Cluster Analysis using Python

Statinfer.com

Contents

- Introduction to Segmentation & Cluster analysis
- Applications of Cluster Analysis
- Types of Clusters
- Similarity measure
- K-Means clustering
- The Algorithm
- Building clusters Python
- Deciding the cluster numbers
- Working with non-numerical data

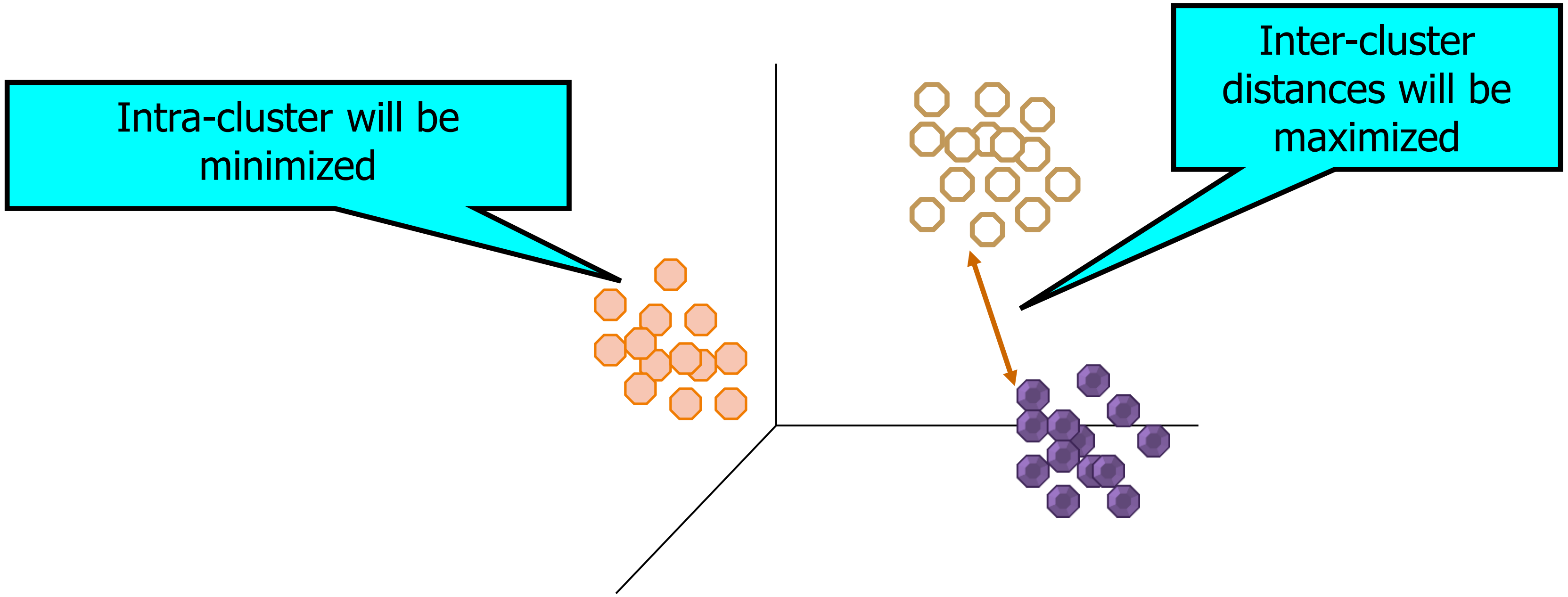
Supervised vs Unsupervised Learning

x1	x2	x3	…	xk	y

Supervised vs Unsupervised Learning

x1	x2	x3	…	xk

Segmentation and Cluster Analysis



Applications of Cluster Analysis

- **Market Segmentation:** Grouping people (with the willingness, purchasing power, and the authority to buy) according to their similarity
- **Sales Segmentation:** Clustering can tell you what types of customers buy what products
- **Operations:** High performer segmentation & promotions based on person’s performance
- **Insurance:** Identifying groups of motor insurance policy holders with a high average claim cost.

What is the need of segmentation?

Problem:

- 10,000 Customers - we know their city name, income, employment status, designation
- You have to sell 100 smart phones(each costs $1000) to the people in this group. You have maximum of 7 days
- If you start giving demos to each individual, 10,000 demos will take more than one year. How will you sell maximum number of phones by giving minimum number of demos?

What is the need of segmentation?

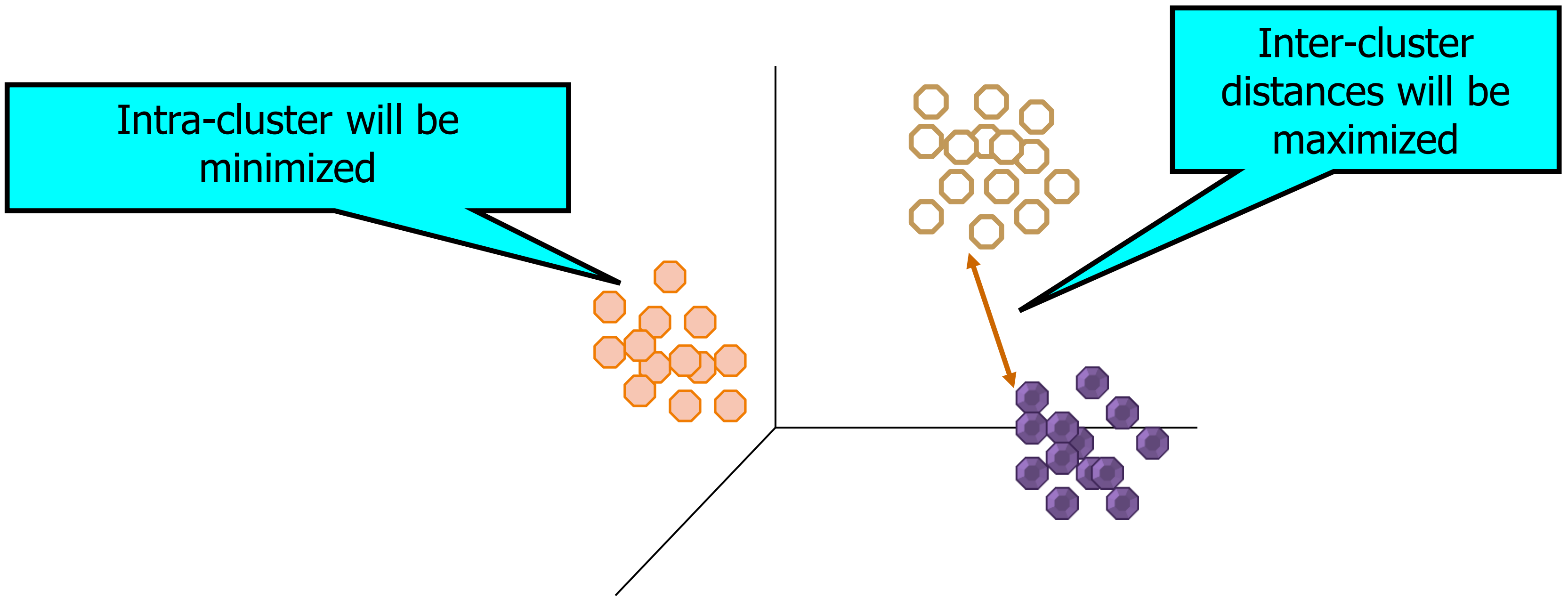
Solution

- Partition the whole population into groups
- Same type of customers should be clubbed together
- Dis-similar customers should not be in the same group

Segmentation and Cluster Analysis

- Cluster is a group of similar objects (cases, points, observations, examples, members, customers, patients, locations, etc)
- Finding the groups of cases/observations/ objects in the population such that the objects are
- Homogeneous within the group (high intra-class similarity)
- Heterogeneous between the groups(low inter-class similarity)

Segmentation and Cluster Analysis



Dissimilarity & Similarity

Dissimilarity & Similarity

	Income
Cust1	68,000
Cust2	72,000
Cust3	1,00,000

Which two customers are similar?

Dissimilarity & Similarity

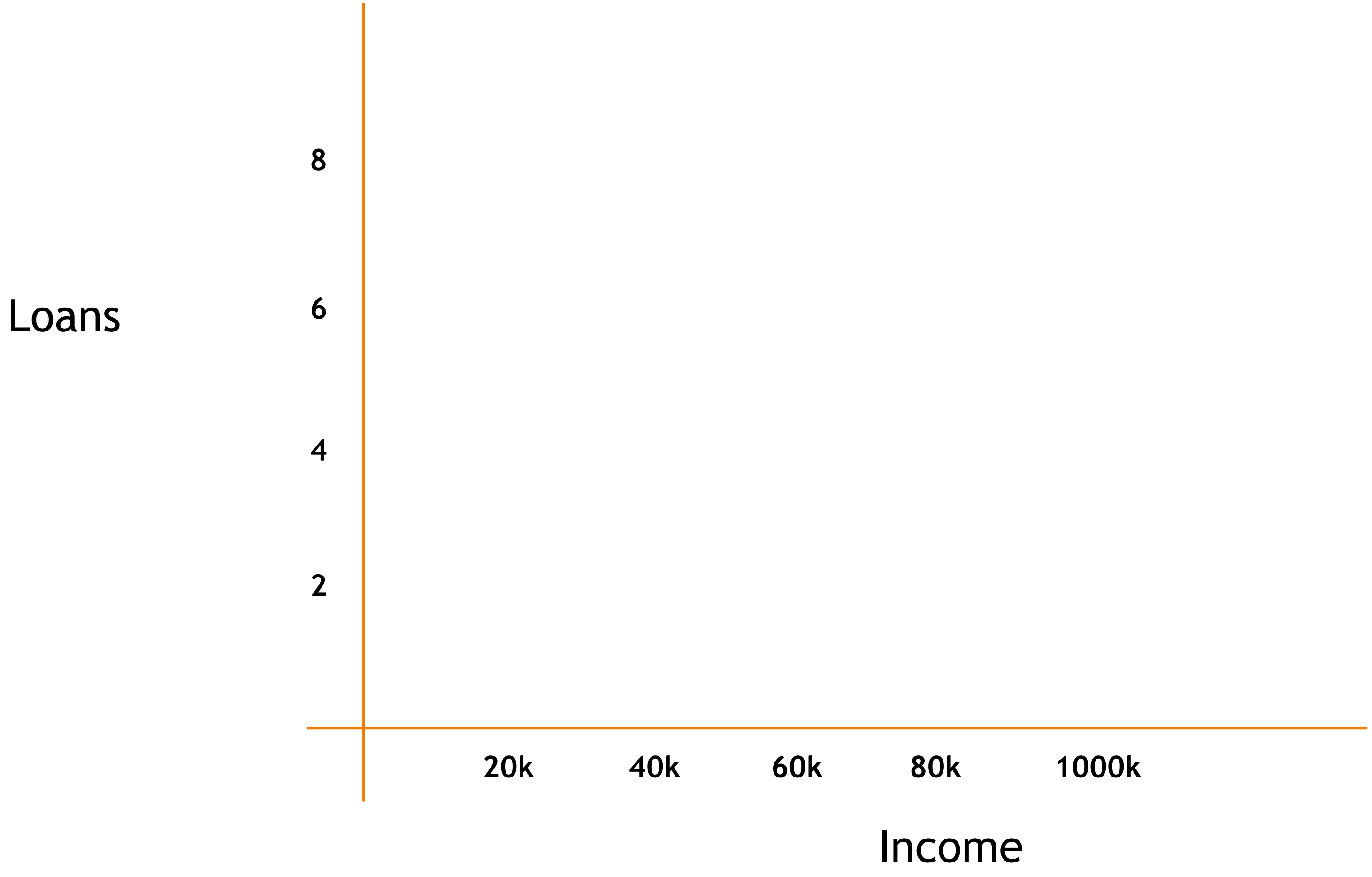
	Income
Cust1	68,000
Cust2	72,000
Cust3	1,00,000

Which two customers are similar?

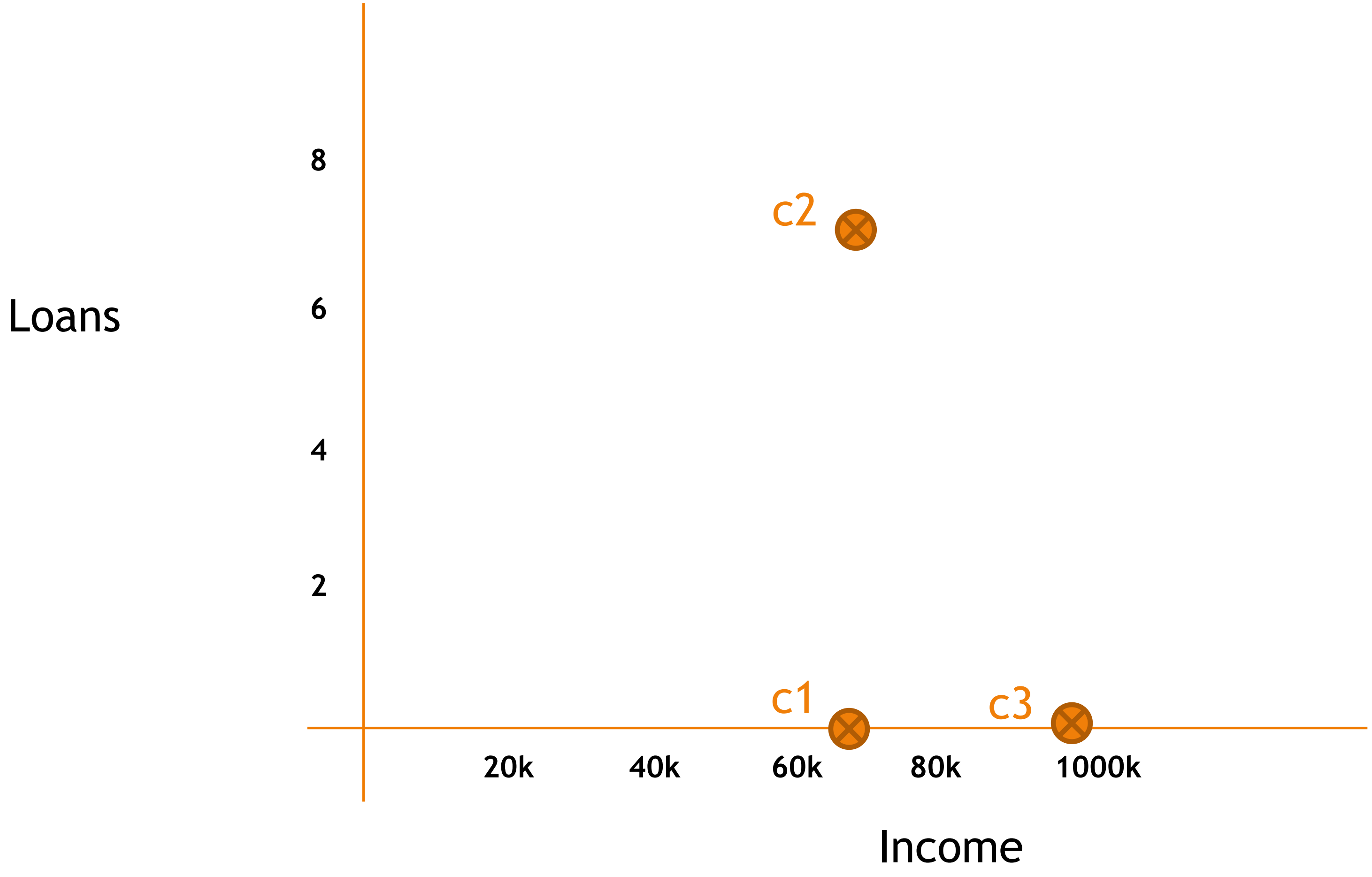
	Income	Loans
Cust1	68,000	0
Cust2	72,000	7
Cust3	1,00,000	0

Which two customers are similar now?

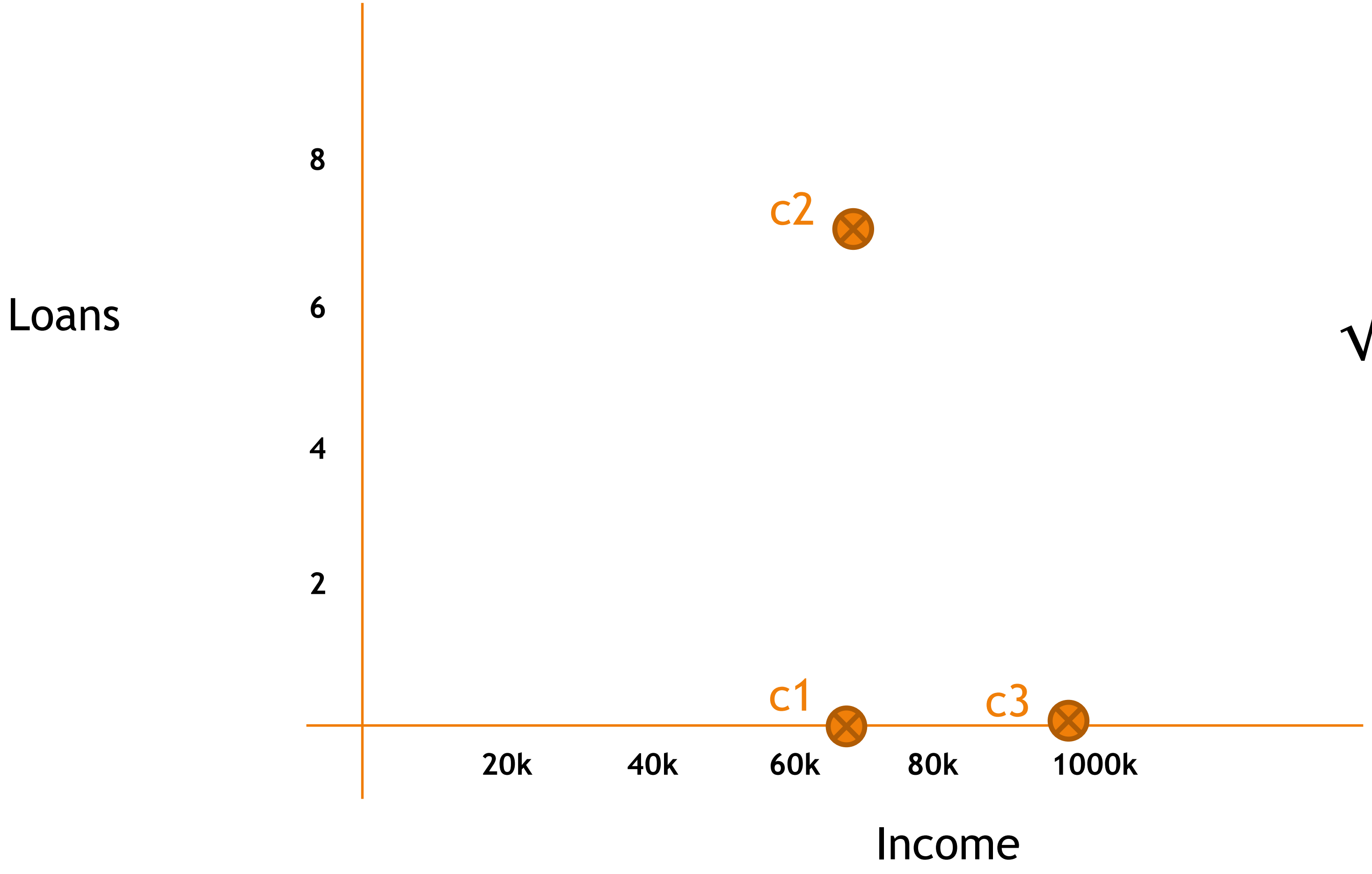
Quantify dissimilarity -Distance measures



Quantify dissimilarity -Distance measures



Quantify dissimilarity -Distance measures



Euclidian Distance

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

statinfer

Quantify dissimilarity -Distance measures

- To measure similarity between two observations a distance measure is needed. With a single variable, similarity is straightforward
- Example: income – two individuals are similar if their income level is similar and the level of dissimilarity increases as the income gap increases
- Multiple variables require an aggregate distance measure
- Many characteristics (e.g. income, age, consumption habits, family composition, owning a car, education level, job…), it becomes more difficult to define similarity with a single value
- The most known measure of distance is the Euclidean distance, which is the concept we use in everyday life for spatial coordinates.

Distance Matrix

Data matrix

Dissimilarity matrix

$$\begin{bmatrix} x_{11} & \cdots & x_{1f} & \cdots & x_{1p} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{i1} & \cdots & x_{if} & \cdots & x_{ip} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & \cdots & x_{nf} & \cdots & x_{np} \end{bmatrix}$$

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \cdots & \cdots & 0 \end{bmatrix}$$

Dissimilarity & Similarity

	Income
Cust1	68,000
Cust2	72,000
Cust3	1,00,000

Which two customers are similar?

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

	Income	Loans
Cust1	68,000	0
Cust2	72,000	7
Cust3	1,00,000	0

Which two customers are similar now?

LAB: Calculation of distance

- Import the data `Data: "./Credit_Score_Expenses/Credit_Score_Expenses.csv"`
- Calculate the pairwise distances
- Which two customers are close to each other?
- Which two customers are very dis-similar?

statinfer

Code: Calculation of distance

```
# Euclidean Distance Caculator
def distance_matrix(data_frame):
    import numpy as np
    result_distance=np.zeros((data_frame.shape[0],data_frame.shape[0]))
    for i in range(0 , data_frame.shape[0]):
        for j in range(0 , data_frame.shape[0]):
            result_distance[i,j]=round(math.sqrt(sum((data_frame.iloc[i] - data_frame.iloc[j])**2)),1)
    print(result_distance)

distance_matrix(Credit_Score_Expenses)
```

```
    ...: distance_matrix(Credit_Score_Expenses)
[[     0.   22628.1   8194.   20988.7   5678. ]
 [22628.1      0.   14439.7   1648.3 16950.3]
 [ 8194.   14439.7      0.   12804.2   2528.8]
 [20988.7   1648.3 12804.2      0.   15310.7]
 [ 5678.   16950.3   2528.8 15310.7      0. ]]
```

Examples of distances

$$\sqrt{\sum_{k=1}^{n}\left(x_{ik}-x_{jk}\right)^2}$$ Euclidian Distance

$$\sum_{k=1}^{n}\left|x_{ik}-x_{jk}\right|$$ Manhattan distance

$$r\left(x_{ik},x_{jk}\right)$$ Correlation -Similarity measure

$$\max_k\left|x_{ik}-x_{jk}\right|$$ Chebyshev distance

Other distance measures:

- Minkowski
- Mahalanobis
- maximum distance
- Cosine similarity
- Jacob's distance

statinfer

Clustering algorithms

K -Means Clustering – Algorithm

1. The number *k* of clusters is fixed
2. An initial set of *k “seeds” (aggregation centres)* is provided
   1. First *k* elements
   2. Other seeds (randomly selected or explicitly defined)
3. Given a certain fixed threshold, all units are assigned to the nearest cluster seed
4. New seeds are computed
5. Go back to step 3 until no reclassification is necessary

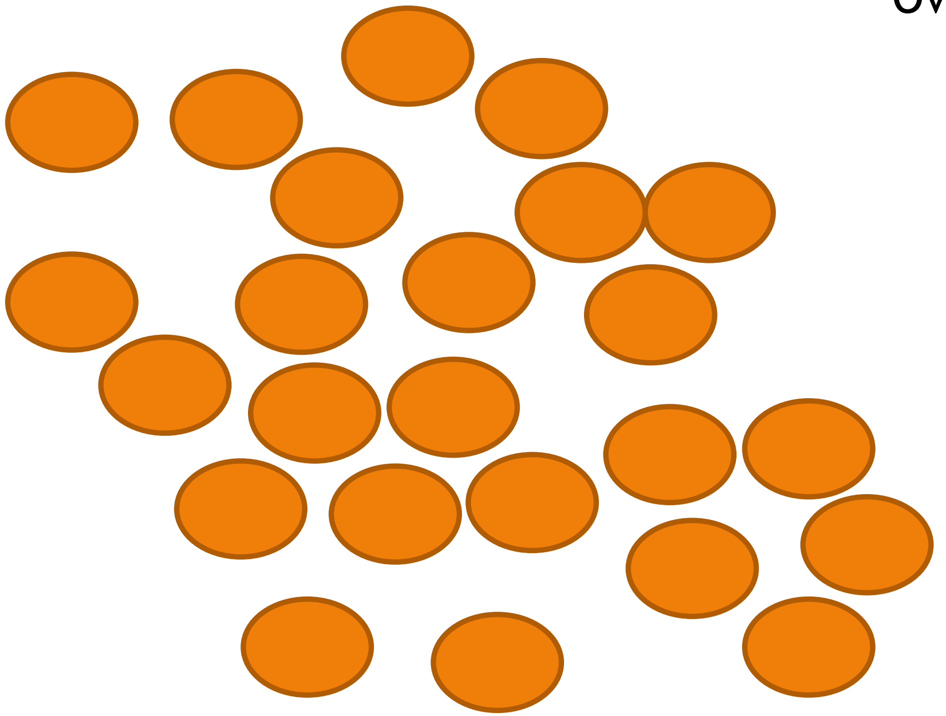
K -Means Clustering – Algorithm

In simple terms

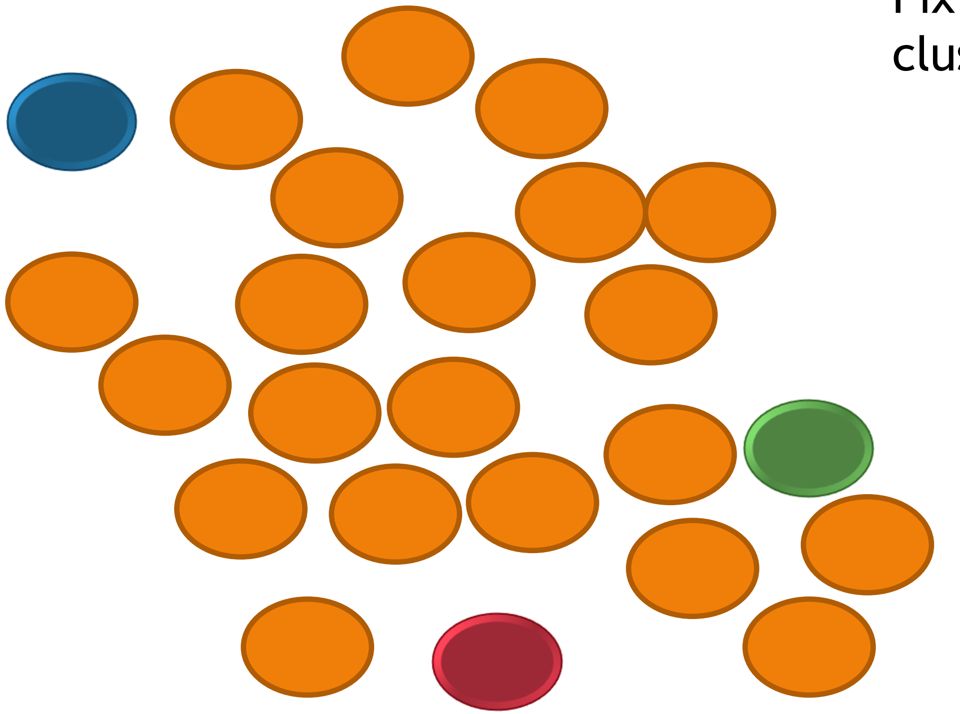
- Initialize k cluster centres
  - Do
    - Assignment step: Assign each data point to its closest cluster center
    - Re-estimation step: Re-compute cluster centers
  - While (there are still changes in the cluster centers)

K-Means clustering

Overall population

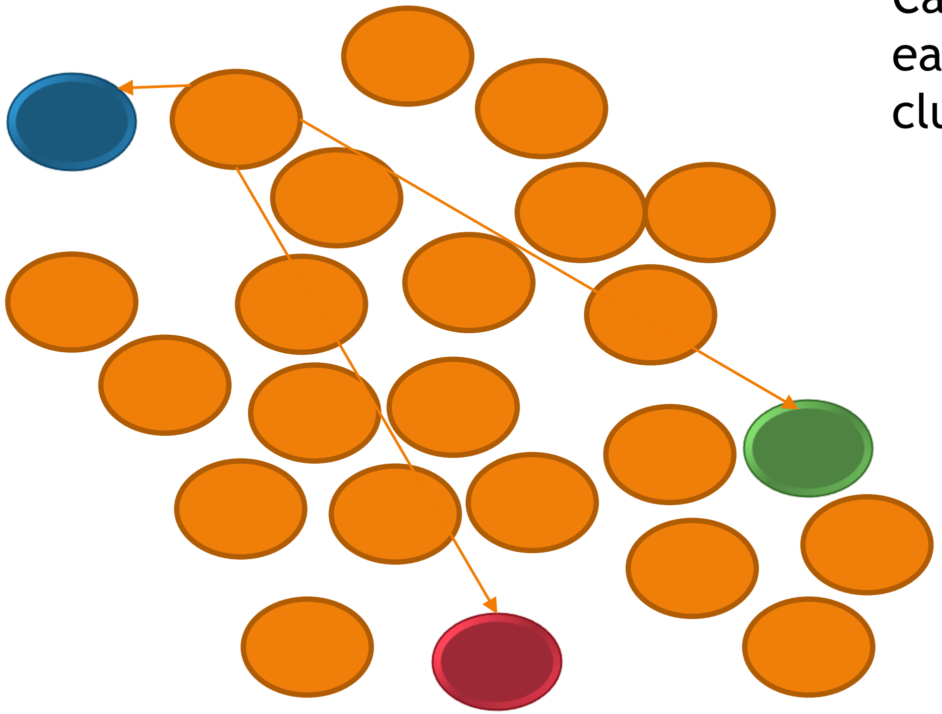


K-Means clustering



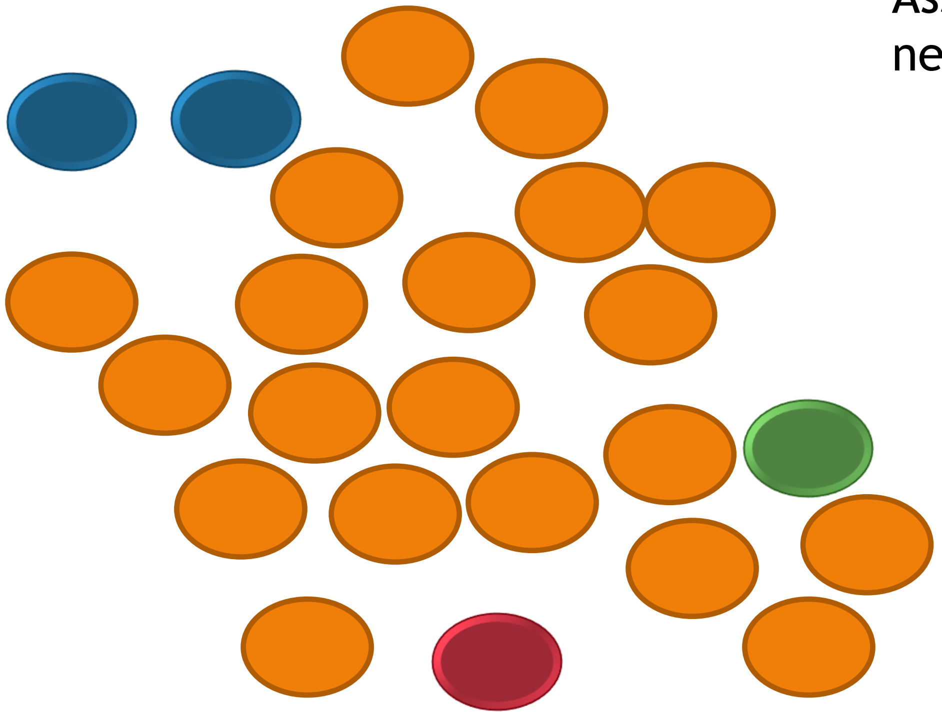
Fix the number of clusters

K-Means clustering



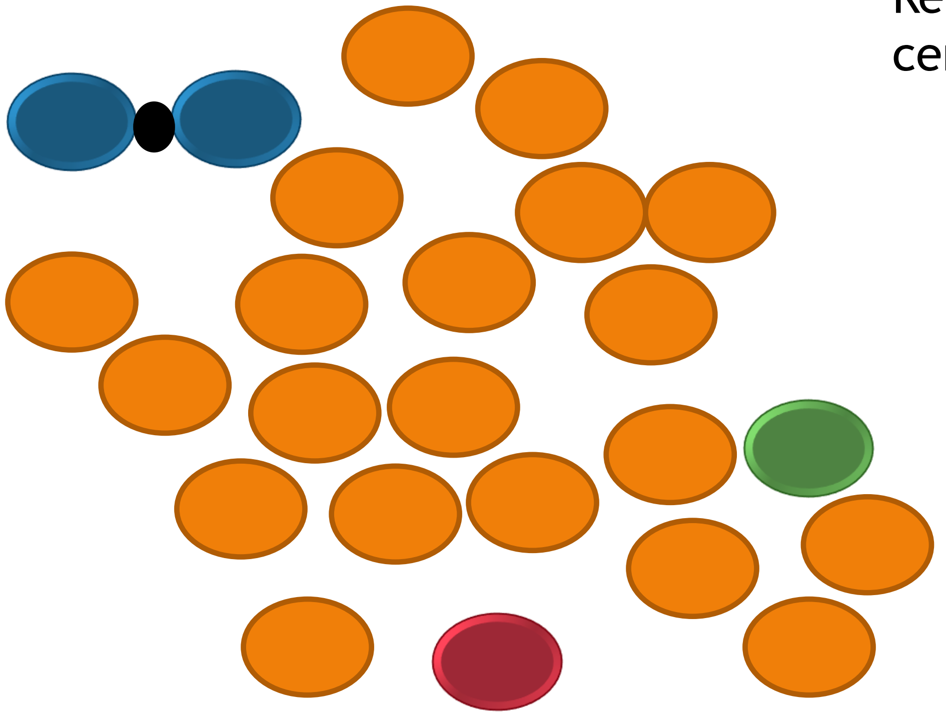
Calculate the distance of each case from all clusters

K-Means clustering



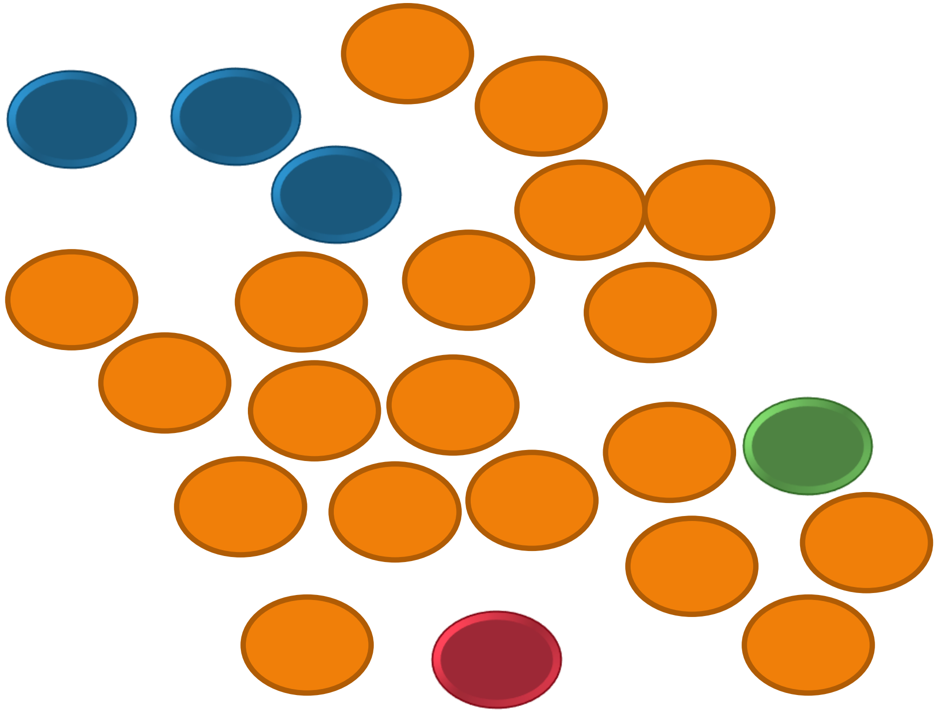
Assign each case to nearest cluster

K-Means clustering

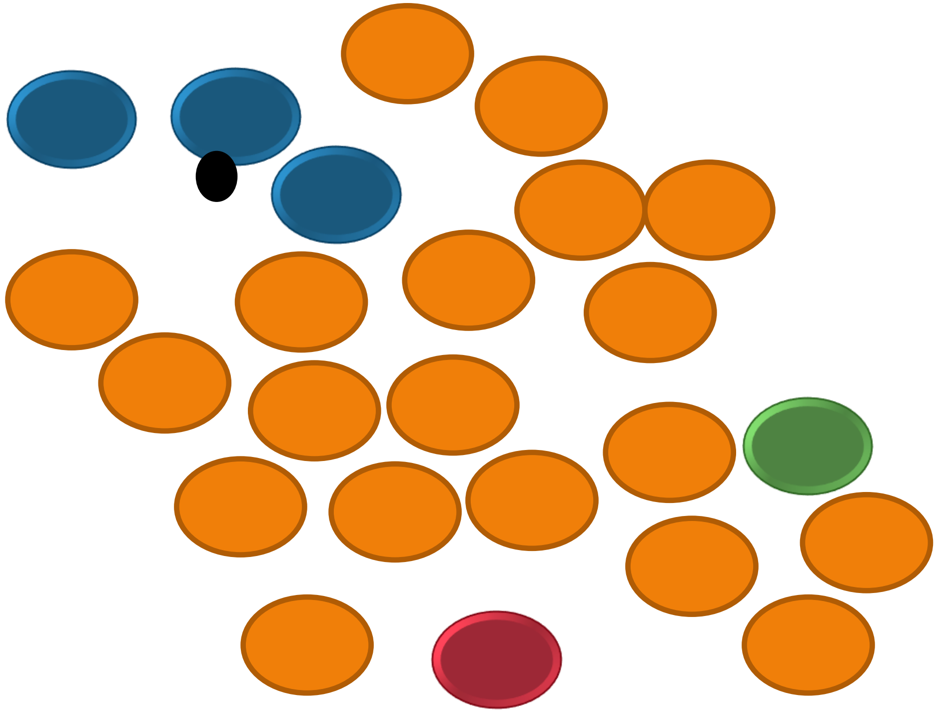


Re calculate the cluster centers

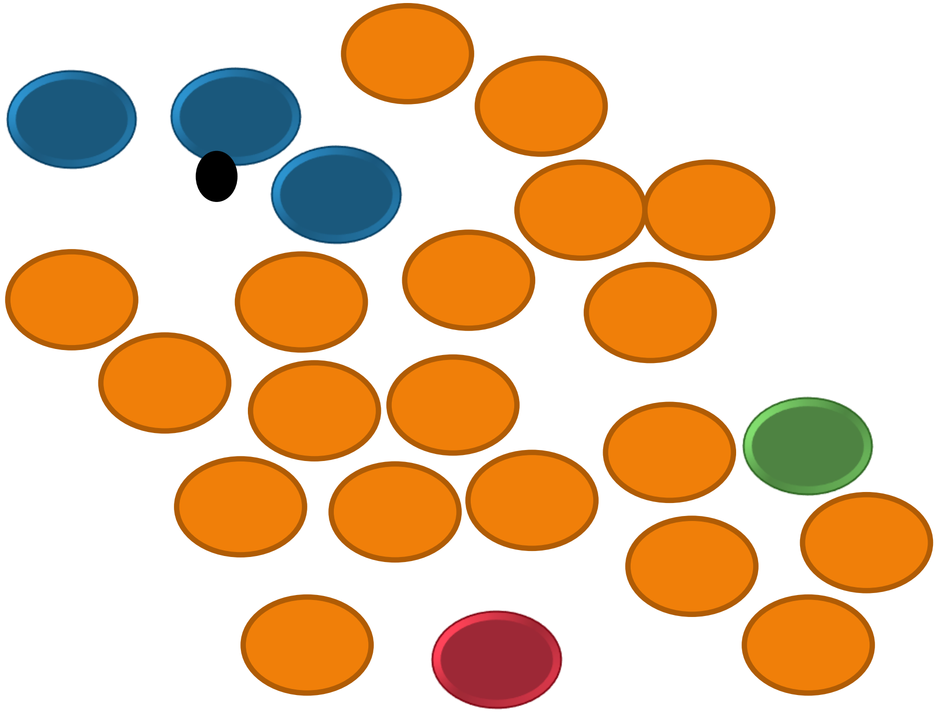
K-Means clustering



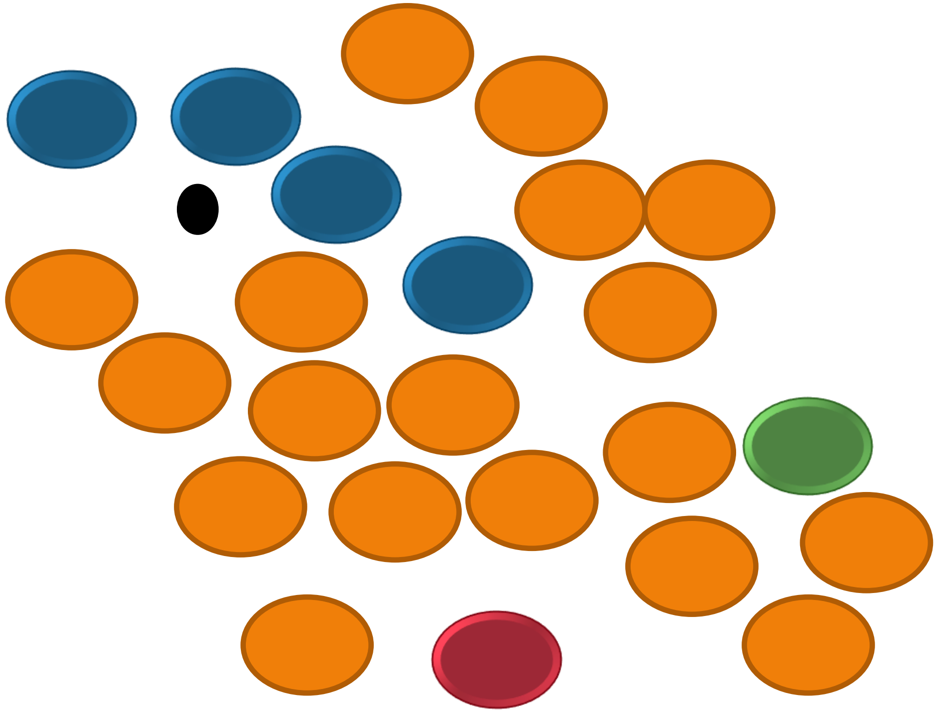
K-Means clustering



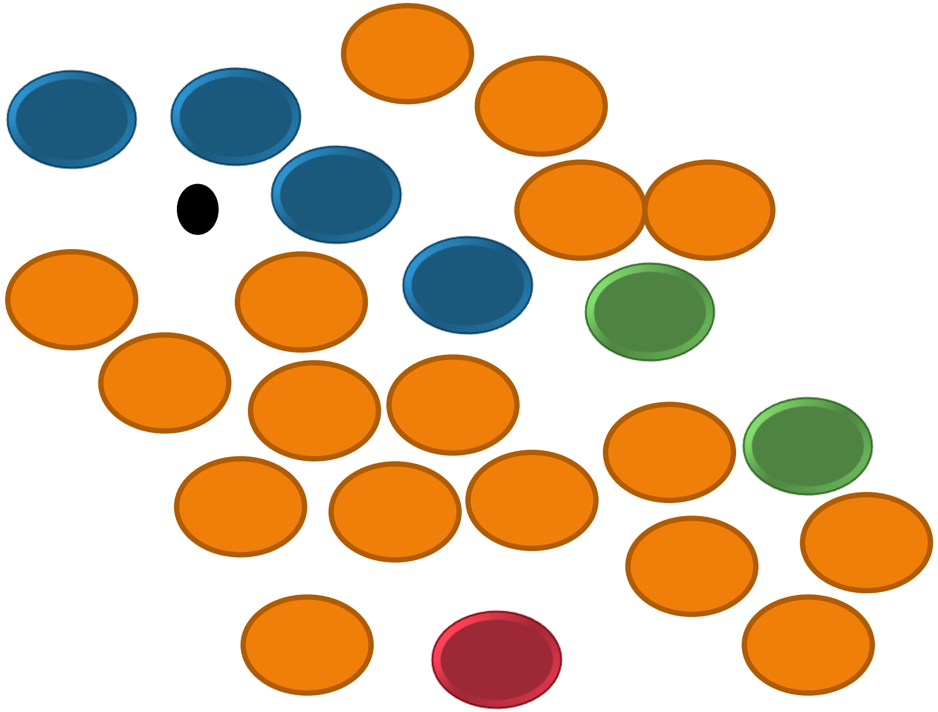
K-Means clustering



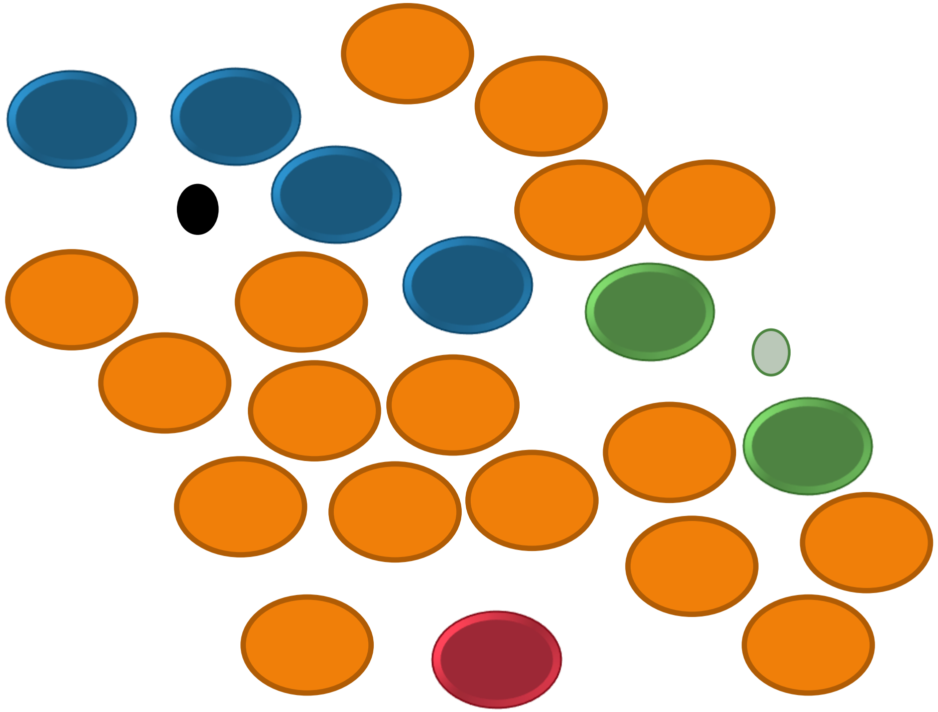
K-Means clustering



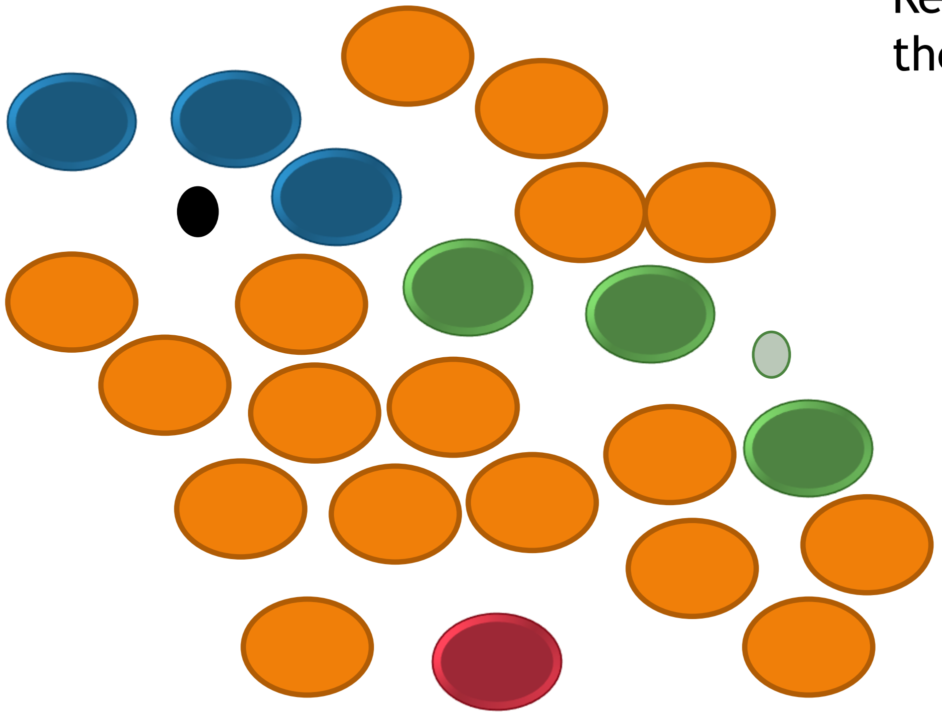
K-Means clustering



K-Means clustering

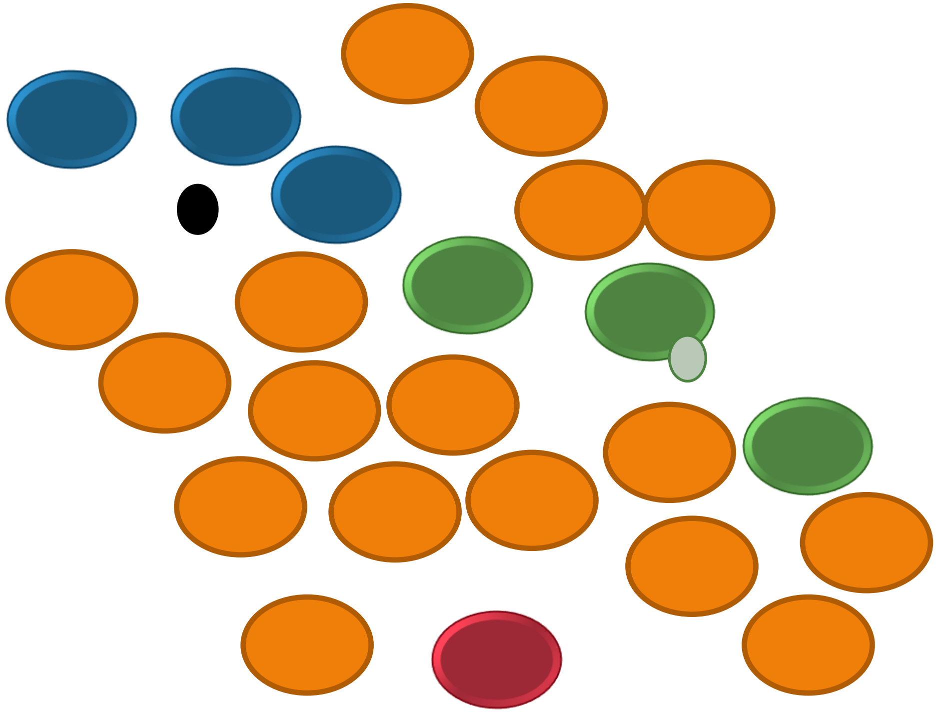


K-Means clustering

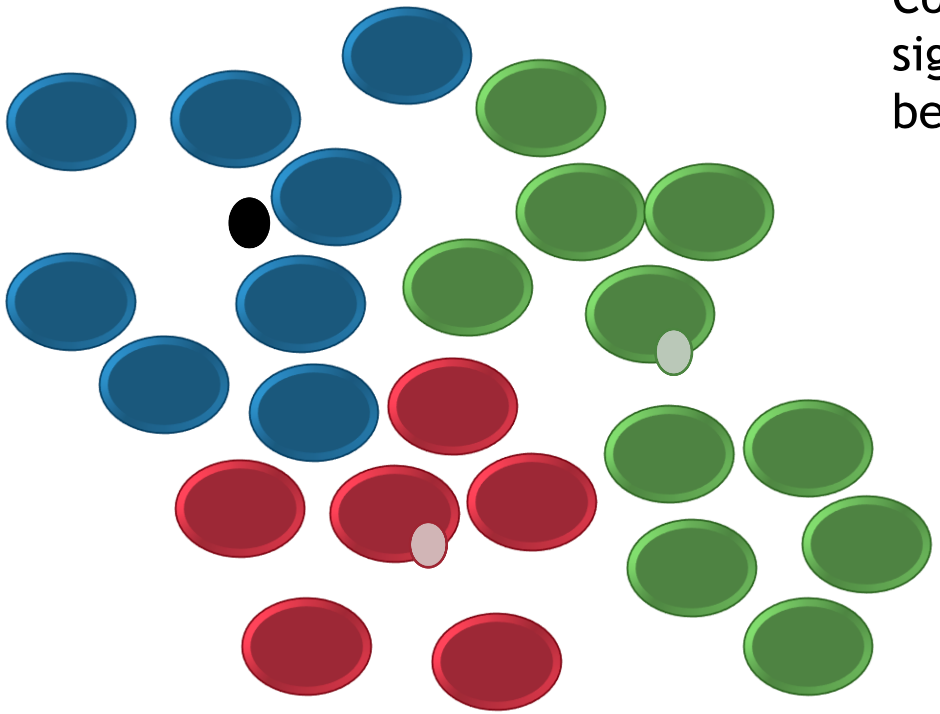


Reassign after changing the cluster centers

K-Means clustering



K-Means clustering



Continue till there is no significant change between two iterations

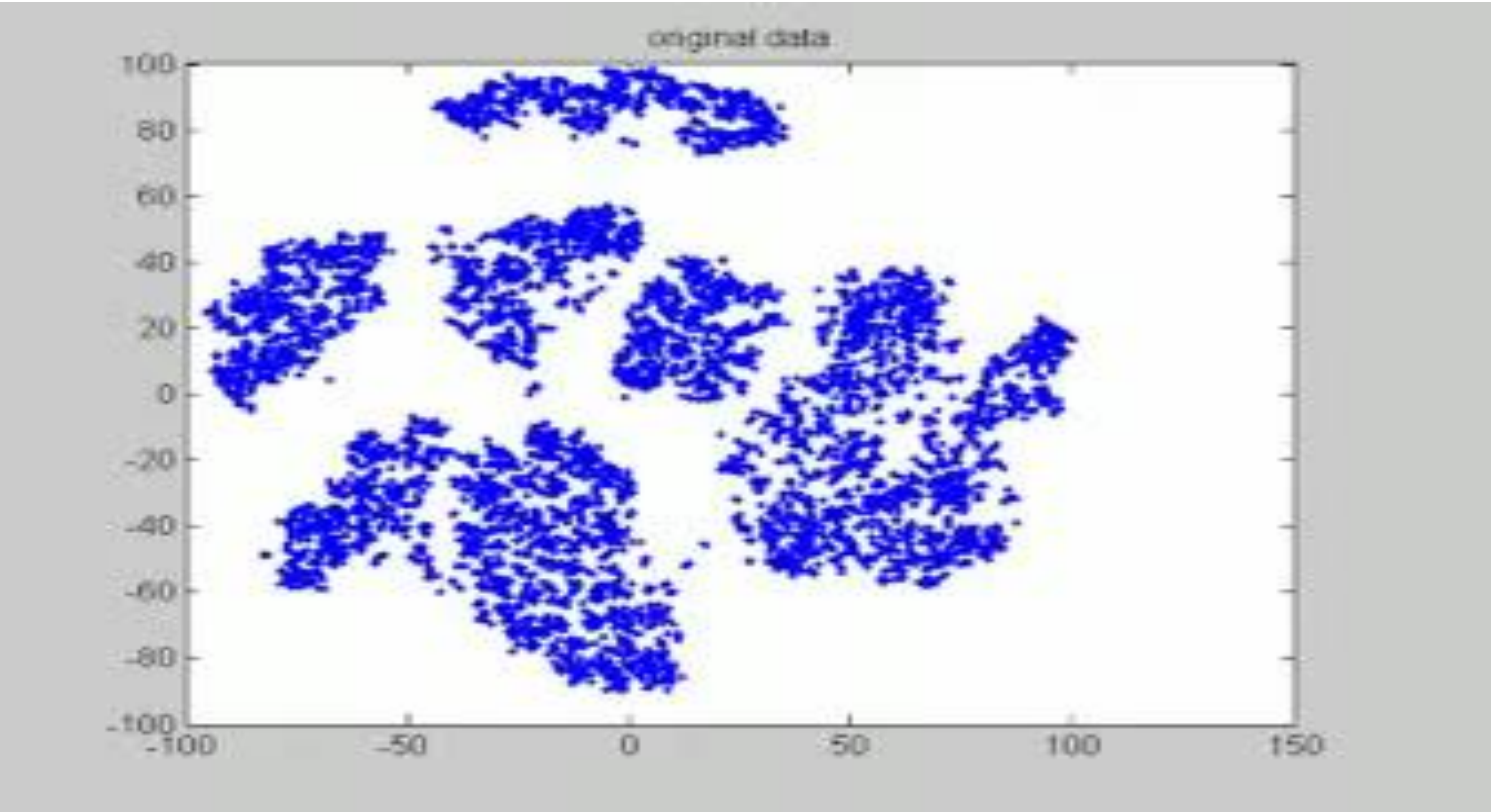
K -Means Clustering – Algorithm

In simple terms

- Initialize k cluster centres
  - Do
    - Assignment step: Assign each data point to its closest cluster center
    - Re-estimation step: Re-compute cluster centers
  - While (there are still changes in the cluster centers)

K Means clustering in action

Dividing the data into 10 clusters using K-Means



LAB: Building Clusters using K-Means

- A Supermarket wanted to send some promotional coupons to 100 families
- The idea is to identify 100 customers with medium income and low recent spends

Code: Building Clusters using K-Means

```
In [134]: sup_market = pd.read_csv("D:\\Google Drive\\Training\\5. Machine Learning Python\\3.Reference\
\15. Cluster Analysis\\DataSets\\Super Market Coupons\\Super_market_Coupons.csv")
     ...: print(sup_market.shape)
     ...: print(sup_market.columns.values)
     ...: print(sup_market.head())
(3000, 6)
['cust_id' 'age' 'Estimated_income' 'recent_spends' 'family_size'
 'Avg_visits_permonth']
   cust_id  age  Estimated_income  recent_spends  family_size  \
0        1   30              3300     771.572261            1
1        2   46             12454     128.922027            3
2        3   76                 0       0.000000            1
3        4   38              3000      76.967031            3
4        5   39              2500    2499.999750            1

   Avg_visits_permonth
0                    4
1                    3
2                    8
3                    3
4                    1
```

Code: Building Clusters using K-Means

```
In [135]: from sklearn.cluster import KMeans
     ...: kmeans = KMeans(n_clusters=5,  random_state=333) # Mention the Number of clusters
     ...: X=sup_market.drop(["cust_id"],axis=1) # Custid is not needed
     ...: kmeans = kmeans.fit(X) #Model building
     ...:
     ...: #The Results
     ...: centers= kmeans.cluster_centers_
     ...: #Format and print
     ...: np.set_printoptions(suppress=True)
     ...: print(np.around(centers))
[[    51.    5624.    1637.       2.       5.]
 [    53.   26894.   10636.       2.       6.]
 [    53.    1054.     312.       1.       6.]
 [    53.   11632.    3205.       2.       5.]
 [    52.  101864.   10441.       4.       8.]]
```

Code: Building Clusters using K-Means

```
In [136]: labels = kmeans.predict(X)
     ...: print(labels)
     ...: sup_market["Cluster_id"]=labels
     ...: sup_market.head()
[2 3 2 ... 2 0 2]
Out[136]:
   cust_id  age  Estimated_income  recent_spends  family_size  \
0        1   30              3300     771.572261            1
1        2   46             12454     128.922027            3
2        3   76                 0       0.000000            1
3        4   38              3000      76.967031            3
4        5   39              2500    2499.999750            1

   Avg_visits_permonth  Cluster_id
0                    4           2
1                    3           3
2                    8           2
3                    3           2
4                    1           2
```

Code: Building Clusters using K-Means

```
In [138]: print(sup_market.groupby(['Cluster_id']).mean())
     ...: print(sup_market.groupby(['Cluster_id']).count())
                cust_id        age  Estimated_income  recent_spends  \
Cluster_id
0           1495.376117  51.418359       5618.512591    1630.466006
1           1801.404762  53.428571      26893.857143   10636.029508
2           1505.197368  52.894737       1051.930099     311.633100
3           1471.984252  52.986220      11616.998031    3212.710263
4           2315.000000  52.000000     101864.333333   10441.193440

            family_size  Avg_visits_permonth
Cluster_id
0              1.822908             5.497157
1              2.380952             5.571429
2              1.427632             5.629934
3              2.139764             5.397638
4              4.000000             8.333333
            cust_id   age  Estimated_income  recent_spends  family_size  \
Cluster_id
0              1231  1231              1231           1231         1231
1                42    42                42             42           42
2              1216  1216              1216           1216         1216
3               508   508               508            508          508
4                 3     3                 3              3            3
```

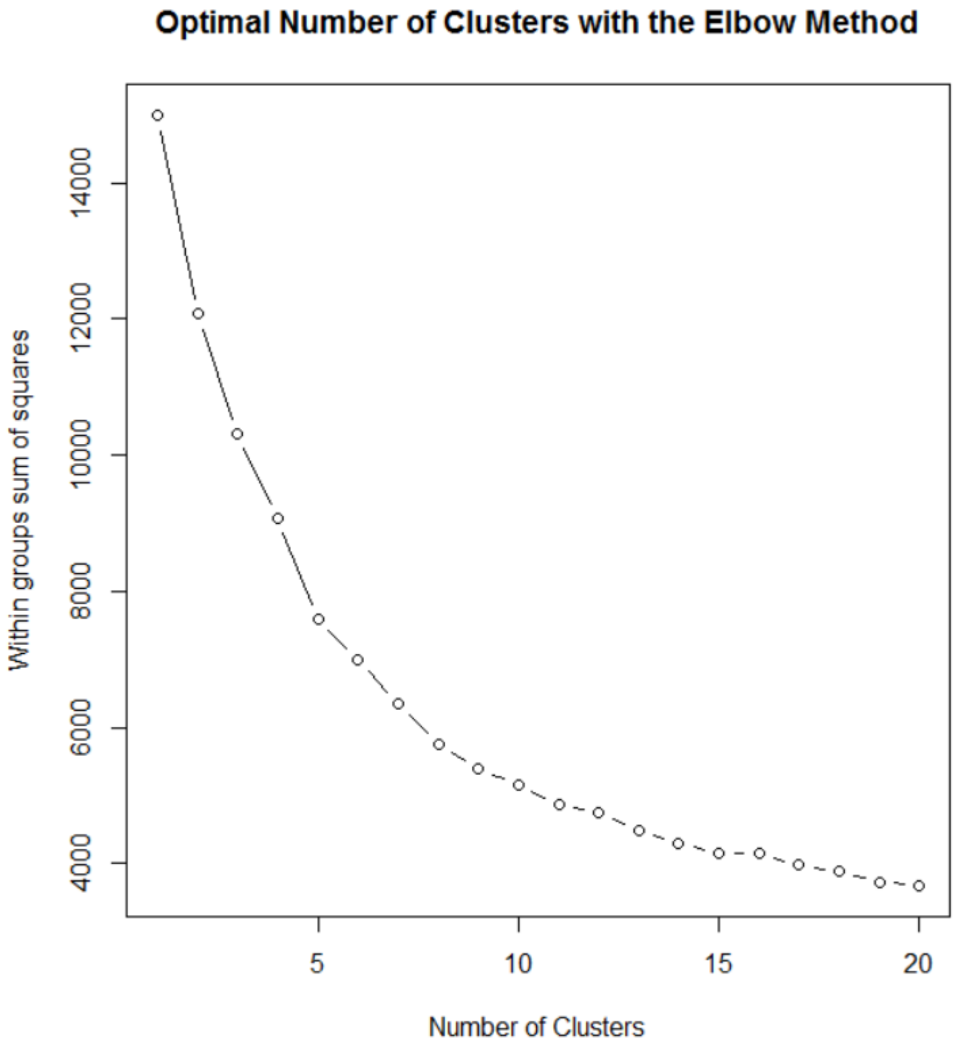
Code: Building Clusters using K-Means

```
In [141]: target_data=sup_market[(sup_market["Cluster_id"]==1) | (sup_market["Cluster_id"]==3)]
     ...: print(target_data.shape)
     ...: target_data.sample(100)
(550, 7)
Out[141]:
      cust_id  age  Estimated_income  recent_spends  family_size  \
2846     2847   62             12450    1192.840078            3
60         61   47             14000    4142.995528            2
125       126   49              8785    6447.183793            3
800       801   39             33333   14121.178320            3
1253     1254   52             14122    4362.042888            1
1855     1856   52             18683    5037.822505            3
801       802   73             10364    3648.240367            1
873       874   51              9497    4116.580997            5
1717     1718   48             34000     482.373538            3
708       709   79             22500    1169.460473            2
1518     1519   70             15000       0.000000            2
```

Choosing Number of Clusters - K

- We choose using elbow method
- In Cluster analysis we try to build clusters in such a way that
  - Within cluster variance is small
  - Between the cluster variance is high
- We draw a graph of within cluster distances (overall sum a within cluster distances) vs number of clusters
- In the graph, if we see there is no significant dip in within cluster variance, we can stop the number of clusters there
- Rebuild the clusters using optimal number of clusters

Choosing Number of Clusters - K



- However, for all practical purposes, we manually choose the K, that suits best for our data
- Most of the times, we decide K based on the business scenario, problem statement, count of items in each cluster etc.,

Conclusion

Conclusion

- K – means is a partitional clustering algorithm.
- K-Means is an unsupervised learning method
- There are other methods too. Some algorithms work well on a certain type of problems.
  - Hierarchical Clustering, Density-based ,Grid-based Clustering,Model-based Clustering, Frequent pattern-based Clustering
- Try multiple times to decide the right K-value
- Clustering is also used in text mining
  - Document clustering
  - News articles clustering

Appendix

Data Standardisation

Standardised Data

Actual Data

Custid	Debt Ratio	Credit Limit
C1	0.4	5000
C2	0.39	5100
C3	0.8	5000

Distance Matrix

```
      1     2     3
1   0.0
2 100.0   0.0
3   0.4 100.0   0.0
```

$$\textit{Standardised value} = \frac{x - mean(x)}{sd(x)}$$

Standardised data

```
> Cust_data_sd
  Cust_id Debt_Ratio1 credit_limit1
1      C1  -0.5558399    -0.5773503
2      C2  -0.5985968     1.1547005
3      C3   1.1544366    -0.5773503
```

Distance Matrix

```
    1   2   3
1 0.0
2 1.7 0.0
3 1.7 2.5 0.0
```

Non- Numerical Data

Distance Measure for Non- Numeric data

- Distance measure for Binary Variables/Flag Variable/Indicator variable / Boolean Variable

Point X_j

Point X_i

	1	0	
1	A	B	A+B
0	C	D	C+D
	A+C	B+D	A+B+C+D

$$d = \frac{B+C}{A+B+C+D}$$

Distance Measure For binary Variables

Customer ID	House Loan	Existing Customer	Gender	Marital Status	Premier Customer
C001	Yes	Yes	M	No	No
C002	Yes	No	M	Yes	No

C002

		1-Yes	0-No	
C001	1-Yes	2	1	
	0-No	1	1	
				5

$$d = \frac{B+C}{A+B+C+D}$$

Distance Measure For binary Variables

Customer ID	House Loan	Existing Customer	Gender	Marital Status	Premier Customer
C001	Yes	Yes	M	No	No
C002	Yes	No	M	Yes	No

		C002		
		1-Yes	0-No	
C001	1-Yes	2	1	
	0-No	1	1	
				5

$$d = \frac{B+C}{A+B+C+D}$$

Distance (Dis-similarity) =2/5

Distance Measure for Categorical Variables

- Categorical variables are a generalization of the binary variables that can take more than two values
- We can create multiple binary variables(dummy variables) from one categorical variable. If there are ten classes in a categorical variable then we can create ten dummy variables (Nine are sufficient)

Region
East
West
North
South
West

East	West	North	South
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
0	1	0	0

Distance Measure for Categorical Variables

- Categorical values have lot of classes we can simply calculate the distance by considering Matching vs Non-Matching Cases
- K - Number of variables
- S - Number matching Cases

$$d = \frac{N - S}{N}$$

Distance Measure for Categorical Variables

Customer ID	Region	Card Type	Status Code	Marital Status	Account type
1	EAST	C	A	No	Premier
2	NORTH	B	D	Yes	Premier
3	NORTH	B	H	Yes	Basic

d(1,2)= (5-1)/5 = 4/5
d(1,3)= (5-0)/5 = 5/5
d(2,3)= (5-3)/5 = 2/5

Centroid for Non-Numerical data

- Cluster mean is not possible for categorical data
- We can use two metrics as central tendencies
- Mode
  - Most occurring class is one more measure of central tendency like mean
- Medoids
  - Medoids are similar in concept to means or centroids, but medoids are always members of the data set. Medoids are most commonly used on data when a mean or centroid cannot be defined
  - Medoid one chosen, centrally located object in the cluster.
  - Most centrally located observation in a cluster.

K-Means for Non-Numerical Data: K-modes

- Follow the same algorithm but consider below options
  - Choose a distance matrix that can handle categorical values
  - Choose a centroid that can handle categorical values

Advantages

- Very less **computation time**. This is a huge advantage if you are dealing with large datasets.
- Scaling up is easy and interpretation is simple
- Easy to understand and interpret

Disadvantages of K-Means

- We need to choose the **number of clusters k**, in advance. At times choosing K is not an easy job
- Effective for **numerical data.** Calculating centroid and Euclidian distance requires all the values to be numerical
- Not suitable for data with **outliers and noise.** This type of input data results into clusters with non-homogenous cases in one cluster.
  - Either clean the data for outliers before applying algorithm

Thank you