67406555,   2.50408937,
                 7.91251003,   5.38173971,   9.63836659,   0.05602628,
                 5.87172022,   0.52398009,  -3.43130837,  11.39201739,
                -5.67545836,   0.09423047,   3.55938488,  -9.30976605,
                 6.39426436,  -7.26839654,  -9.57113308,   0.35315078,
                 7.93535678,   4.88870433,   9.12919391,  -7.6978166 ,
                 4.74007434,  -2.44083039,   2.51221197,   7.47228315,
                -2.55276744,  -4.94525091,  -9.50883007,   2.05595679,
```

Create dataset

```
df_dict={'Feature 1':data[0][:,0],'Feature 2':data[0][:,1]}
```

```
final_df=pd.DataFrame.from_dict(df_dict)
```

```
final_df.to_csv('data_for_kmean.csv', index=False)
```

dataroadmap

Elbow Method

```
k=1
```

```
kmeans = KMeans(n_clusters=k)
```

```
kmeans.fit(df)
```

```
KMeans(n_clusters=1)
```

```
kmeans.inertia_
```

```
14010.228683268439
```

SSE = the sum of the squared differences between each point and its centroid.

$$SSE = \sum_{1}^{n} (x_i - C_j)^2$$

dataroadmap

for loop

```
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(df)
    sse[k] = kmeans.inertia_ # Inertia: Sum of distanc
```

dataroadmap

sse

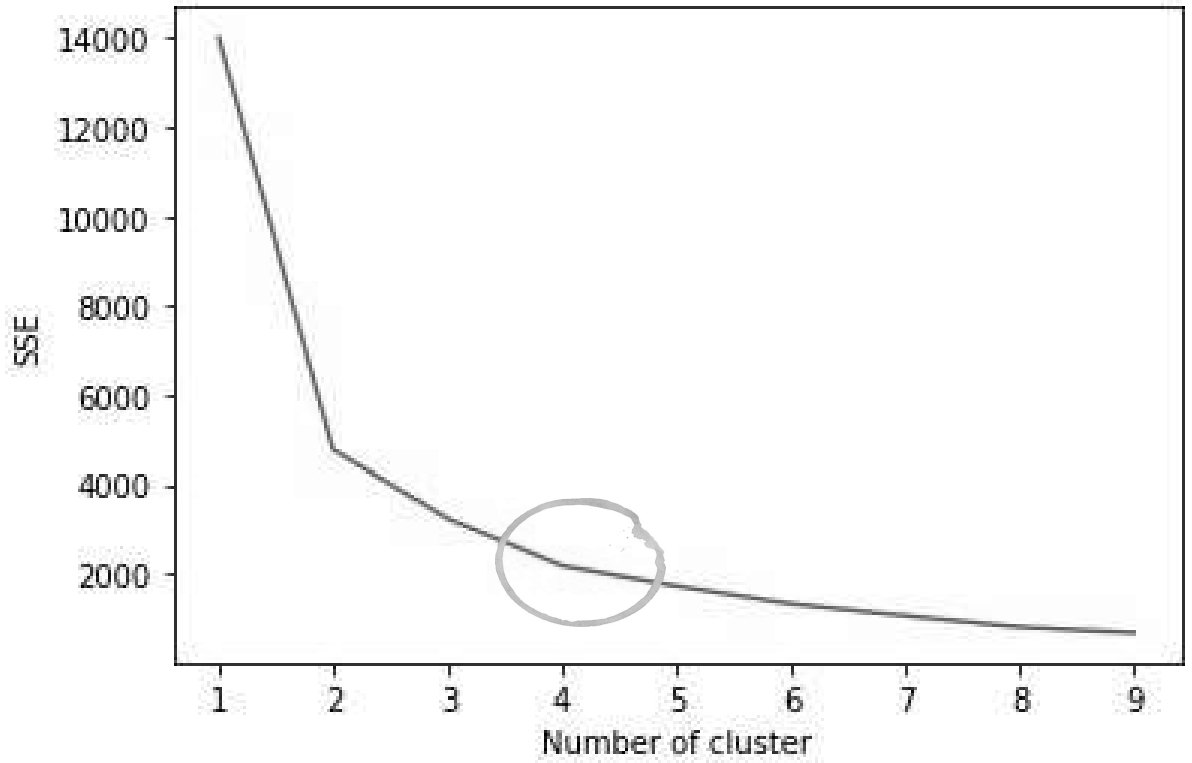
```
In [78]: sse
Out[78]: {1: 14010.228683268439,
          2: 4788.049576608008,
          3: 3241.7230990171934,
          4: 2186.5336916907268,
          5: 1726.3707402813363,
          6: 1328.7298595220077,
          7: 1059.0967616238315,
          8: 805.4078760179675,
          9: 677.7041091263894}
```

dataroadmap

Plot

```
plt.plot(list(sse.keys()), list(sse.values()))
plt.xlabel("Number of cluster")
plt.ylabel("SSE")
```

```
Text(0, 0.5, 'SSE')
```



Assignment:

تمرین:

کدهای ارائه شده در درس را بررسی و اجرا کنید.

برای خود یک دیتاست فیک بسازید و با استفاده از K-Mean دسته بندی کنید.