Practice Unsupervised Modelling

Red & White Consulting Partners LLP

Coffee Break

10:00 - 10:15

Table of Content

Introduction to Unsupervised Learning

What is Unsupervised Learning

- We let the model learn independently to identify information/trends not visible to the human eye
- Uses machine learning algorithm to perform task from **unlabeled data** **WITHOUT** human intervention

Unsupervised

Customer	Balance	Spend
A	$ 20,000	$ 5,000
B	$ 3,000	$ 2,000
C	$ 25,000	$ 4,000
D	$ 35,000	$ 15,000
E	$ 4,000	$ 2,500

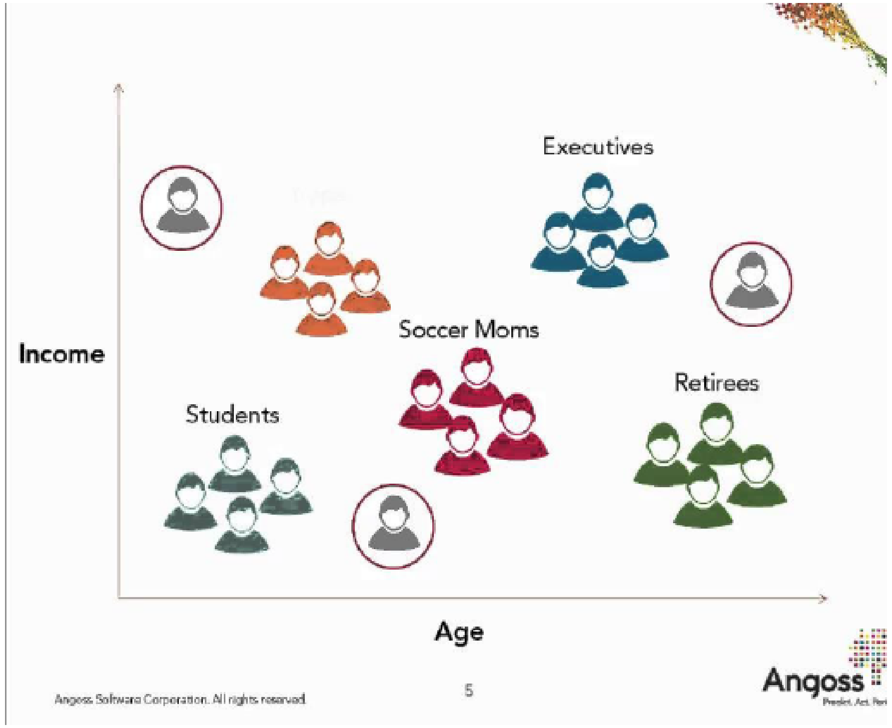
Supervised

Customer	Balance	Unpaid Tagging
A	$2,000	1
B	$1,000	0
C	$1,500	0
D	$500	1
E	$4,500	0

Label

Application of Unsupervised Learning

Customer Segmentation



Dimensionality Reduction

Customer	Income	Balance	Age	Product	Account
A	$1,000	$4,500	41	2	3
B	$2,000	$6,300	23	3	3
C	$3,000	$7,200	35	1	2
D	$4,000	$1,800	55	4	4
E	$5,000	$900	21	2	3

Customer	VAR 1	VAR 2	VAR 3
A	$2,750	41	2.50
B	$4,150	23	3.00
C	$5,100	35	1.50
D	$2,900	55	4.00
E	$2,950	21	2.50

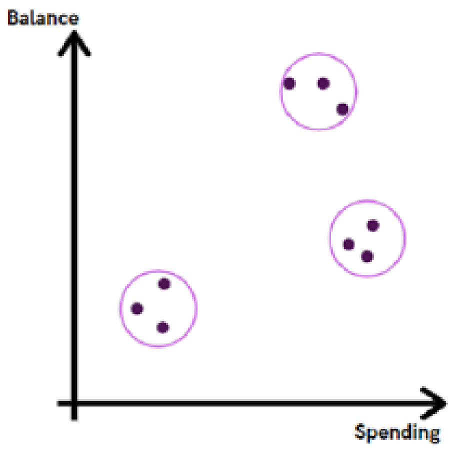
Cluster Analysis

- Cluster Analysis is a part of unsupervised learning, as no class values of data is given.

- It is a common statistical technique in many fields

- Its objective is to group(cluster) data points with similar attributes

- It groups data near (similar to) each other in one cluster, and far from (very different) each other in a different cluster



Common Clustering Algorithm

The most common clustering algorithms are the following:

1. **K-Means Clustering**
2. **Hierarchical Clustering**
3. **Gaussian Mixture Clustering**

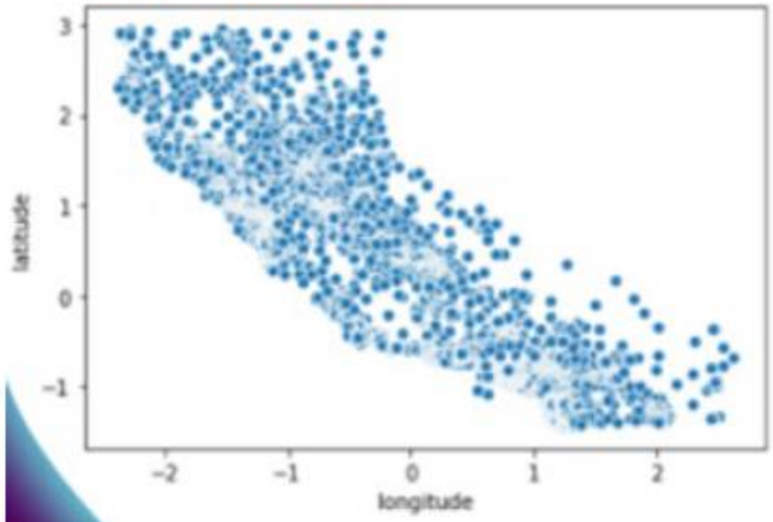
The quality of a clustering result will closely depend on the algorithm, the distance function and its application

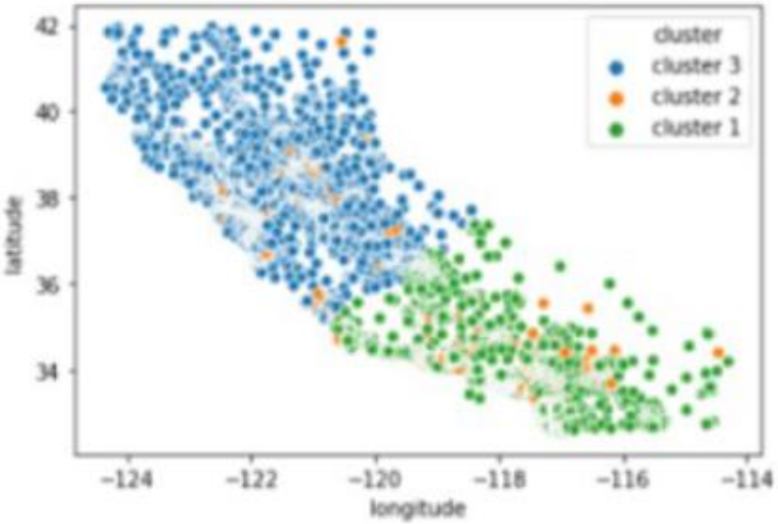
Unsupervised Learning: K-Means Clustering

K-Means Clustering

- K-Means clustering is an iterative partitional clustering algorithm which aims to partition the data into a pre-specified number of clusters (K Clusters)

- *k* is specified by the user

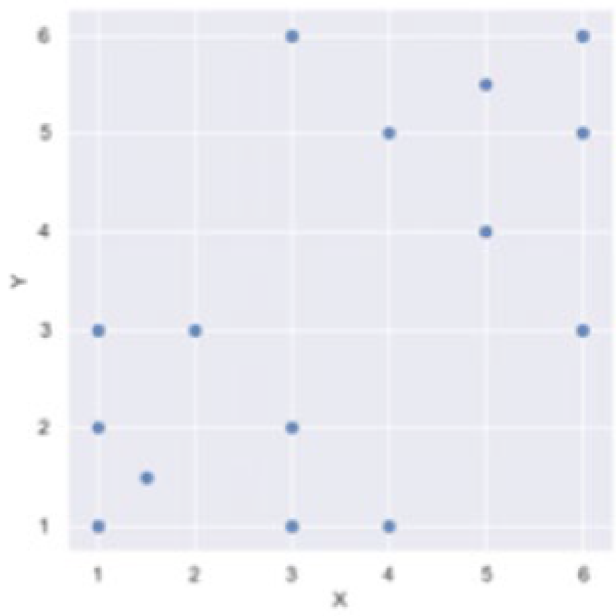


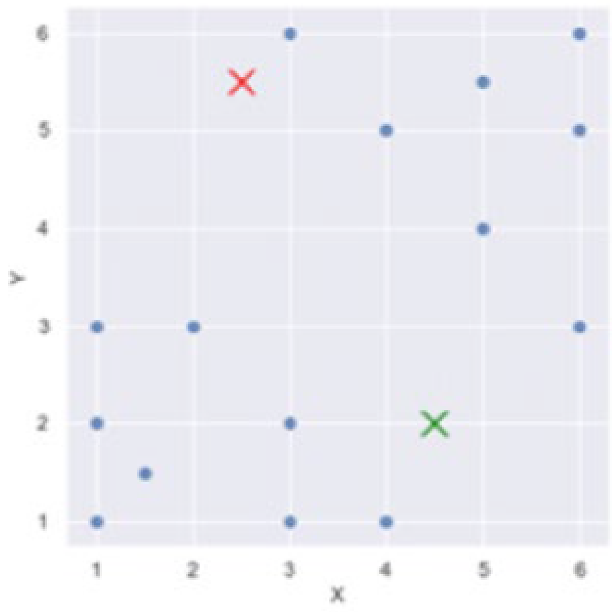


Process of K-Means Cluster

Given a value of *k*, the k-means algorithm works as follows

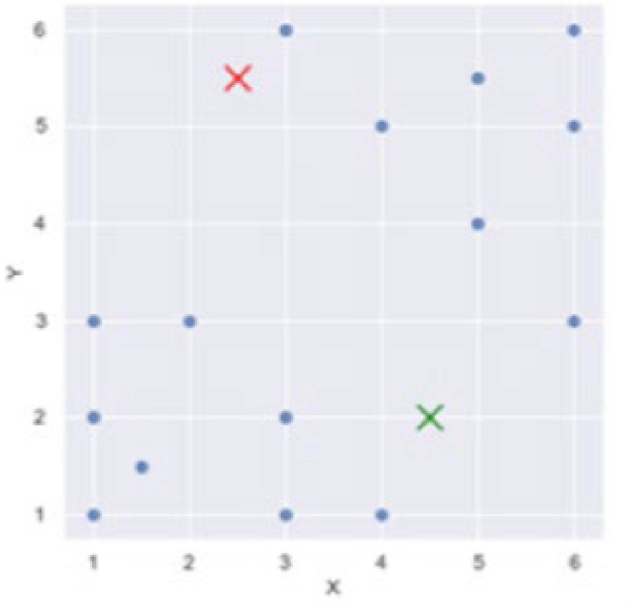
1. Randomly choose k data points (seeds) to be the initial cluster centers (centroids)

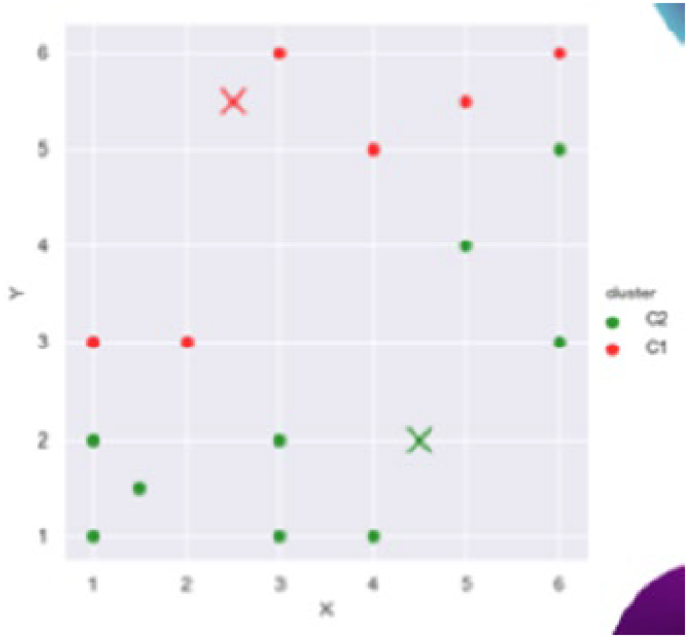




Process of K-Means Cluster

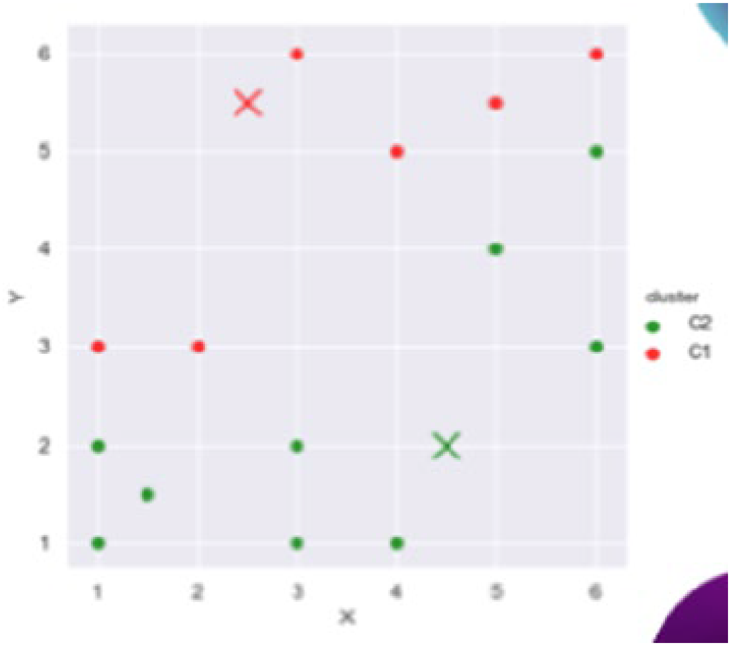
2. Assign each data point to the closest centroid using a distance measure

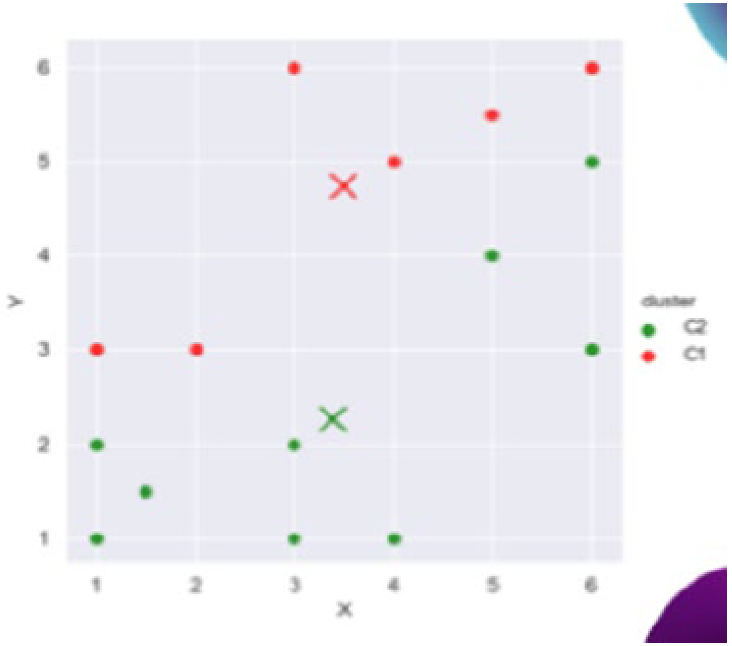




Process of K-Means Cluster

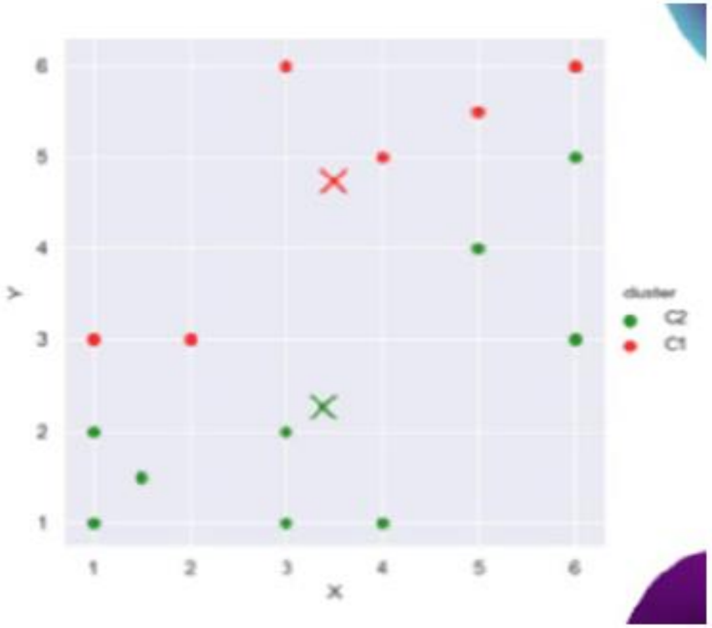
3. Re-compute the centroid using the current cluster membership

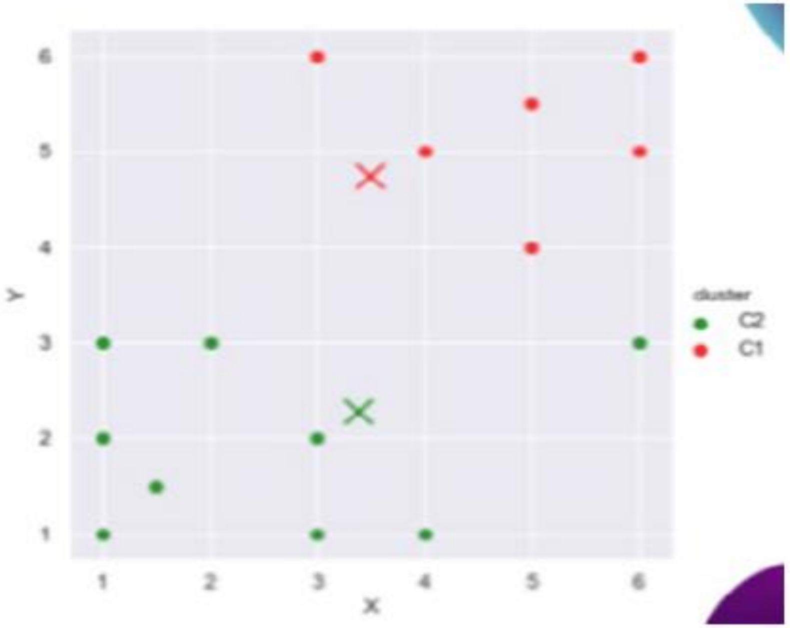




Process of K-Means Cluster

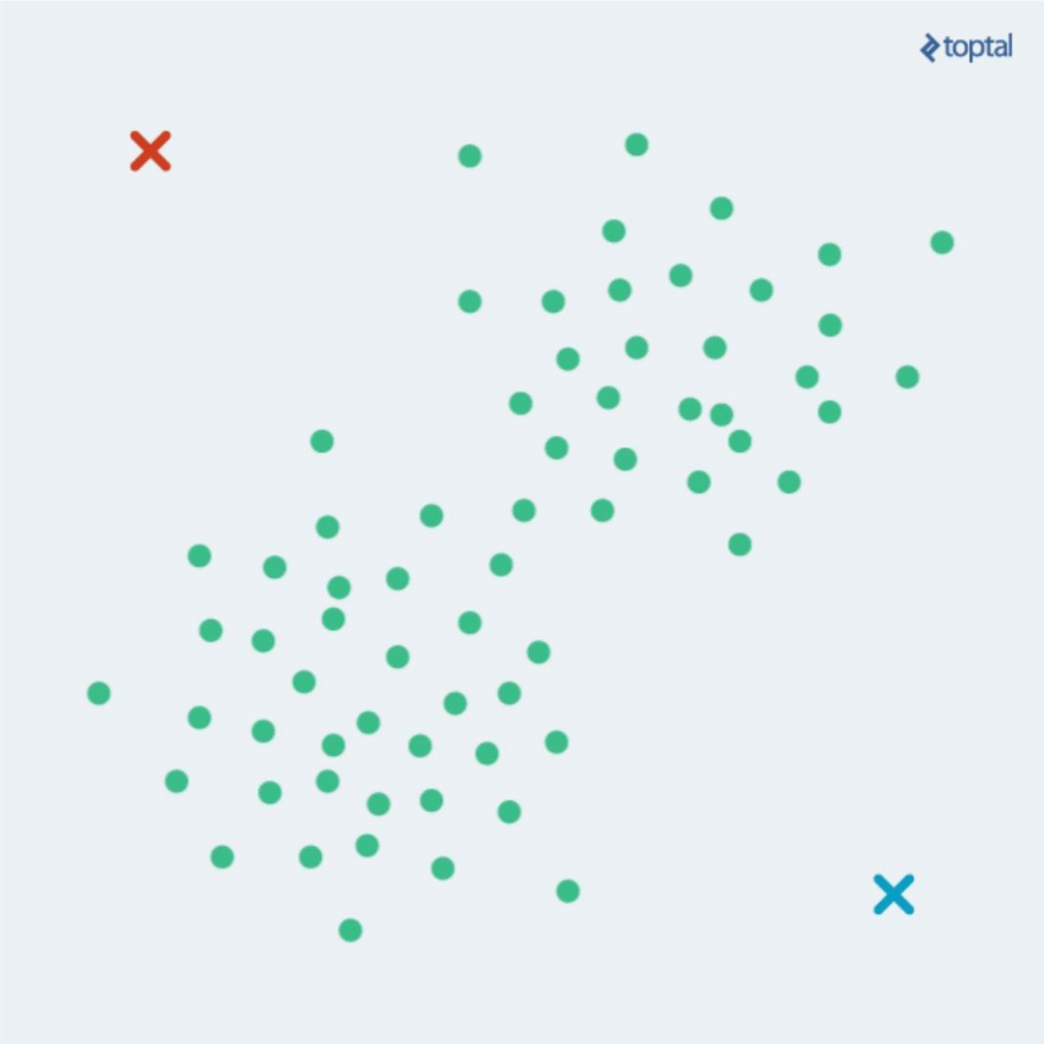
4. Re-assign the data points to the different clusters by considering the new cluster centers





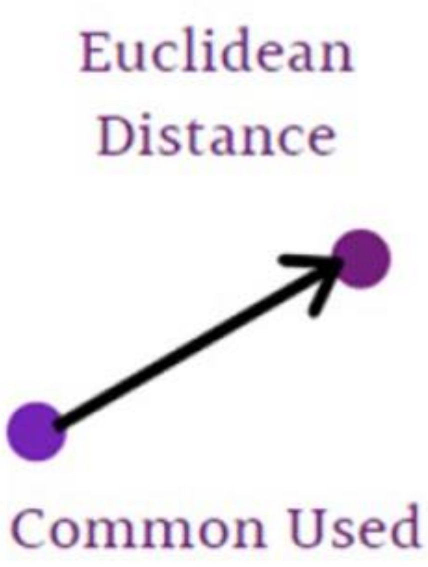
Process of K-Means Cluster

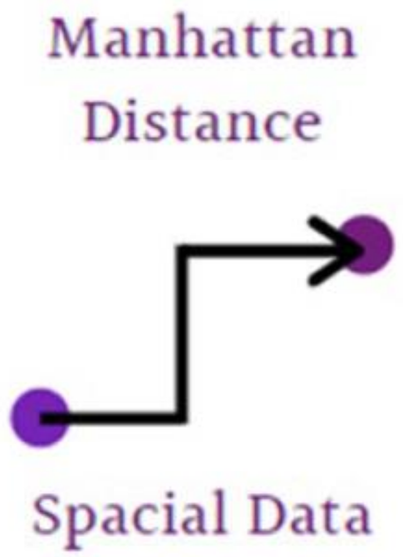
5. Keep iterating until no further movement of data points is possible



Distance Calculation

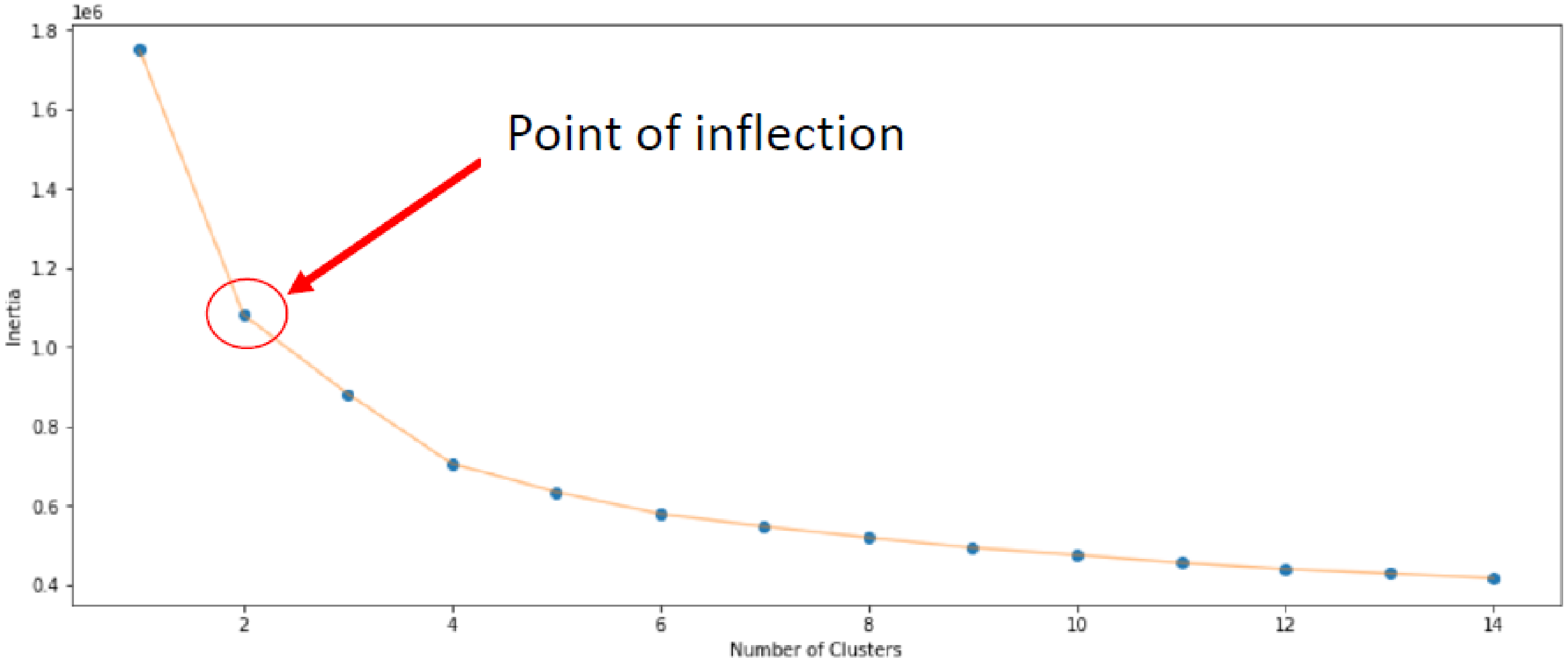
There are several ways to calculate distance, the examples are as follows





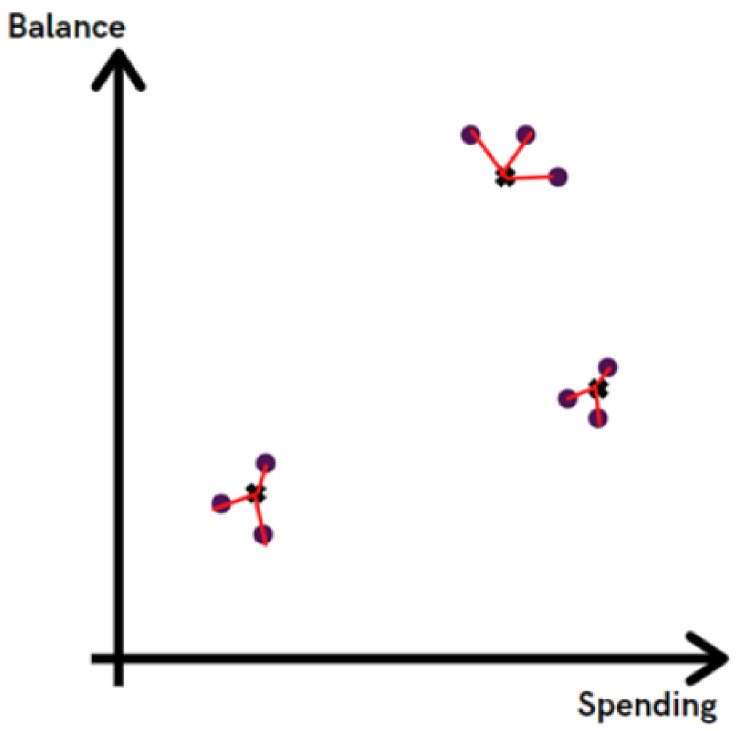
Defining K using Elbow Curve

We can define K using Elbow Curve by choosing the first point where the slope line starts to become straight line.



Defining K using Elbow Curve

The calculation used for Elbow Curve is by averaging the distance for each data into their centroids



Technical Details

Import Package

- Import Package Required for Creating K-Means Clustering

Import Package

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: from sklearn.preprocessing import StandardScaler
        from sklearn.cluster import KMeans, AffinityPropagation
        import warnings
        warnings.filterwarnings("ignore")
```

Import Data

- Import Package Required for Creating K-Means Clustering

4 Read Data

In [3]:
```
df_rm = pd.read_csv('Data.csv', low_memory = False)
```

In [4]:
```
df_rm.head()
```

Out[4]:

	EMPLOYEE_ID	RM_Type	Grade	Salary	INVESTMENT_Q1	INVESTMENT_Target_Q1	Scorecard_Q1	REVENUE_Q1	REVENUE_Target Q1	REVENUE_SCORE
0	124011005.0	RM	Senior	13000000.0	129495263.0	77215139.0	1.34	142444789.0	115822708.0	
1	124011007.0	RM	Senior	13000000.0	337443931.0	217108672.0	0.81	371188324.0	325663008.0	
2	124011010.0	RM	Medium	11000000.0	450522438.0	217621391.0	1.57	495574682.0	326432087.0	
3	124011014.0	RM	Senior	12000000.0	49306204.0	34398229.0	0.85	54236824.0	51597343.0	
4	124011015.0	RM	Senior	14000000.0	347115278.0	250258456.0	1.08	381826806.0	375387684.0	

5 rows × 52 columns

Exploratory Data Analysis (EDA)

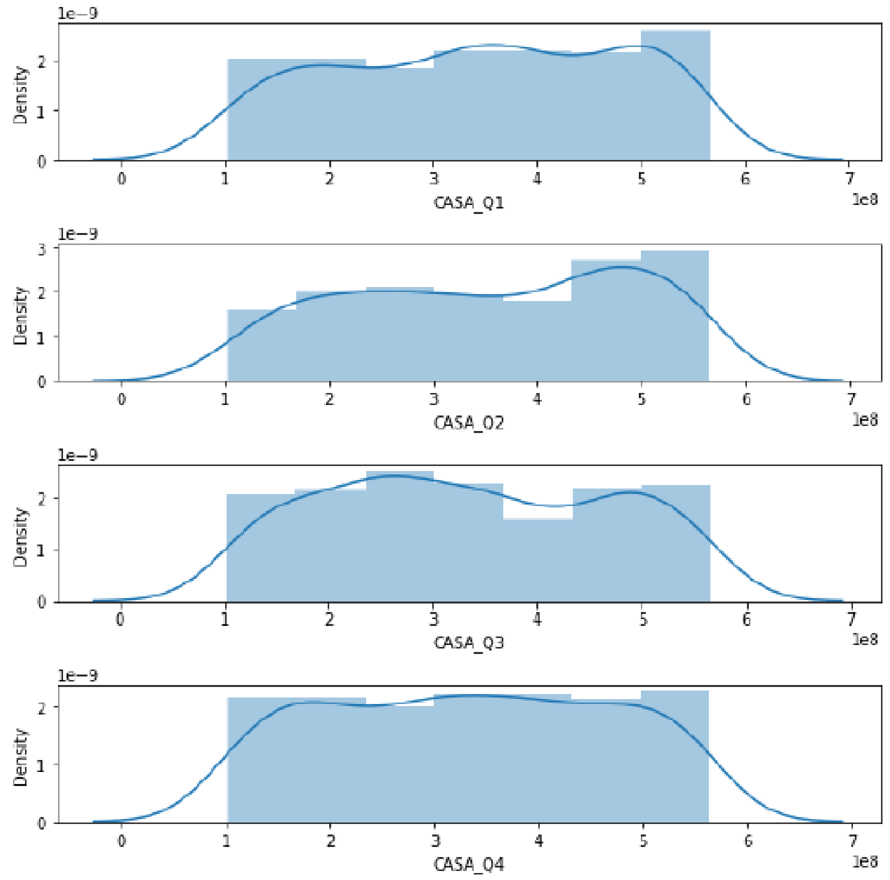
- We clean up our data first before putting the data to the cluster model

```
In [6]: df_rm.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 335 entries, 0 to 334
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   EMPLOYEE_ID              335 non-null    int64
 1   RM_Type                  335 non-null    object
 2   Grade                    335 non-null    object
 3   Salary                   335 non-null    int64
 4   Scorecard_Q1             335 non-null    float64
 5   CASA_Q1                  335 non-null    int64
 6   CASA_Target_Q1           335 non-null    int64
 7   NEW_CUSTOMER_Q1          335 non-null    int64
 8   NEW_CUSTOMER_TARGET_Q1   335 non-null    int64
 9   NEW_CUSTOMER_SCORE_Q1    335 non-null    float64
 10  Scorecard_Q2             335 non-null    float64
 11  CASA_Q2                  335 non-null    int64
 12  CASA_Target_Q2           335 non-null    int64
 13  CASA_SCORE_Q2            335 non-null    float64
 14  NEW_CUSTOMER_Q2          335 non-null    int64
 15  NEW_CUSTOMER_TARGET_Q2   335 non-null    int64
 16  NEW_CUSTOMER_SCORE_Q2    335 non-null    float64
 17  Scorecard_Q3             335 non-null    float64
 18  CASA_Q3                  335 non-null    int64
 19  CASA_Target_Q3           335 non-null    int64
 20  CASA_SCORE_Q3            335 non-null    float64
 21  NEW_CUSTOMER_Q3          335 non-null    int64
```

Data Transformation

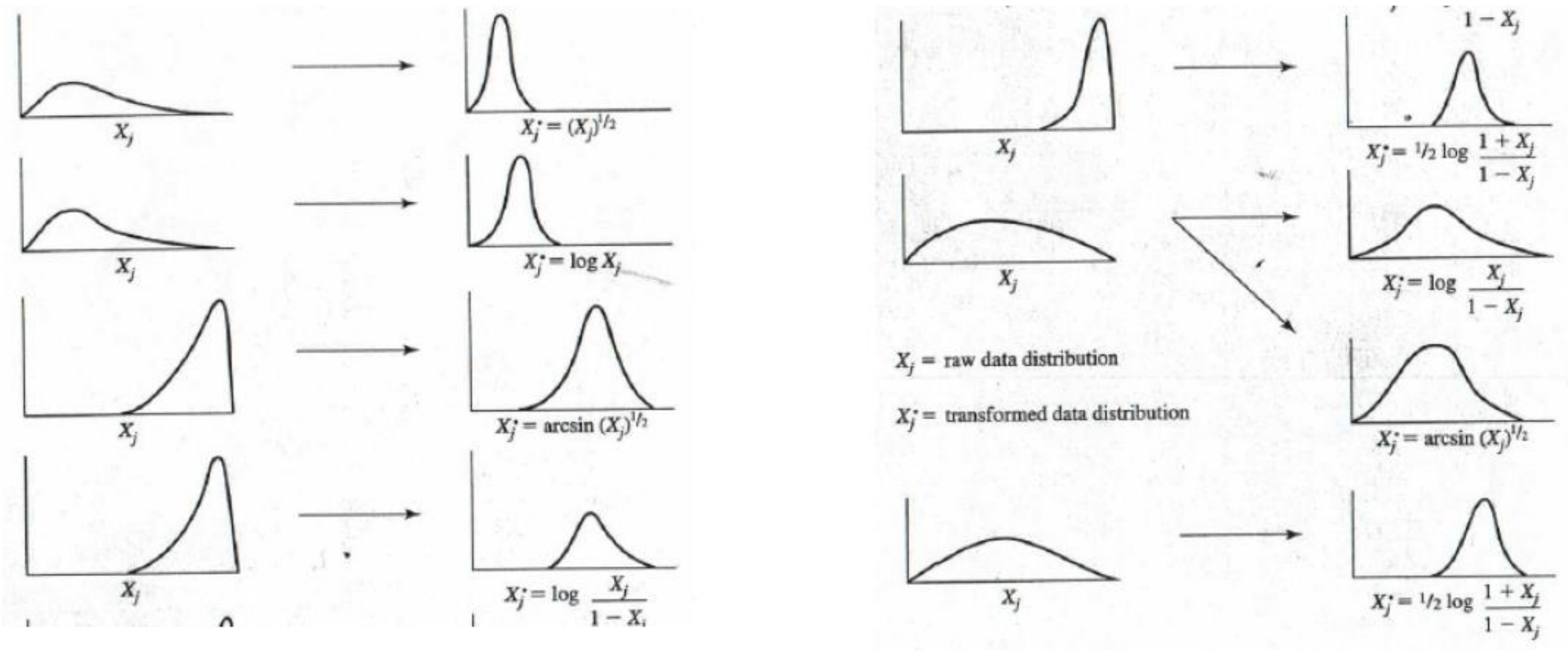
- To make sure that the cluster analysis result is better, we need to transform the data so that it follows normal distribution.

```
In [10]: def distributions(df):
             fig, (ax1, ax2, ax3, ax4) = plt.subplots(4,1, figsize=(8,8))
             sns.distplot(df["CASA_Q1"], ax=ax1)
             sns.distplot(df["CASA_Q2"], ax=ax2)
             sns.distplot(df["CASA_Q3"], ax=ax3)
             sns.distplot(df["CASA_Q4"], ax=ax4)
             plt.tight_layout()

In [11]: distributions(df_rm)
```



Data Transformation

- To make sure that the cluster analysis result is better, we need to transform the data so that it follows normal distribution.



One-Hot Encoder

- Use One-Hot Encoder for Categorical Data

```
In [19]: cluster_onehot = pd.get_dummies(df_rm, columns = ["Grade"])
         cluster_onehot.head()
```

Out[19]:

	EMPLOYEE_ID	RM_Type	Salary	Scorecard_Q1	CASA_Q1	CASA_Target_Q1	NEW_CUSTOMER_Q1	NEW_CUSTOMER_TARGET_Q1	NEW_CUSTOMER_SC(
0	124011005	RM	13000000	1.57	460971818	256415965	2	3	
1	124011007	RM	13000000	0.32	167912817	504403168	1	4	
2	124011010	RM	11000000	1.71	421422066	217200770	4	5	
3	124011014	RM	12000000	0.55	229267934	377369157	1	3	
4	124011015	RM	14000000	1.13	298354735	299262719	5	3	

5 rows × 33 columns

Standard Scaler

- Use Standard Scaler for Categorical Data

In [25]:
```
sc=StandardScaler()
cluster_scaled = pd.DataFrame(sc.fit_transform(cluster_num))
cluster_scaled.columns=cluster_num.columns
cluster_scaled.head()
```

Out[25]:

	CASA_Q1	CASA_Q2	CASA_Q3	CASA_Q4
0	0.847469	-0.431375	1.287372	-0.242068
1	-1.301909	-1.254405	1.245650	0.865305
2	0.557400	1.203510	-1.597003	-0.693226
3	-0.851913	0.567075	-0.531253	1.204633
4	-0.345211	-0.114492	-1.383600	-1.648796

Combine Numerical & Categorical Data

- Combine both Numerical & Categorical Data

Combine Data

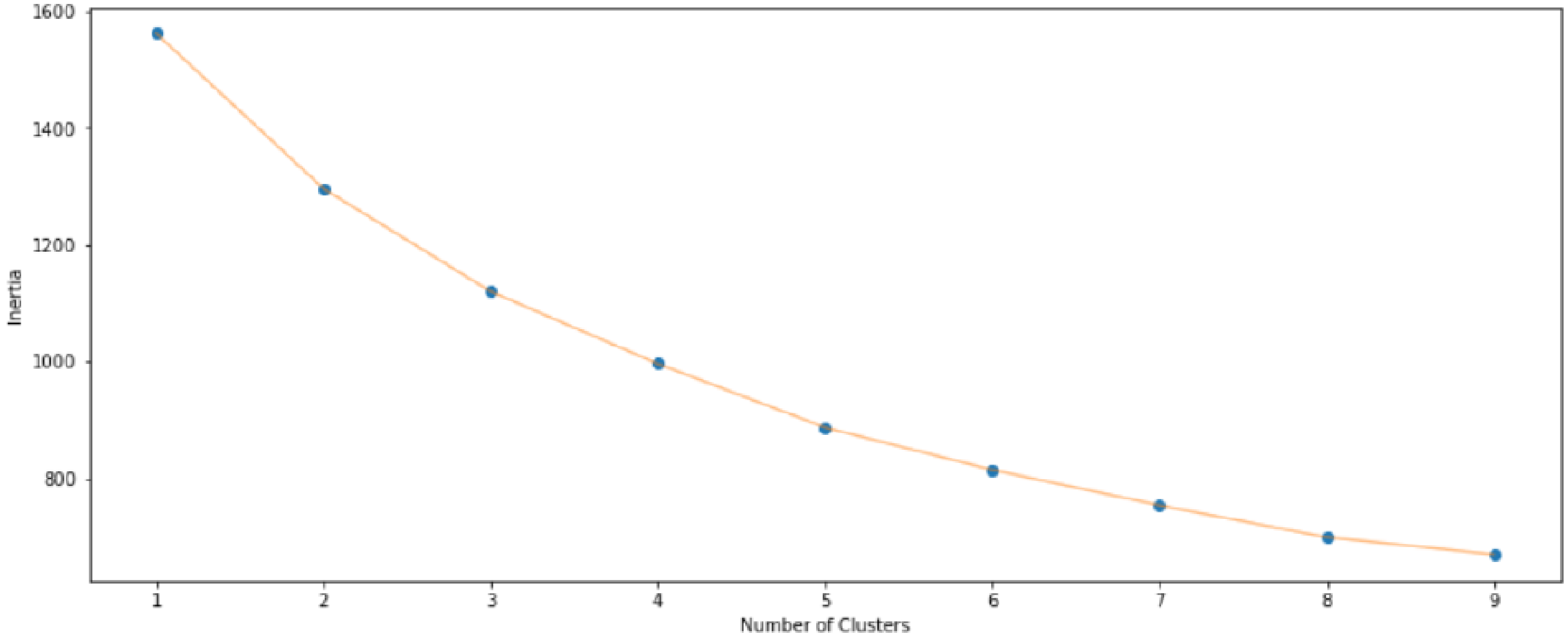
In [26]:
```
cluster =  pd.concat([cluster_scaled, cluster_onehot], axis=1)
cluster.head()
```

Out[26]:

	CASA_Q1	CASA_Q2	CASA_Q3	CASA_Q4	Grade_Junior	Grade_Medium	Grade_Senior
0	0.847469	-0.431375	1.287372	-0.242068	0	0	1
1	-1.301909	-1.254405	1.245650	0.865305	0	0	1
2	0.557400	1.203510	-1.597003	-0.693226	0	1	0
3	-0.851913	0.567075	-0.531253	1.204633	0	0	1
4	-0.345211	-0.114492	-1.383600	-1.648796	0	0	1

Elbow Curve

In [27]:
```
wcss = []
for i in range(1, 10):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(cluster)
    wcss.append(kmeans.inertia_)
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 10) , wcss , 'o')
plt.plot(np.arange(1 , 10) , wcss , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')
plt.show()
```



Cluster K-Means = 2

- Calculate the cluster and put the tagging result to the data.

Clustering K-Means K=2

In [33]:
```
kmeans = KMeans(n_clusters=2, random_state = 42)
kmeans.fit(cluster)
mapping_dict = { 0: 'Cluster 1', 1: 'Cluster 2'}
mapped_predictions = [ mapping_dict[x] for x in kmeans.labels_]
df_rm['Cluster_KM_2']=mapped_predictions
df_rm.head()
```

Out[33]:

Scorecard_Q4	CASA_Q4	CASA_Target_Q4	CASA_SCORE_Q4	NEW_CUSTOMER_Q4	NEW_CUSTOMER_TARGET_Q4	NEW_CUSTOMER_SCORE_Q4	Cluster_KM_2
0.52	299590281	546420272	0.55	2	5	0.40	Cluster 2
0.80	450707409	556917399	0.81	3	4	0.75	Cluster 1
0.54	238023137	387936763	0.61	1	4	0.25	Cluster 2
0.77	497013539	553109491	0.90	1	4	0.25	Cluster 2
0.47	107621731	324151500	0.33	4	4	1.00	Cluster 1

Cluster Evaluation

The common methods to evaluate clustering results are the following:

1. **Descriptive Analytics for each Clustering Type**
2. **Compare with Clustering Result with different number of cluster (K Number)**
3. **Compare with different Clustering Method Results**

Descriptive for Each Cluster

We can find the descriptive analytics for each cluster to determine whether the cluster method differentiate each cluster properly.

In [36]:
```
grouped_km = df_rm[['CASA_Q1', 'CASA_Q2','CASA_Q3', 'CASA_Q4', "Cluster_KM_4"]].groupby(['Cluster_KM_4']).mean().round(1)
grouped_km
```

Out[36]:

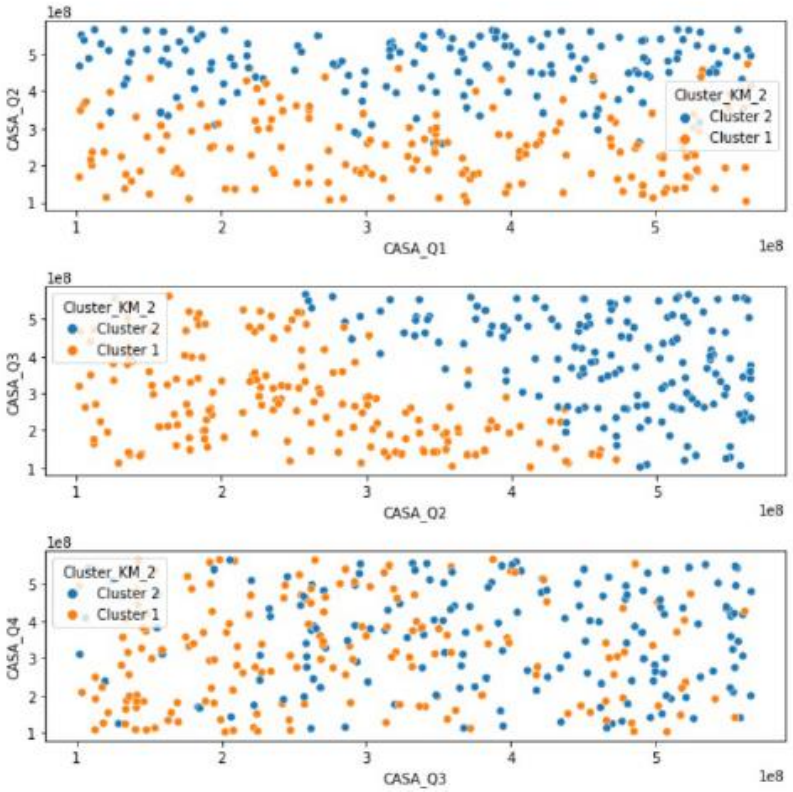
Cluster_KM_4	CASA_Q1	CASA_Q2	CASA_Q3	CASA_Q4
Cluster 1	354255152.4	202302251.3	317485971.1	364409297.9
Cluster 2	409423328.0	392750047.3	483691797.1	263192361.5
Cluster 3	305583501.4	395775486.3	200007207.4	202045881.1
Cluster 4	308864409.9	468016498.0	325975256.6	462663982.7

Descriptive for Each Cluster

We can find the descriptive analytics for each cluster to determine whether the cluster method differentiate each cluster properly.

```
In [40]: def scatters(data=df_rm, h=None, pal=None):
             fig, (ax1, ax2, ax3) = plt.subplots(3,1, figsize=(8,8))
             sns.scatterplot(x="CASA_Q1",y="CASA_Q2", hue=h, palette=pal, data=data, ax=ax1)
             sns.scatterplot(x="CASA_Q2",y="CASA_Q3", hue=h, palette=pal, data=data, ax=ax2)
             sns.scatterplot(x="CASA_Q3",y="CASA_Q4", hue=h, palette=pal, data=data, ax=ax3)
             plt.tight_layout()
```

```
In [41]: scatters(h = "Cluster_KM_2")
```



Compare with Different K Number

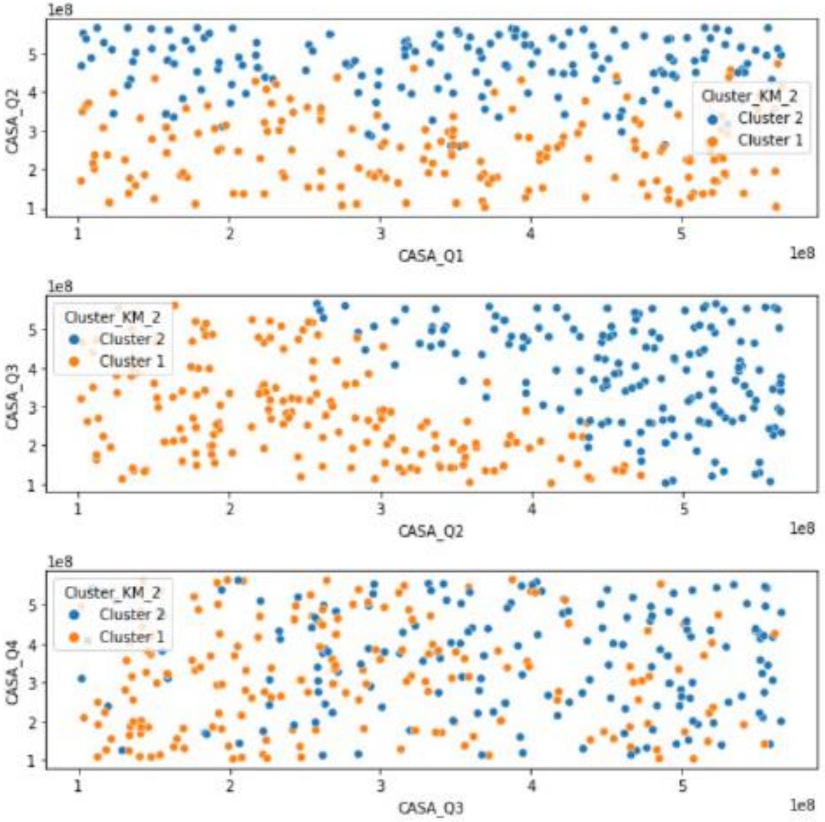
We can compare with different number of K as well to determine the goodness of clustering results.

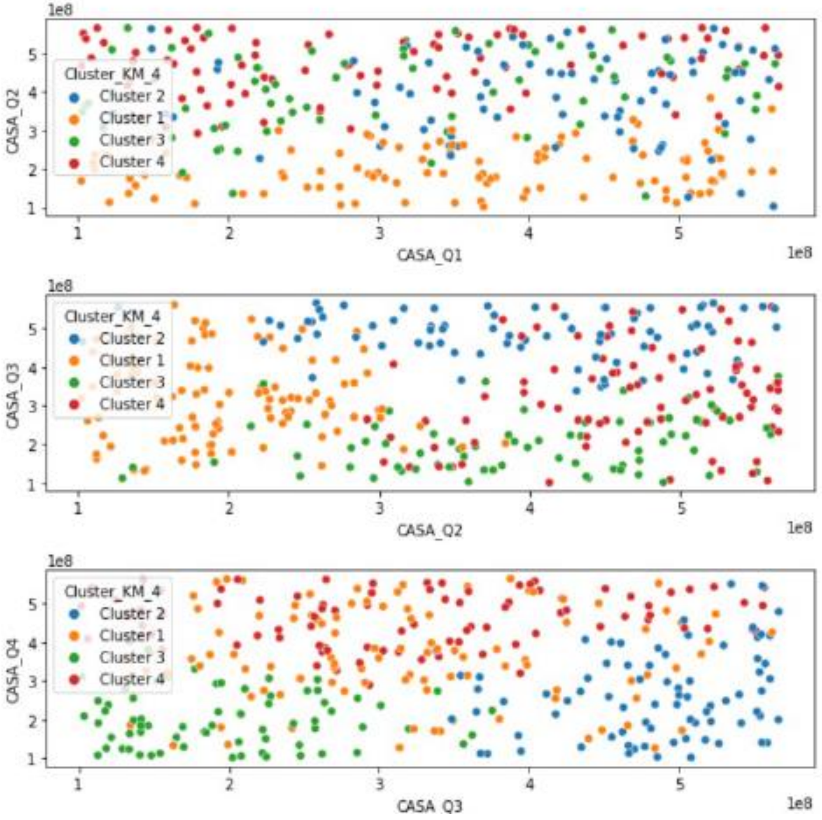
Cluster_KM_4	CASA_Q1	CASA_Q2	CASA_Q3	CASA_Q4
Cluster 1	354255152.4	202302251.3	317485971.1	364409297.9
Cluster 2	409423328.0	392750047.3	483691797.1	263192361.5
Cluster 3	305583501.4	395775486.3	200007207.4	202045881.1
Cluster 4	308864409.9	468016498.0	325975256.6	462663982.7

Cluster_KM_2	CASA_Q1	CASA_Q2	CASA_Q3	CASA_Q4
Cluster 1	340159101.8	250972540.6	282810022.4	310698715.3
Cluster 2	350977180.7	464171679.3	388359354.3	355759766.0

Compare with Different K Number

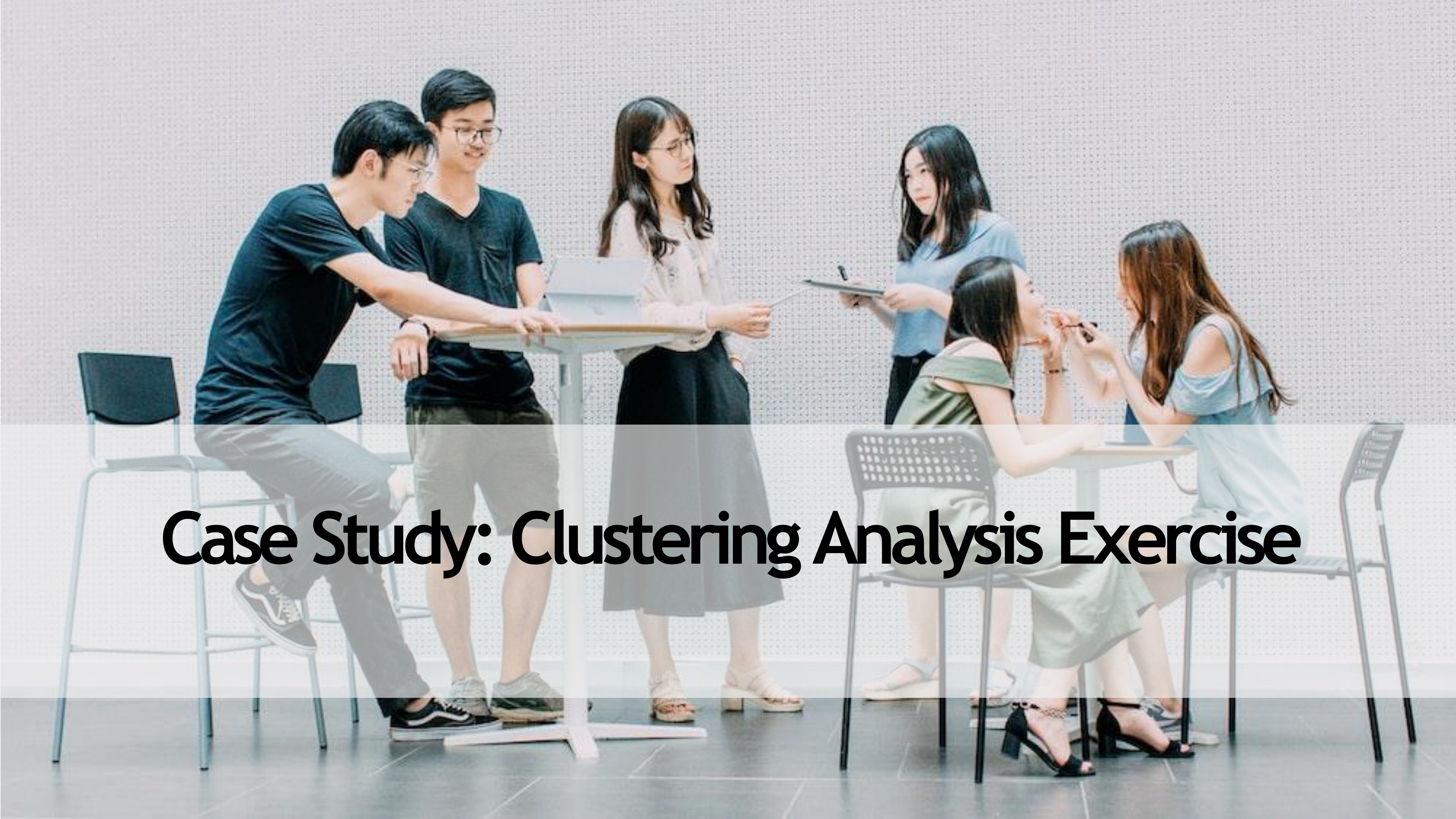
We can compare with different number of K as well to determine the goodness of clustering results.





Ishoma

12:00 - 13:00



Case Study: Clustering Analysis Exercise

Let's Share!


Q & A

Thank you