



Clustering

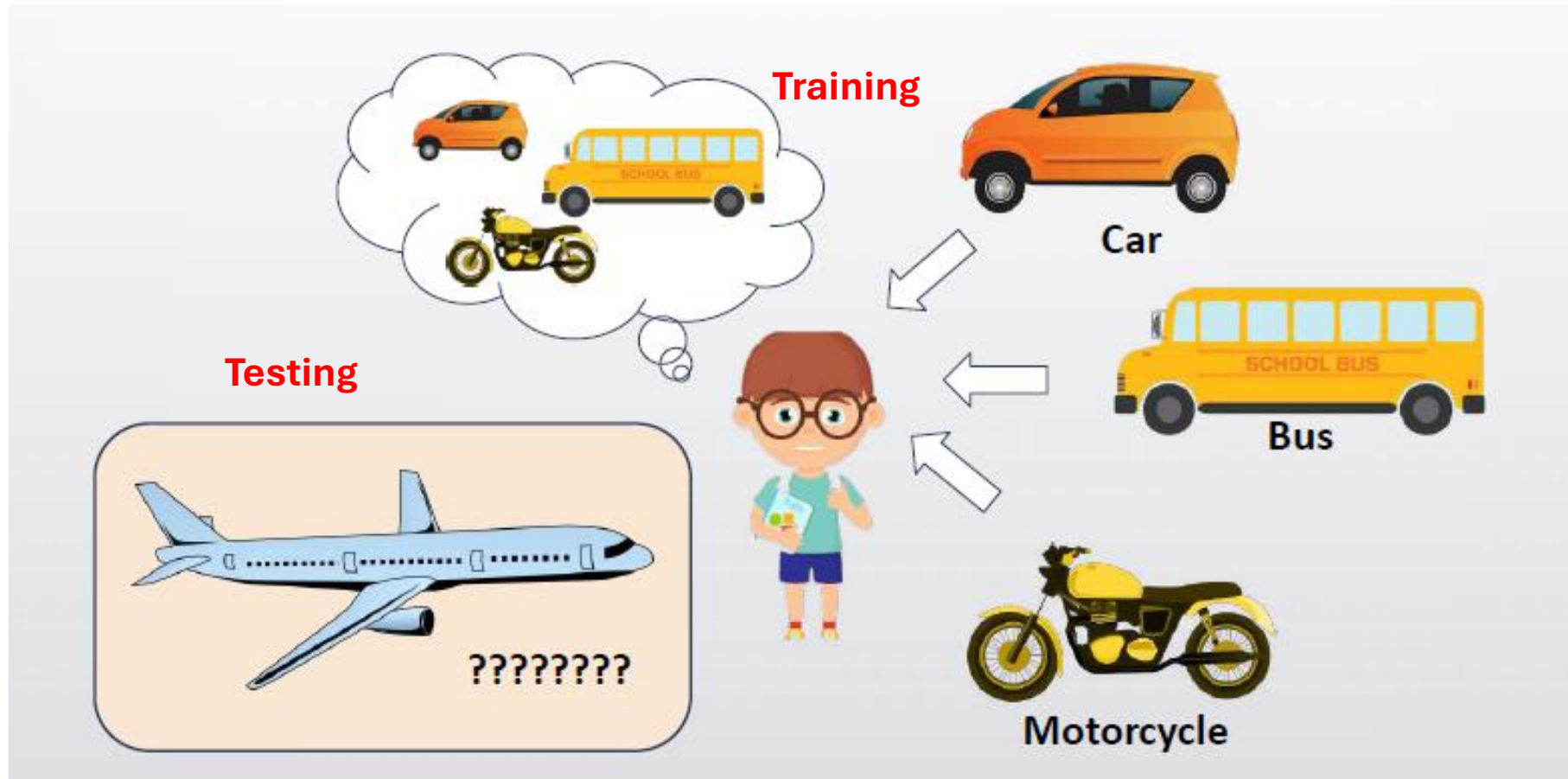
Koko Friansa

15-11-2024

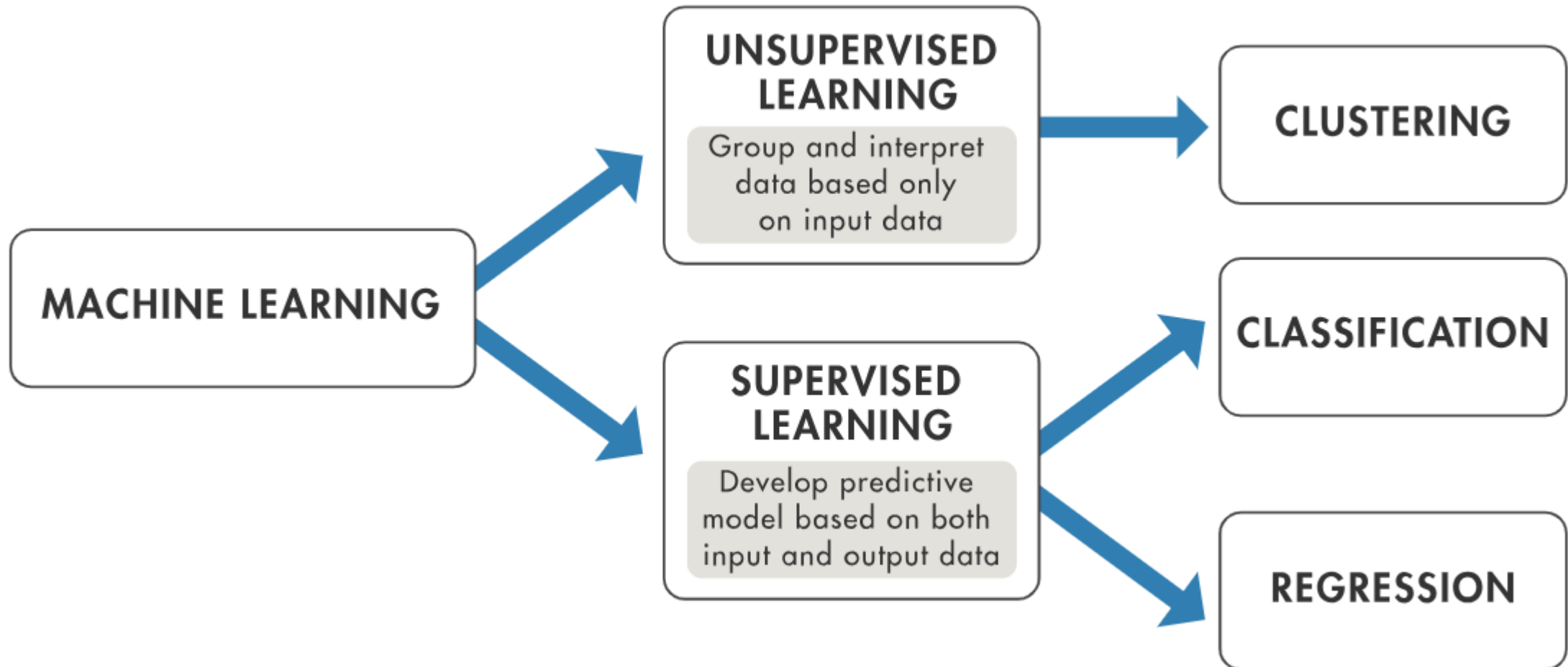
(Sekolah Bisnis Manajemen ITB)



Introduction: Machine Learning

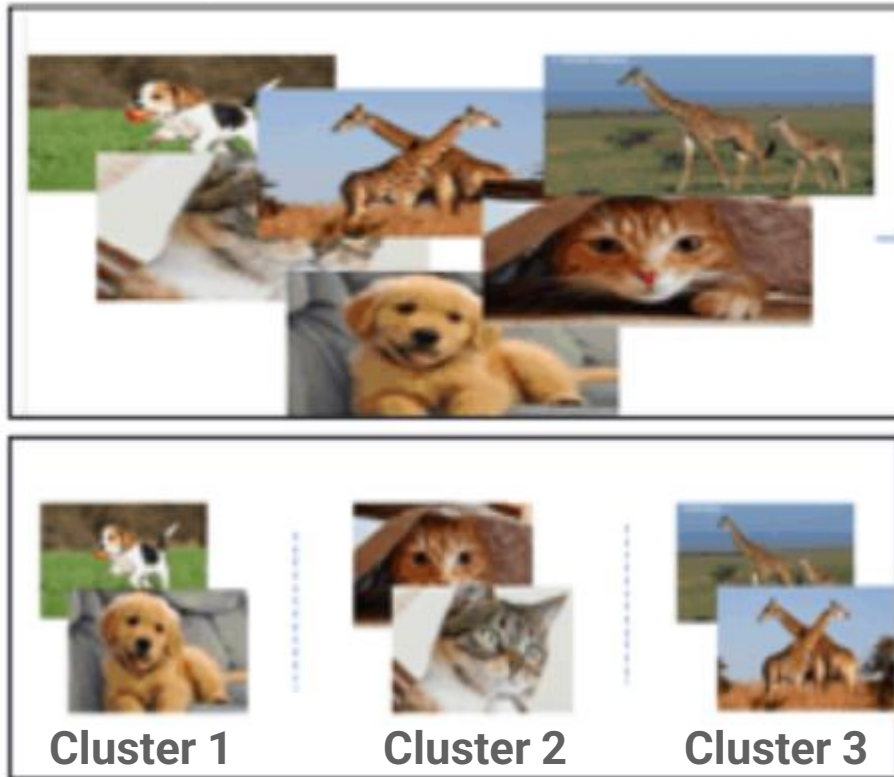


Machine Learning Techniques

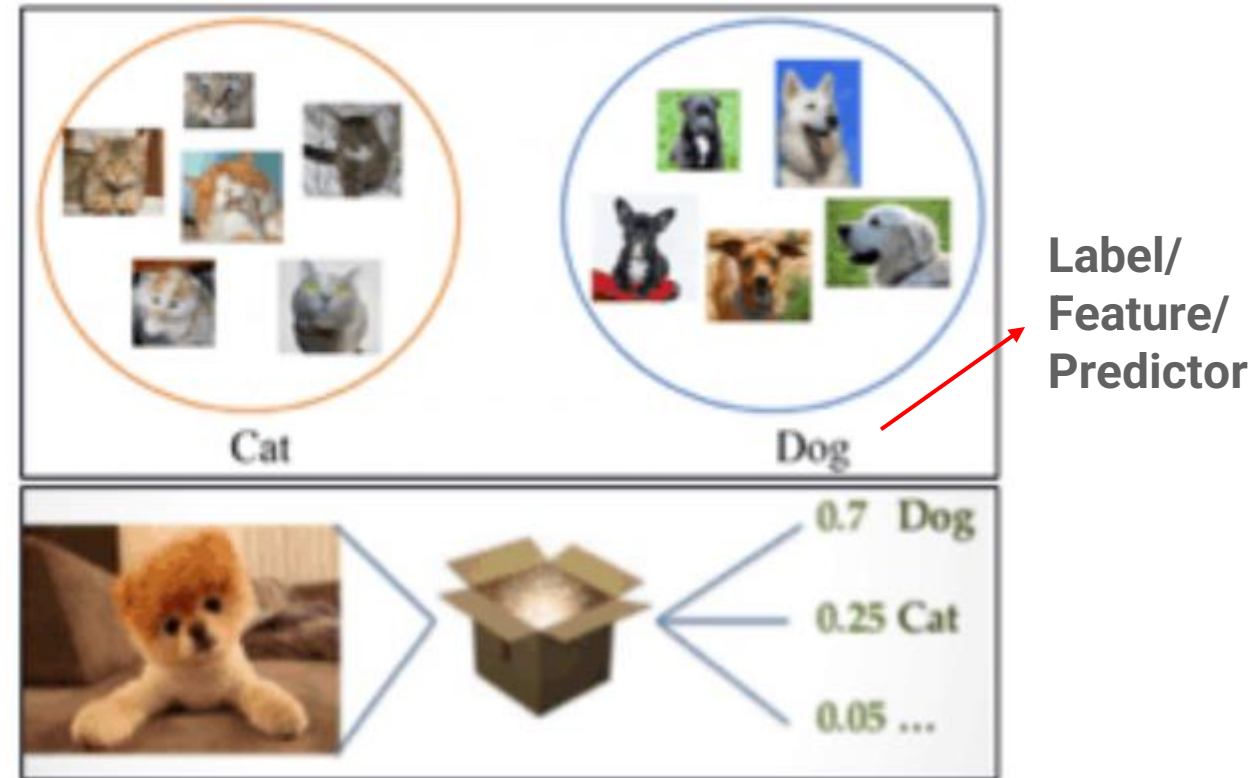


Clustering vs Classification

Clustering

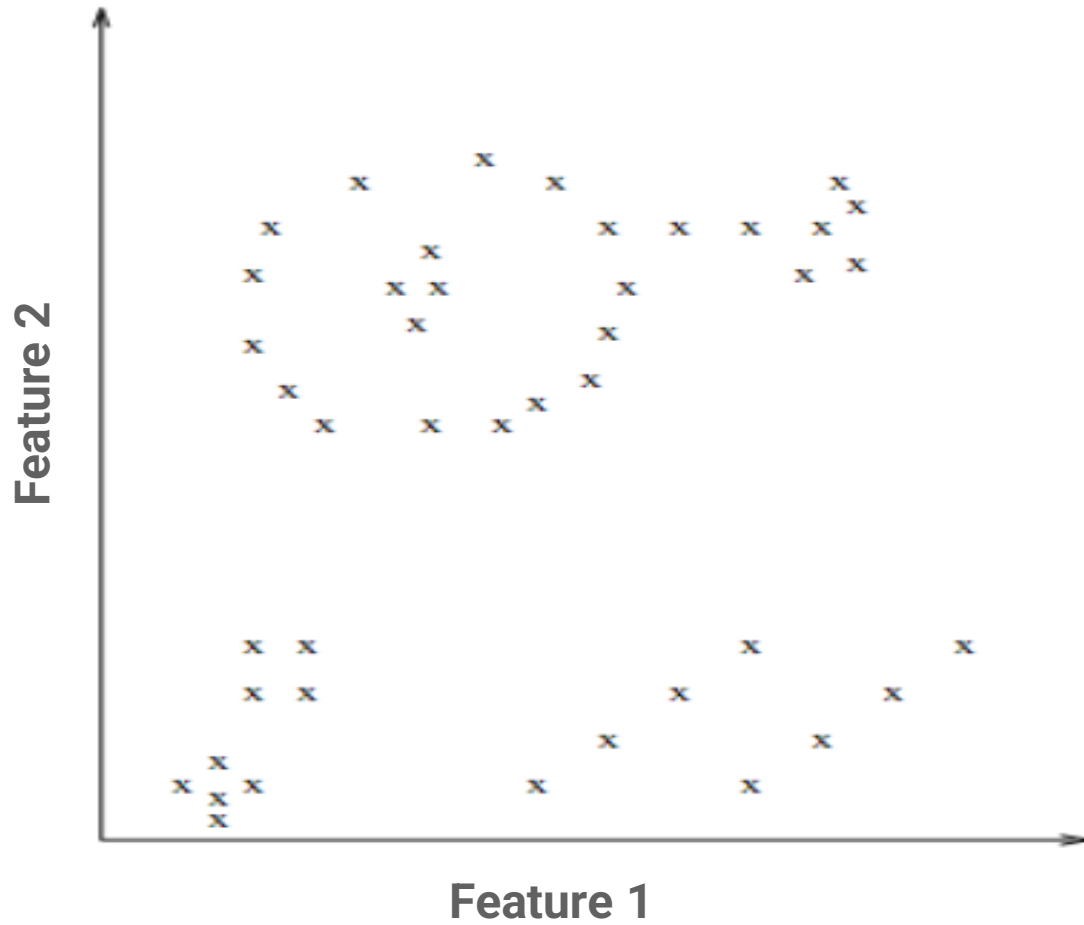


Classification

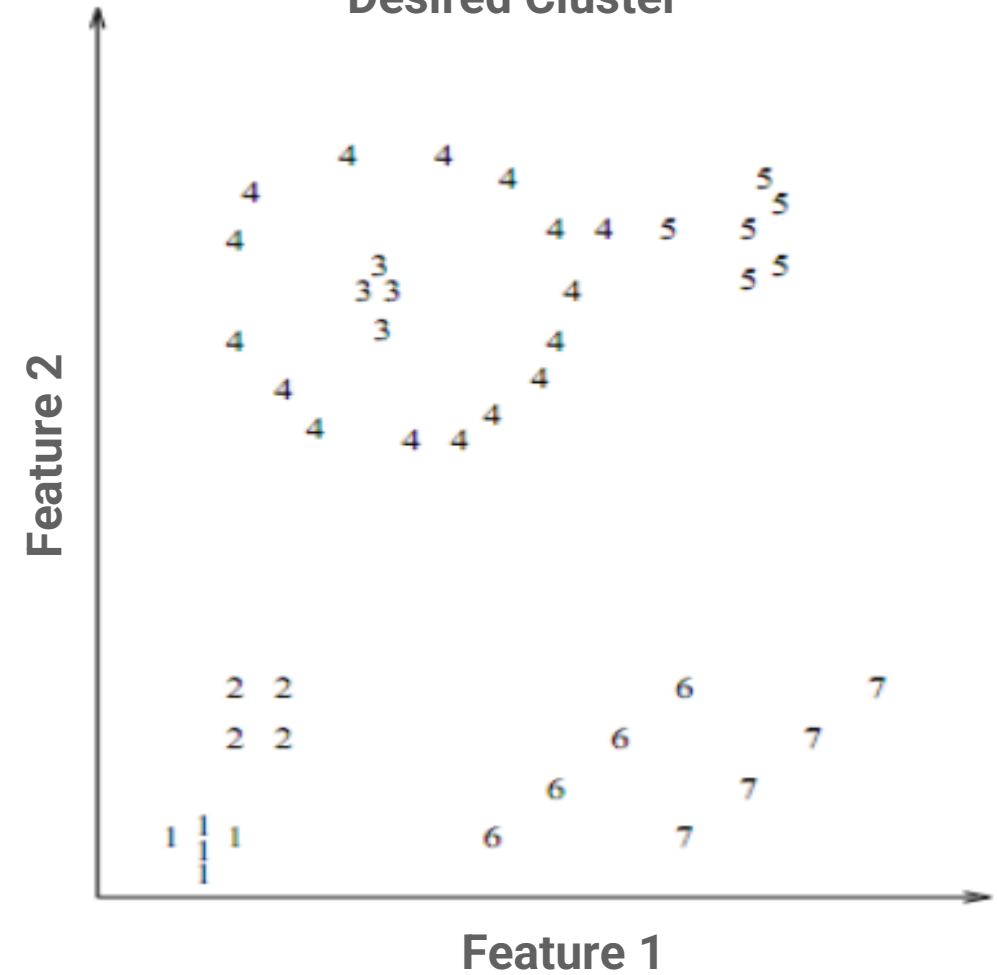


Clustering

Input Data



Desired Cluster



Clustering: Finding Natural Groups

High intra-cluster similarity

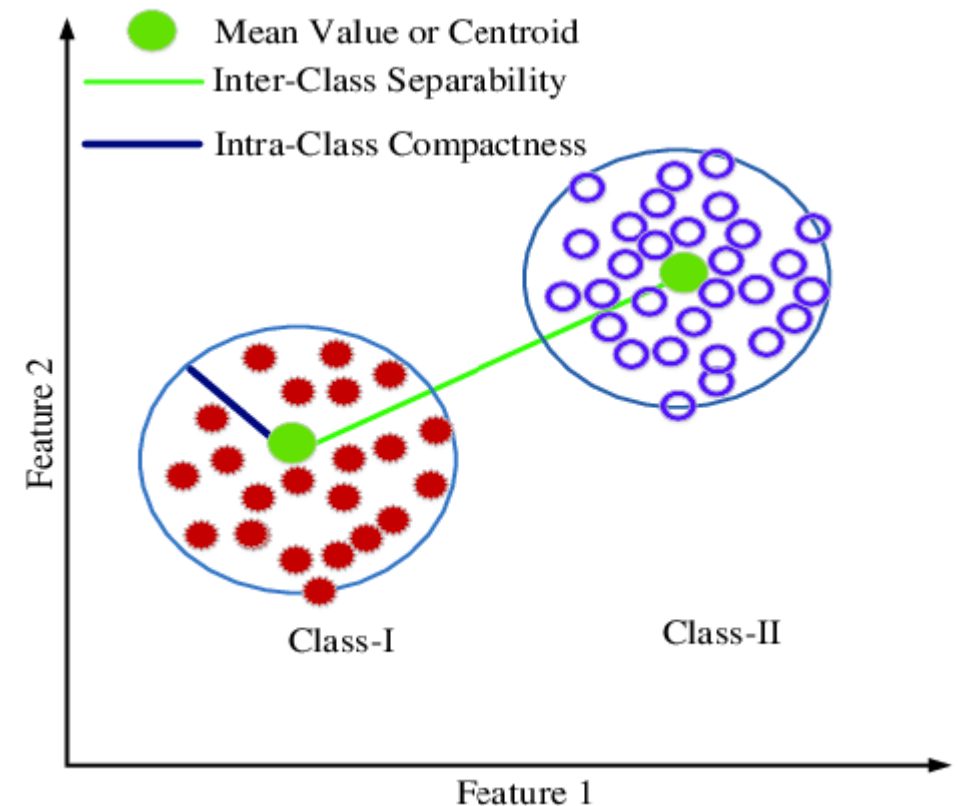
Data within the same cluster should be as similar as possible.

Low inter-cluster similarity

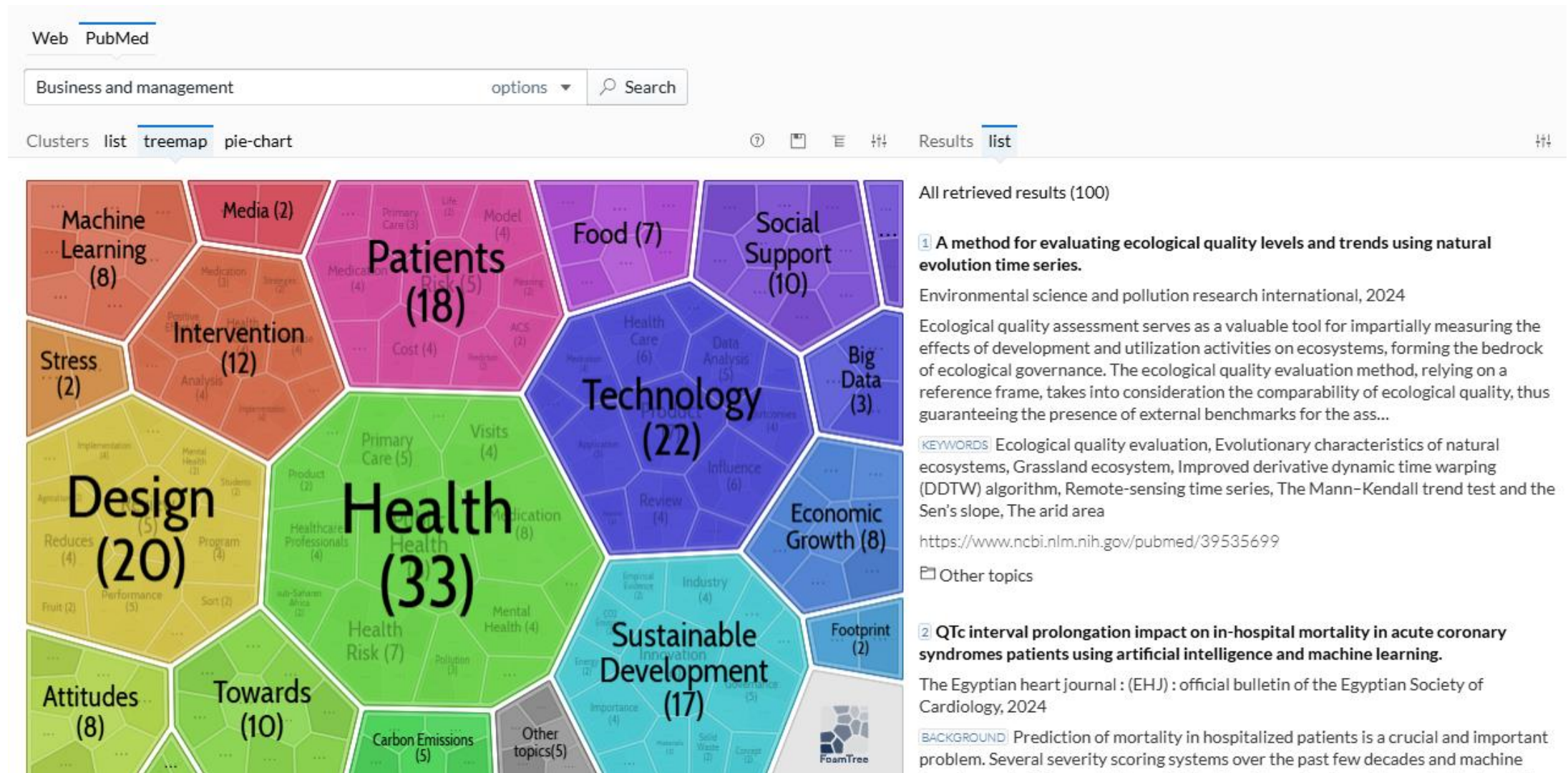
Data in different clusters should be as far apart as possible.
Clear and practical measurement of similarity and distance

Well defined

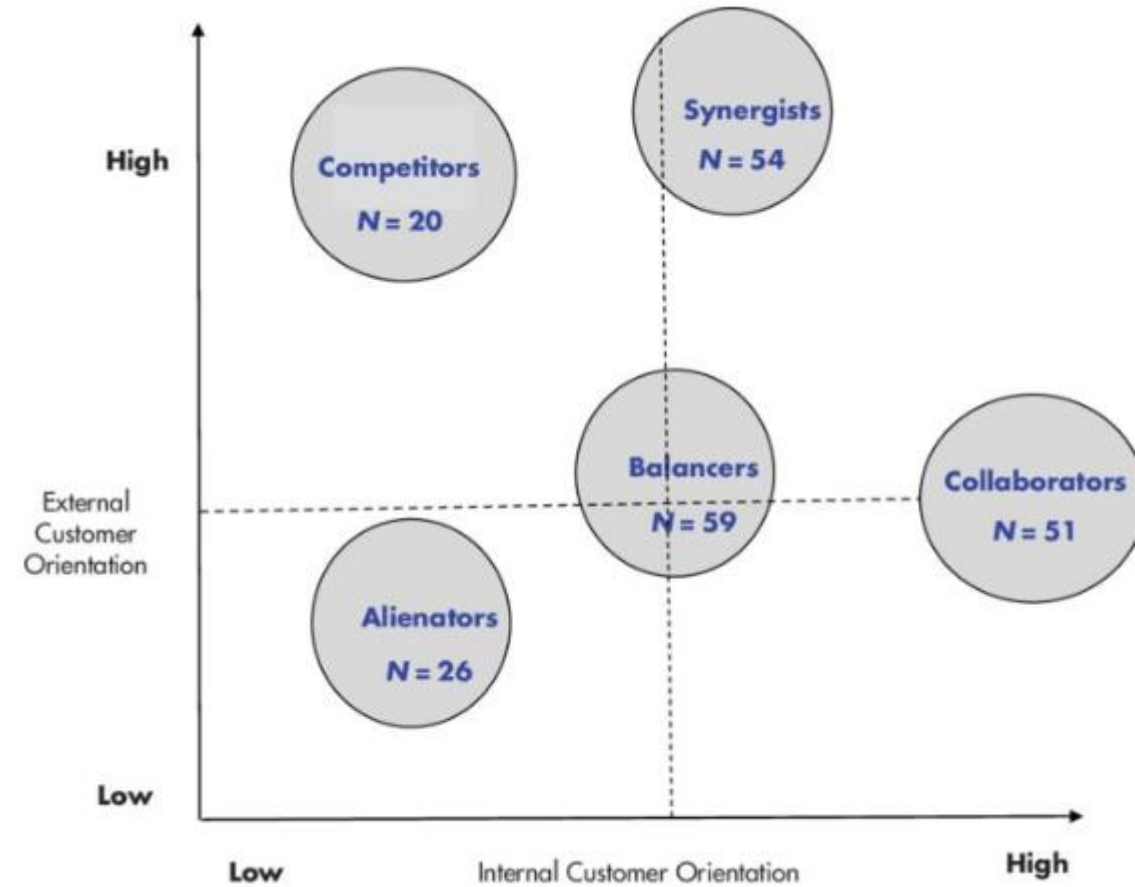
The measurement of similarity and distance should be well-defined and have practical semantics relevant to the domain.




Clustering in Search Engine




Clustering in Employment




Clustering in GoFood



Before relevance ranking & personalisation
(For both)



After relevance ranking & personalisation
(For Mila)




After relevance ranking & personalisation
(For Husain)


← Near me

Q What would you like to eat?

PROMO




Muaithaee Ayam
Chicken & Duck • \$\$\$\$
★ 4.1 • 1.1 km (10-20 min)




Nasi Goreng Spot
Western, Fast Food • \$\$\$\$
★ 3.9 • 1.3 km (10-20 min)

PROMO



Warung Dim Sum
Japanese • \$\$\$\$
★ 3.1 • 1.7 km (10-20 min)




Miemie King
Noodles, Seafood • \$\$\$\$
★ 3.4 • 2.1 km (10-20 min)

← Near me


Q What would you like to eat?

PROMO




Burger Palace
Western, Fast Food • \$\$\$\$
★ 4.5 • 2.9 km (30-40 min)


PROMO



Muaithaee Ayam
Chicken & Duck • \$\$\$\$
★ 4.1 • 1.1 km (10-20 min)




Sate Ayam Joint
Western, Fast Food • \$\$\$\$
★ 4.2 • 2.3 km (30-40 min)



Nasi Goreng Spot
Western, Fast Food • \$\$\$\$
★ 3.9 • 1.3 km (10-20 min)


← Near me

Q What would you like to eat?




Miemie King
Noodles, Seafood • \$\$\$\$
★ 3.4 • 2.1 km (10-20 min)

PROMO




Muaithaee Ayam
Chicken & Duck • \$\$\$\$
★ 4.1 • 1.1 km (10-20 min)

PROMO

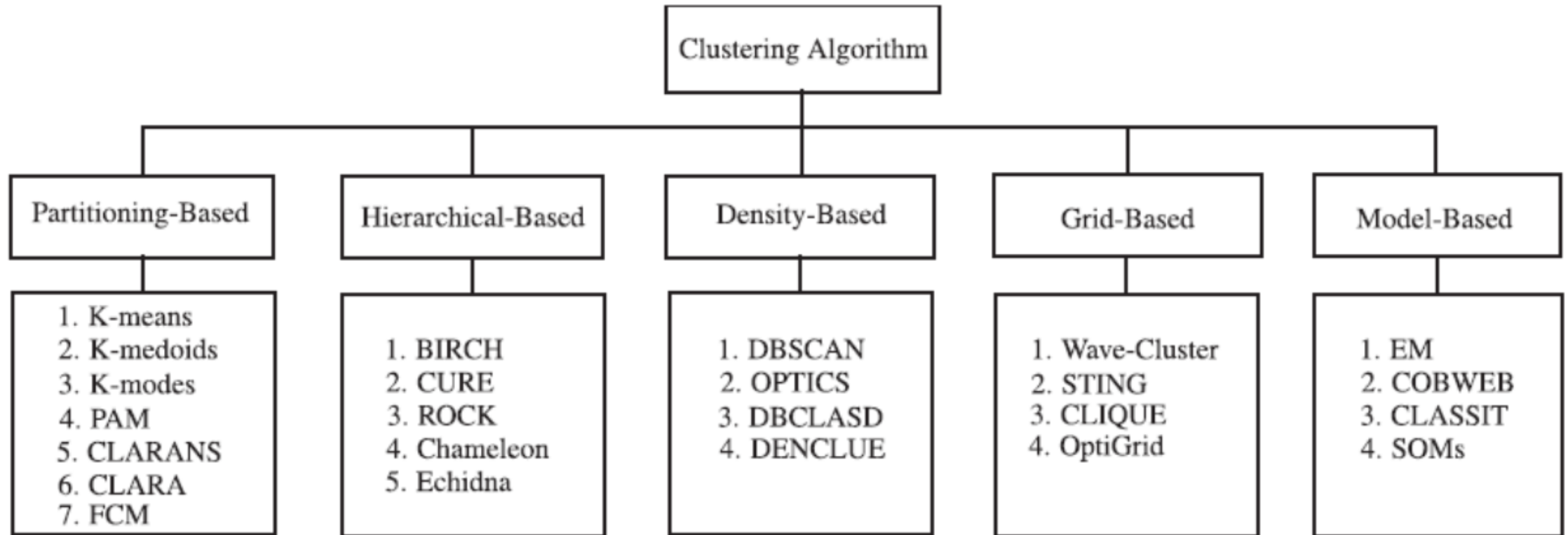


Blok M Mie Goreng
Noodles, Chicken & Duck • \$\$\$\$
★ 2.9 • 3.1 km (30-40 min)



Nasi Goreng Spot
Western, Fast Food • \$\$\$\$
★ 3.9 • 1.3 km (10-20 min)

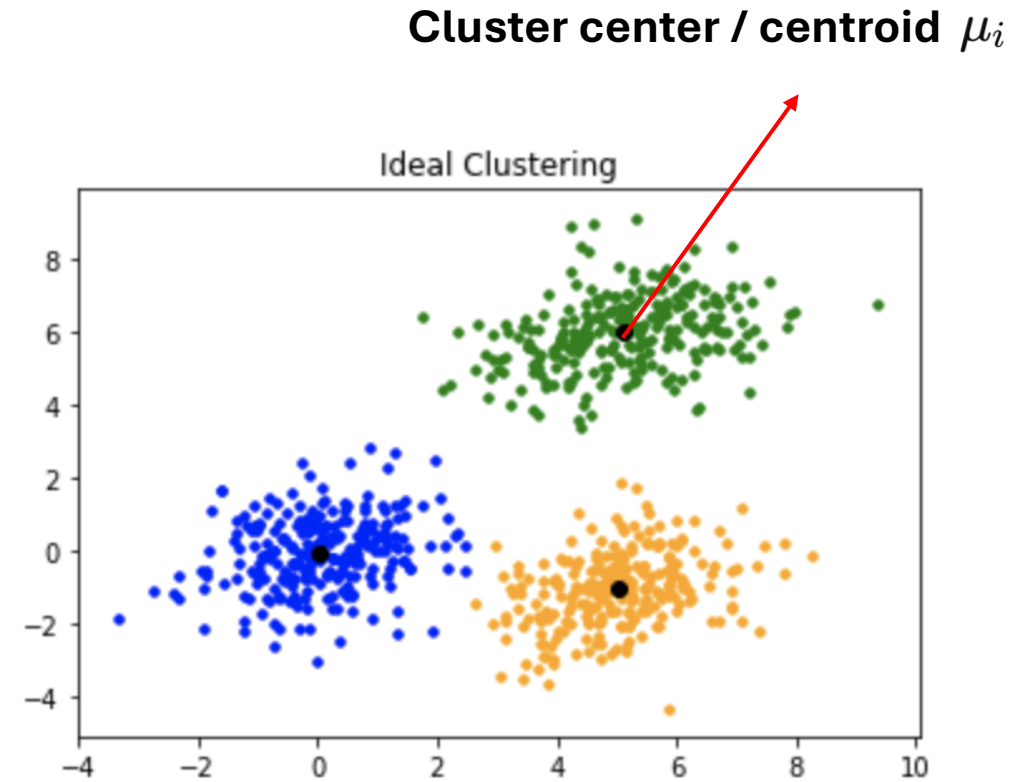
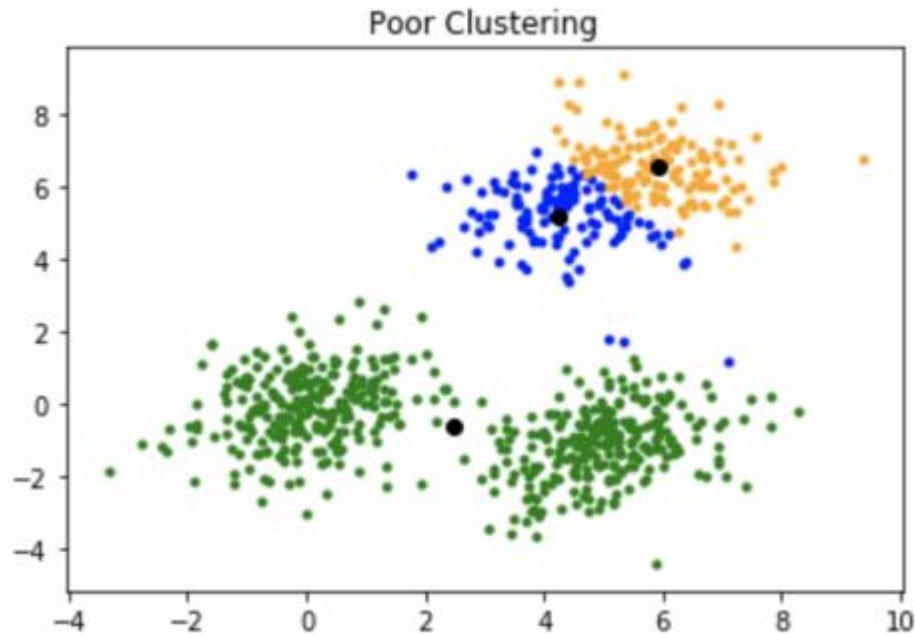
Clustering Method



K-Means Clustering



K-Means Clustering



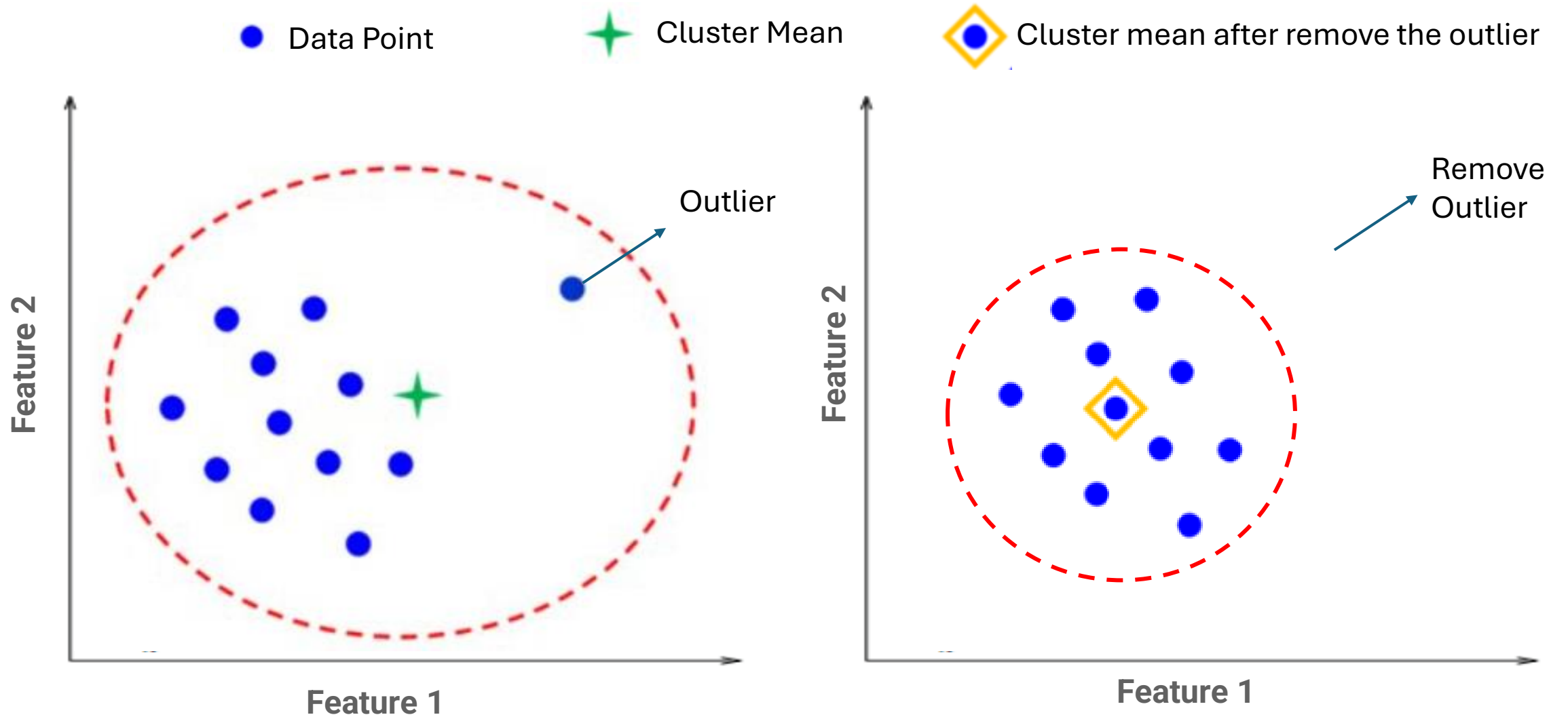
The K-means algorithm minimizes the **within-cluster sum of squares (WCSS)**, also called inertia:

$$\text{WCSS} = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- k : Number of clusters.
- C_i : The set of points in cluster i .
- μ_i : The centroid of cluster i .
- $\|x - \mu_i\|^2$: The squared distance between a data point x , and its cluster centroid μ_i .

Outlier in K-Means



Advantages of K-Means

1. Simplicity and Ease of Implementation:

K-Means is straightforward to **understand** and **implement**.

2. Efficiency:

K-Means is computationally efficient, especially for **large datasets**.

3. Interpretable Results:

The algorithm produces cluster centroids and assignments, which are **easy to interpret**.

4. Applicable Across Domains:

Widely used in customer segmentation, image compression, anomaly detection, and more.

Disadvantages of K-Means

1. Predefined Number of Clusters (K)

2. Assumption of Spherical Clusters

It struggles with datasets where clusters are elongated, unevenly sized, or not well-separated.

3. Not Robust to Outliers

Outliers can pull centroids toward them, distorting the clusters.
Preprocessing (e.g., outlier removal) may be needed.

4. Difficulty Handling Non-Linear Boundaries

K-Means assumes linear separability of clusters, making it less effective for complex datasets.

5. Scalability to High Dimensions

In high-dimensional spaces, the concept of "distance" becomes less meaningful.
Dimensionality reduction (e.g., PCA) may be needed before applying K-Means.

K-Means Clustering in R

```
1 #2 Features Clustering----  
2 # Load the dataset  
3 library(ISLR)  
4 data(Credit)  
5 View(Credit)  
6
```

Load a Dataset

K-means Features.R* x Credit x													
Filter													
	ID	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance	
1	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333	
2	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903	
3	3	104.593	7075	514	4	71	11	Male	No	No	Asian	580	
4	4	148.924	9504	681	3	36	11	Female	No	No	Asian	964	
5	5	55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331	
6	6	80.180	8047	569	4	77	10	Male	No	No	Caucasian	1151	
7	7	20.996	3388	259	2	37	12	Female	No	No	African American	203	
8	8	71.408	7114	512	2	87	9	Male	No	No	Asian	872	
9	9	15.125	3300	266	5	66	13	Female	No	No	Caucasian	279	
10	10	71.061	6819	491	3	41	19	Female	Yes	Yes	African American	1350	
11	11	63.095	8117	589	4	30	14	Male	No	Yes	Caucasian	1407	
12	12	15.045	1311	138	3	64	16	Male	No	No	Caucasian	0	
13	13	80.616	5308	394	1	57	7	Female	No	Yes	Asian	204	
14	14	43.682	6922	511	1	49	9	Male	No	Yes	Caucasian	1081	
15	15	19.144	3291	269	2	75	13	Female	No	No	African American	148	
16	16	20.089	2525	200	3	57	15	Female	No	Yes	African American	0	

Dataset preview before clustering

2 Features Clustering

Feature Selection

```
1 #2 Features Clustering----
2 # Load the dataset
3 library(ISLR)
4 data(Credit)
5
6 # Select two features
7 selected_features <- Credit[, c("Age", "Rating")]
8 scaled_selected <- scale(selected_features)
9
10 # K-means clustering
11 set.seed(123)
12 kmeans_result <- kmeans(scaled_selected, centers = 2, nstart = 25)
13
14 # Add cluster assignments to the data
15 Credit$Cluster <- kmeans_result$cluster
16 View(Credit)
17
```

Adding cluster column to the table.

•K-Means clustering with 2 cluster

•Recommended:

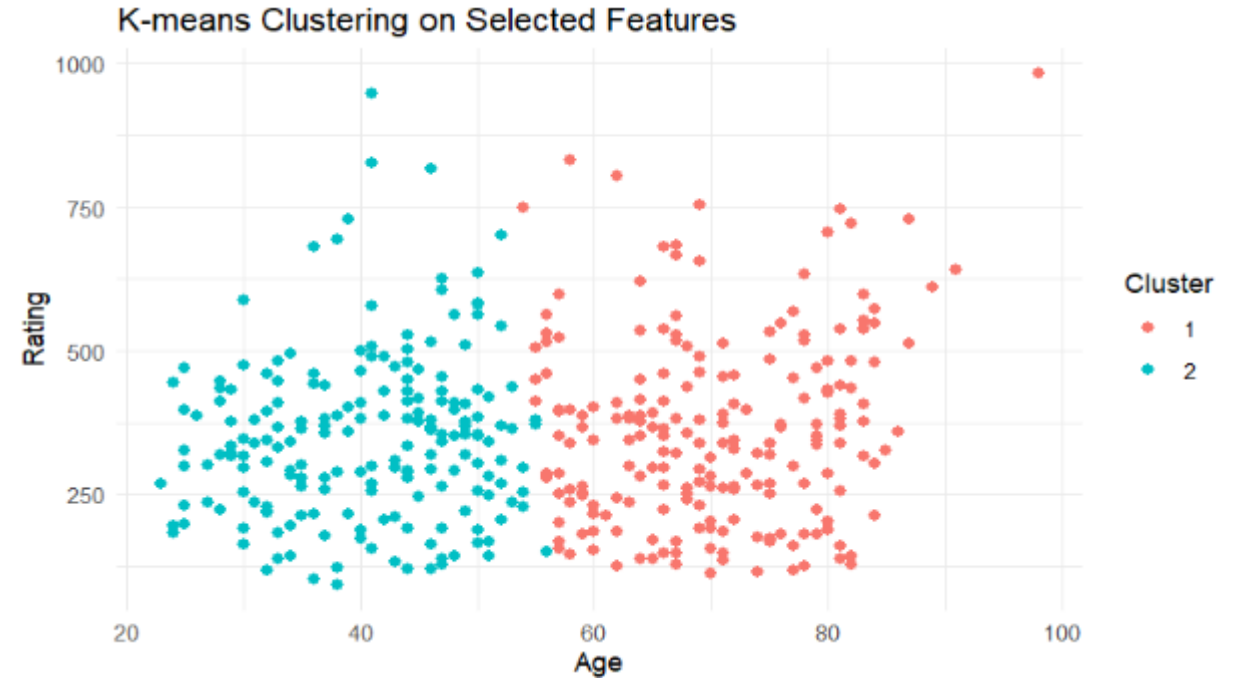
- Use at least **10-25** for small datasets.
- Use higher values (e.g., **50-100**) for larger datasets or when results vary significantly across runs.

	ID	Income	Limit	Rating	Cards	Age	Education	Gender	Student	Married	Ethnicity	Balance	Cluster
1	1	14.891	3606	283	2	34	11	Male	No	Yes	Caucasian	333	2
2	2	106.025	6645	483	3	82	15	Female	Yes	Yes	Asian	903	1
3	3	104.593	7075	514	4	71	11	Male	No	No	Asian	580	1
4	4	148.924	9504	681	3	36	11	Female	No	No	Asian	964	2
5	5	55.882	4897	357	2	68	16	Male	No	Yes	Caucasian	331	1
6	6	80.180	8047	569	4	77	10	Male	No	No	Caucasian	1151	1
7	7	20.996	3388	259	2	37	12	Female	No	No	African American	203	2
8	8	71.408	7114	512	2	87	9	Male	No	No	Asian	872	1
9	9	15.125	3300	266	5	66	13	Female	No	No	Caucasian	279	1
10	10	71.061	6819	491	3	41	19	Female	Yes	Yes	African American	1350	2
11	11	63.095	8117	589	4	30	14	Male	No	Yes	Caucasian	1407	2
12	12	15.045	1311	138	3	64	16	Male	No	No	Caucasian	0	1
13	13	80.616	5308	394	1	57	7	Female	No	Yes	Asian	204	1
14	14	43.682	6922	511	1	49	9	Male	No	Yes	Caucasian	1081	2
15	15	19.144	3291	269	2	75	13	Female	No	No	African American	148	1
16	16	20.089	2525	200	3	57	15	Female	No	Yes	African American	0	1

The cluster column has been successfully added to the table.

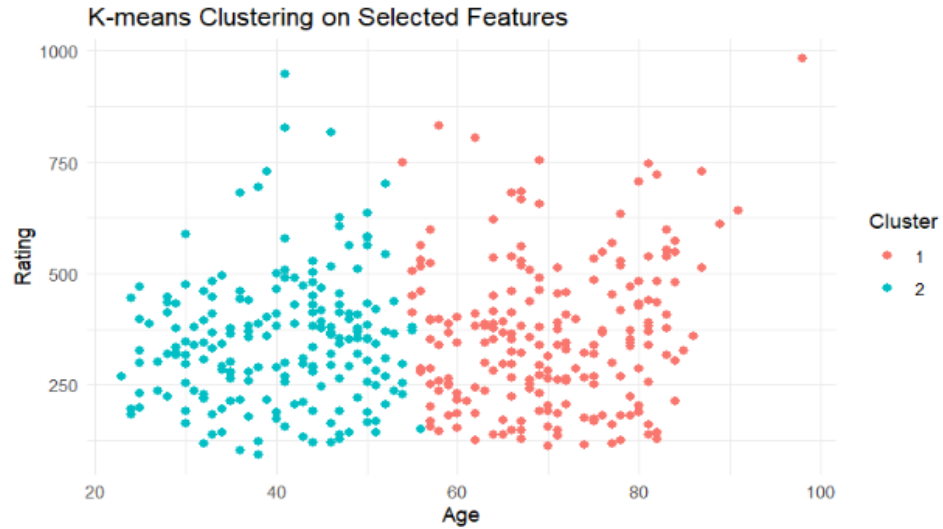
2D Scatter Plot

```
18 # 2D Scatter plot of clusters
19 library(ggplot2)
20 ggplot(Credit, aes(x = Age, y = Rating, color = factor(Cluster))) +
21   geom_point(size = 2) +
22   labs(title = "K-means Clustering on Selected Features",
23        x = "Age",
24        y = "Rating",
25        color = "Cluster") +
26   theme_minimal()
27
```

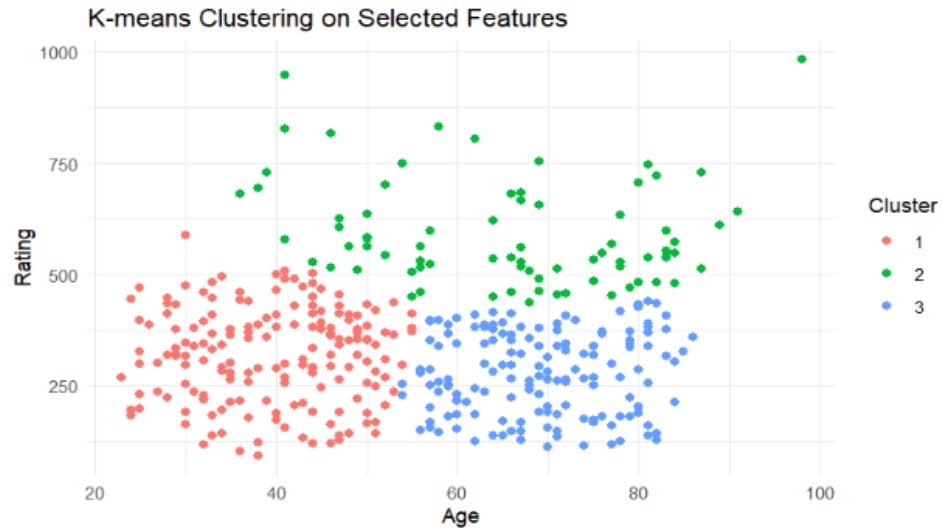


How many clusters are optimal?

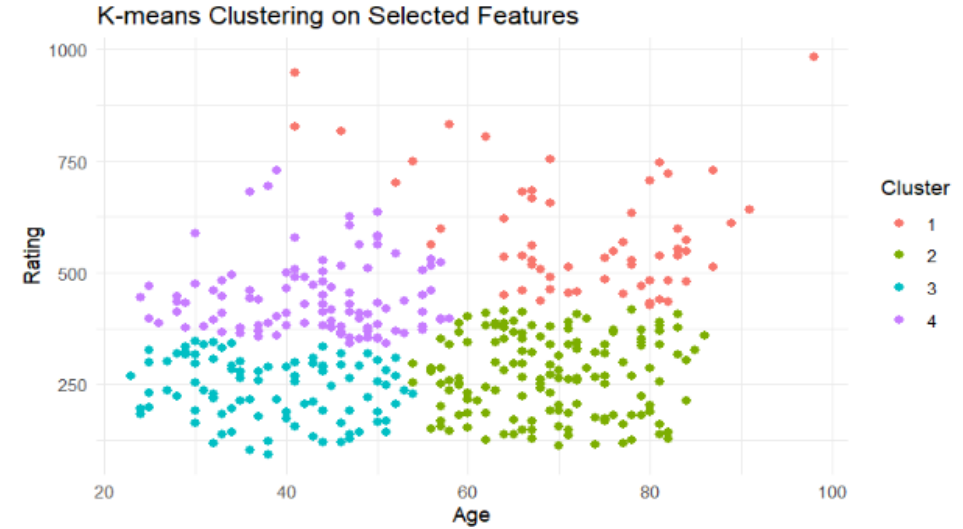
2 clusters



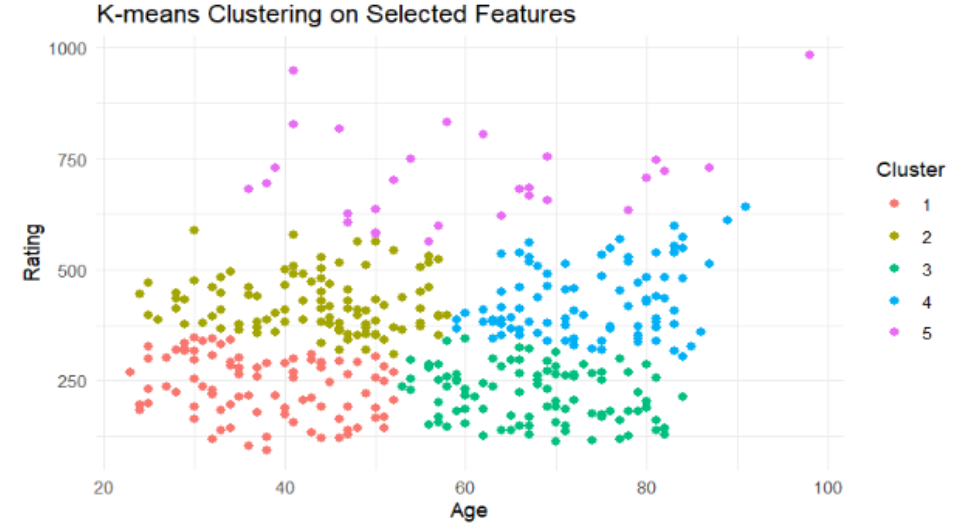
3 clusters



4 clusters



5 clusters



How to find optimal K?

- Elbow Method
- Silhouette Method
- Gap Statistic
- Davies-Bouldin Index

Elbow Method

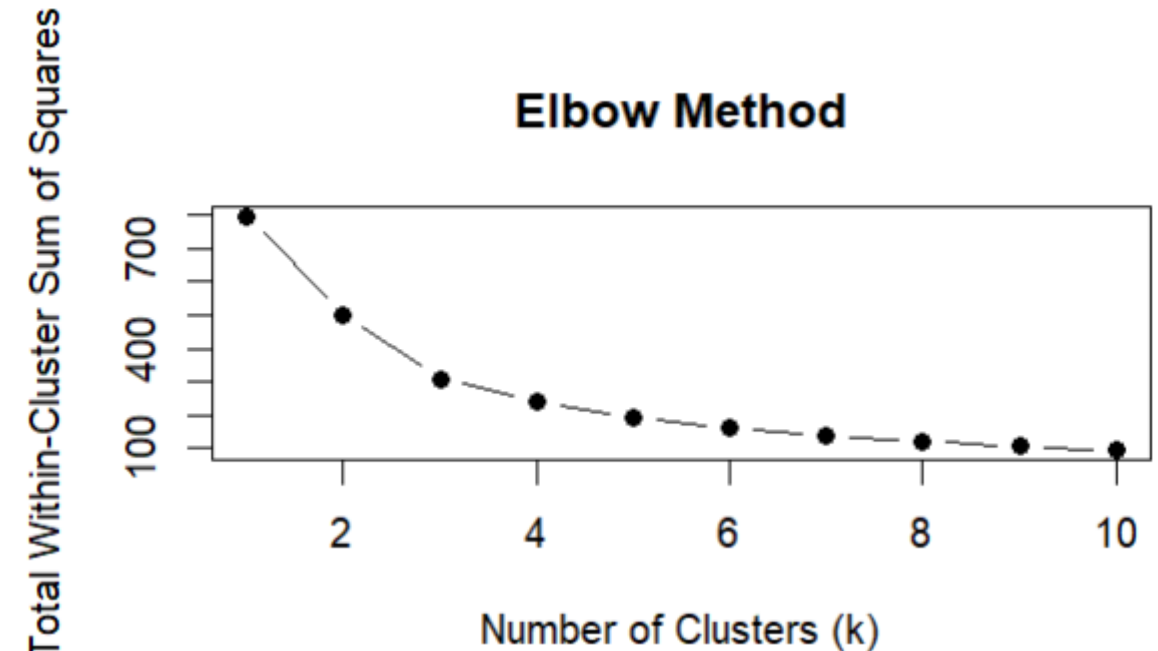
```
29 #Find optimum K - Elbow Method ----
30 #Choose the low point
31 wcss <- sapply(1:10, function(k) {
32   kmeans(scaled_selected, centers = k, nstart = 25)$tot.withinss
33 })
34
35 plot(1:10, wcss, type = "b", pch = 19,
36      xlab = "Number of Clusters (k)",
37      ylab = "Total Within-Cluster Sum of Squares",
38      main = "Elbow Method")
39
```

Best Point: The "elbow point," where the reduction in Within-Cluster Sum of Squares (WCSS) slows significantly (looks like an elbow).

Interpretation:

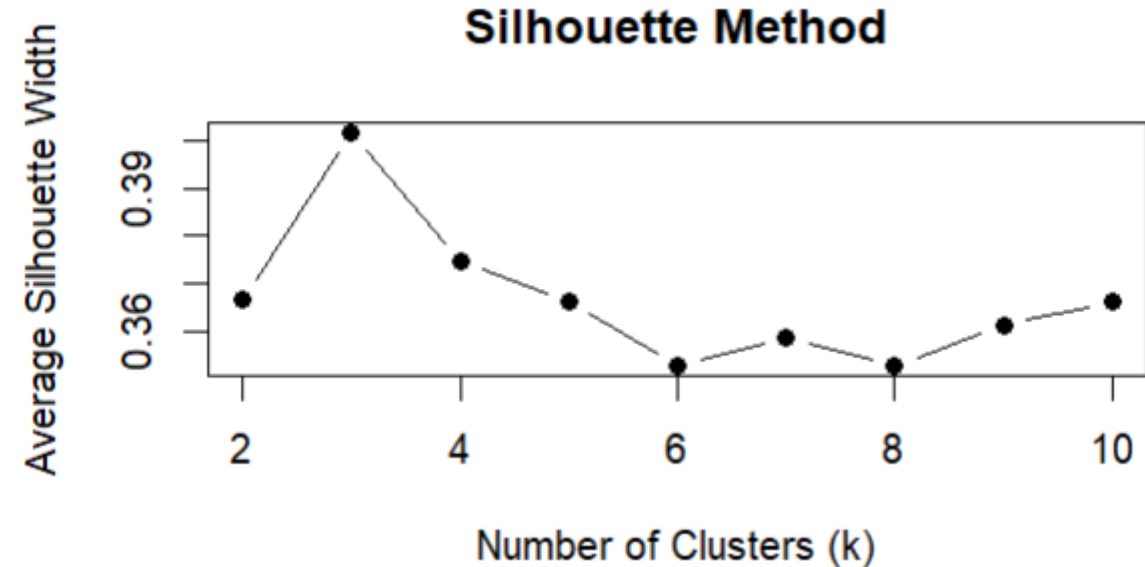
High points represent poor clustering (too few clusters).

Low points beyond the elbow indicate diminishing returns from adding more clusters.



Silhouette Method

```
#Find Optimum K - Silhouette Method ----  
#Choose the high point  
library(cluster)  
library(dplyr) # or library(magrittr)  
  
sil_width <- sapply(2:10, function(k) {  
  kmeans_result <- kmeans(scaled_selected, centers = k, nstart = 25)  
  silhouette(kmeans_result$cluster, dist(scaled_selected))[, 3] %>% mean()  
})  
  
plot(2:10, sil_width, type = "b", pch = 19,  
     xlab = "Number of Clusters (k)",  
     ylab = "Average Silhouette width",  
     main = "Silhouette Method")
```



Graph: Plots the average silhouette width (ranges from -1 to 1) for each k.

Best Point: The highest point in the graph indicates the optimal k.

Interpretation:

A high silhouette score means points are well-matched to their own cluster and poorly matched to other clusters.

Gap Statistic

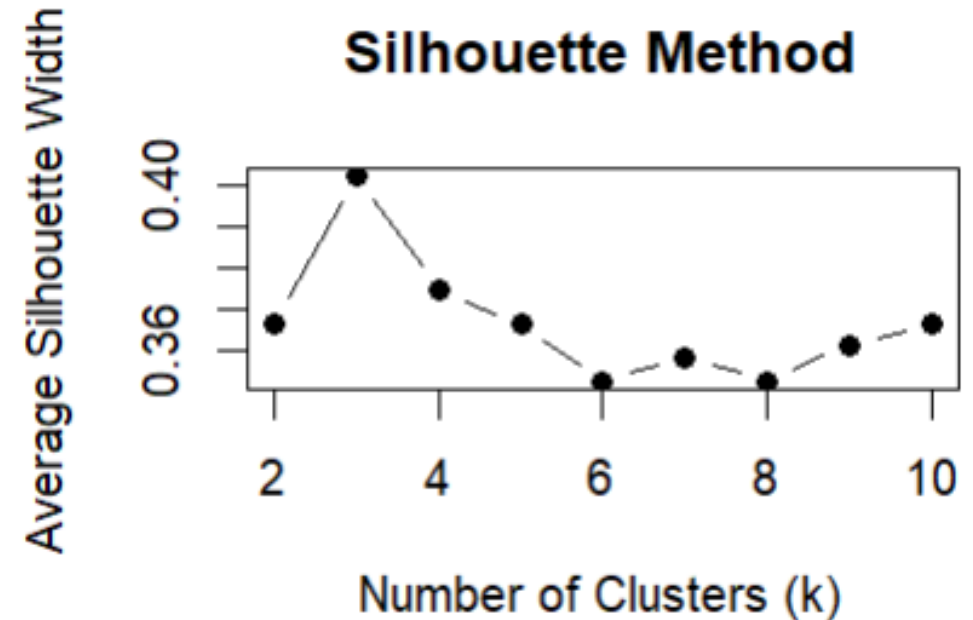
```
60 library(cluster)
61 set.seed(123)
62 gap_stat <- clusGap(scaled_selected, FUN = kmeans, nstart = 25,
63                   K.max = 10, B = 50)
64 print(gap_stat)
65 plot(gap_stat)
66
```

Graph: Plots the gap statistic against k.

Best Point: The highest point in the graph indicates the optimal k.

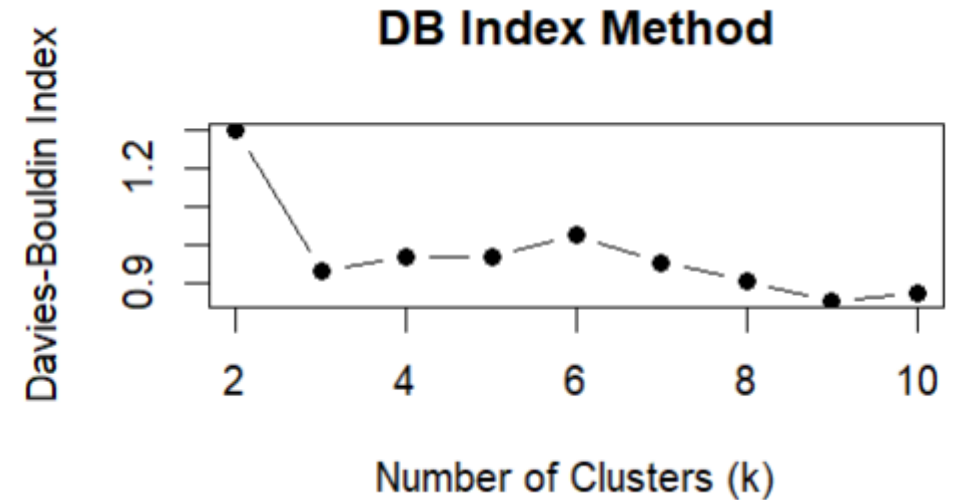
Interpretation:

A large gap means clustering is significantly better than random clustering.



Davies-Bouldin Index

```
68 #Find Optimum K - Davies-Bouldin Index ----
69 #Choose the low point
70 #install.packages("clusterSim")
71 library(clusterSim)
72
73 db_values <- sapply(2:10, function(k) {
74   kmeans_result <- kmeans(scaled_selected, centers = k, nstart = 25)
75   index.DB(scaled_selected, kmeans_result$cluster)$DB
76 })
77
78 plot(2:10, db_values, type = "b", pch = 19,
79      xlab = "Number of Clusters (k)",
80      ylab = "Davies-Bouldin Index",
81      main = "DB Index Method")
```



Graph: Plots the Davies-Bouldin Index against k.

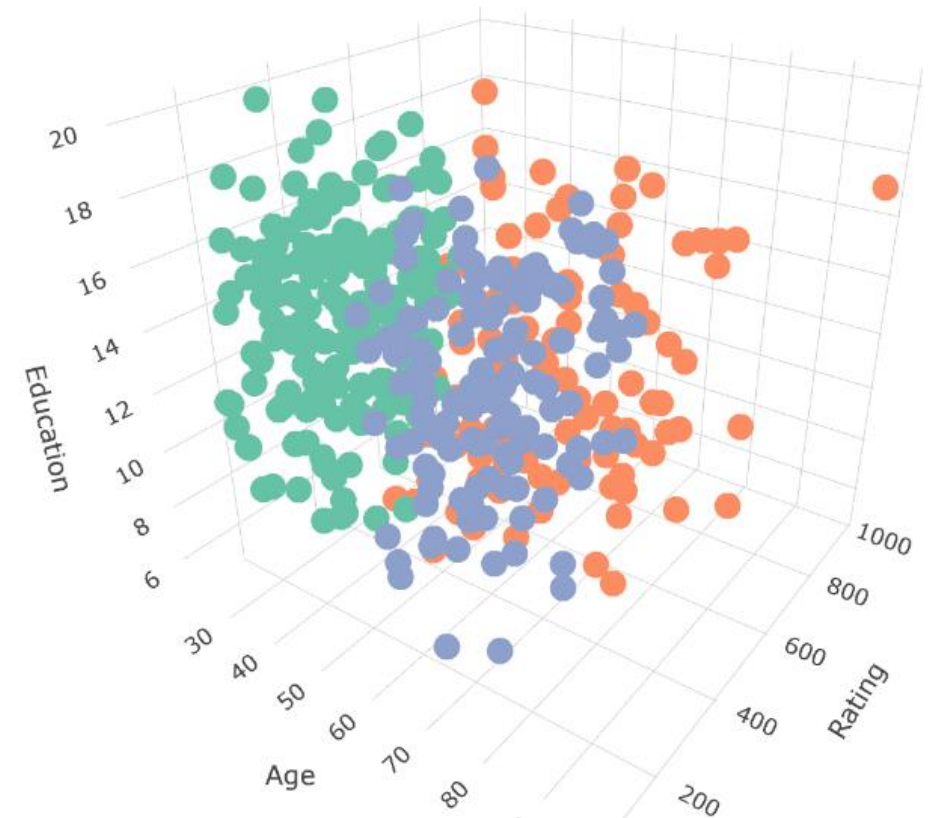
Best Point: The lowest point in the graph indicates the optimal k.

Interpretation:

Lower values indicate compact, well-separated clusters.

3 Features Clustering

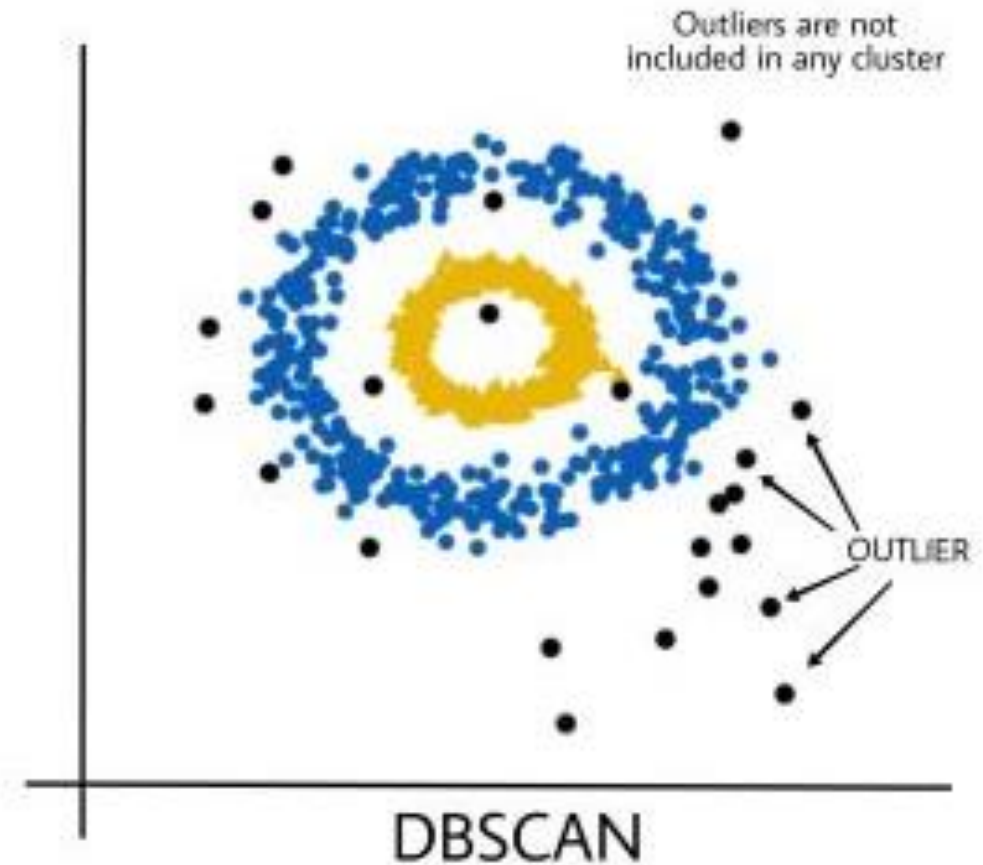
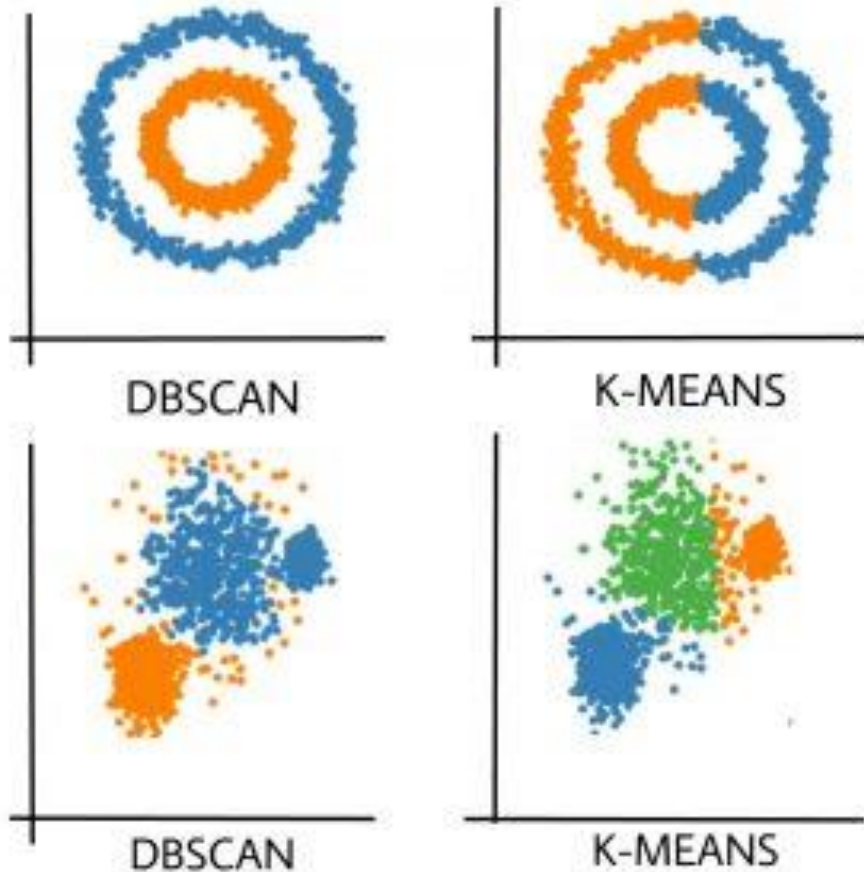
```
83 #3 Features Clustering----
84 library(ISLR)
85
86 # Load the dataset
87 data(Credit)
88
89 # Select two features
90 selected_features <- Credit[, c("Age", "Rating", "Education")]
91 scaled_selected <- scale(selected_features)
92
93 # K-means clustering
94 set.seed(123)
95 kmeans_result <- kmeans(scaled_selected, centers = 3, nstart = 25)
96
97 # Add cluster assignments to the data
98 Credit$Cluster <- kmeans_result$cluster
99 View(Credit)
100
101 # 3D Scatter plot of clusters
102 library(plotly)
103 #install.packages("plotly")
104
105 plot_ly(Credit, x = ~Age, y = ~Rating, z = ~Education,
106         color = ~factor(Cluster), type = 'scatter3d', mode = 'markers')
107
```



Density Based Clustering



Density-Based Spatial Clustering of Applications with Noise (DB-SCAN)



Advantages of DB-SCAN Clustering

1. No Need to Predefine Number of Clusters

2. Identifies Arbitrarily Shaped Clusters

DBSCAN can detect clusters of any shape, such as circular, elliptical, or elongated, making it suitable for datasets with irregular cluster shapes.

3. Handles Noise and Outliers

DBSCAN marks sparse regions as noise and excludes them from clusters. This makes it effective in identifying and isolating outliers

4. Works Well with Different Densities (within a cluster)

5. Efficient for Large Datasets

Disadvantages of DB-SCAN Clustering

1. Poor parameter selection can result in:

- Too many clusters (over-clustering).
- Merging of distinct clusters (under-clustering).
- Excessive noise classification.
- The algorithm is sensitive to the choice of *eps* (radius) and *minPts* (minimum points)

2. Scalability Issues in High Dimensions

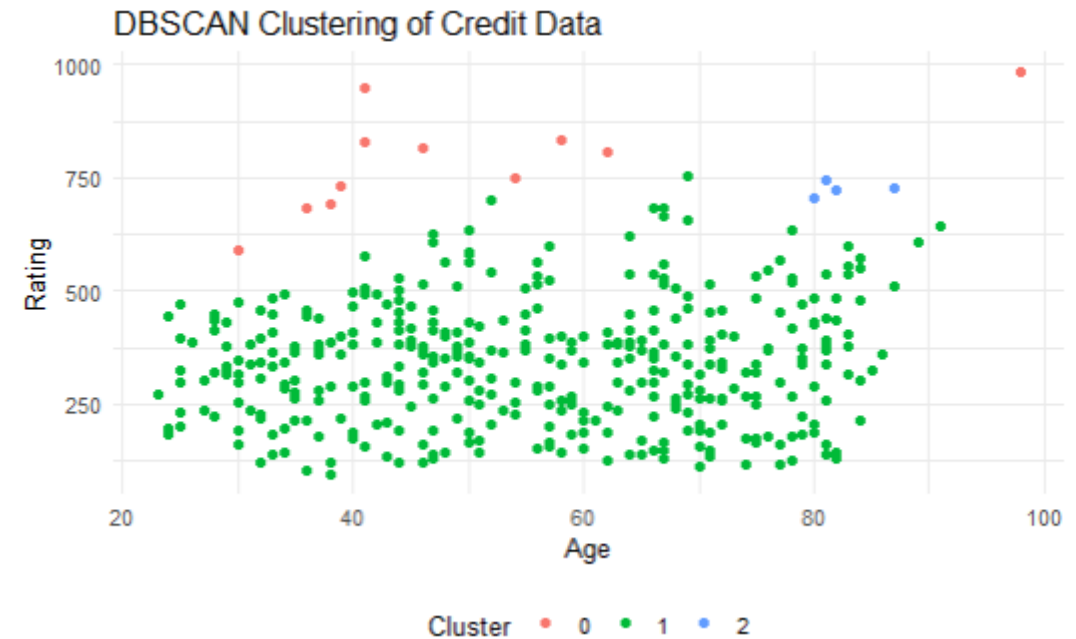
In high-dimensional datasets, the concept of "distance" becomes less meaningful (curse of dimensionality), making it hard to tune parameters or identify clusters effectively.

3. Difficulty with Border Points

Points on the boundary between two clusters may be arbitrarily assigned to one cluster, potentially reducing interpretability

2 Features DB-SCAN

```
110 #DB-SCAN Clustering----
111 # Load necessary libraries
112 #install.packages("dbscan")
113 library(ISLR)
114 library(dbscan)
115
116 # Step 1: Load and prepare the data
117 data(Credit)
118 selected_features <- Credit[, c("Age", "Rating")]
119
120 # Step 2: Standardize the features (recommended for DBSCAN)
121 scaled_features <- scale(selected_features)
122
123 # Step 3: Apply DBSCAN
124 # Set eps (neighborhood size) and minPts (minimum points for a cluster)
125 dbscan_result <- dbscan(scaled_features, eps = 0.5, minPts = 5)
126
127 # Step 4: Inspect the clustering result
128 print(dbscan_result)
129
130 # Add cluster labels back to the original dataset
131 Credit$Cluster <- dbscan_result$cluster
132
133 # View the updated dataset
134 View(Credit)
135
136 # Step 5: Visualize the clusters
137 library(ggplot2)
138 ggplot(Credit, aes(x = Age, y = Rating, color = factor(Cluster))) +
139   geom_point() +
140   theme_minimal() +
141   labs(title = "DBSCAN Clustering of Credit Data",
142        color = "Cluster") +
143   theme(legend.position = "bottom")
144
```



Cluster 0 represents noise points or outliers.

Parameter Tuning:

If **eps** is too small, some points might not have enough neighbors and will be labeled as noise.

If **minPts** is too high, fewer points will qualify as core points, resulting in more points being labeled as noise.

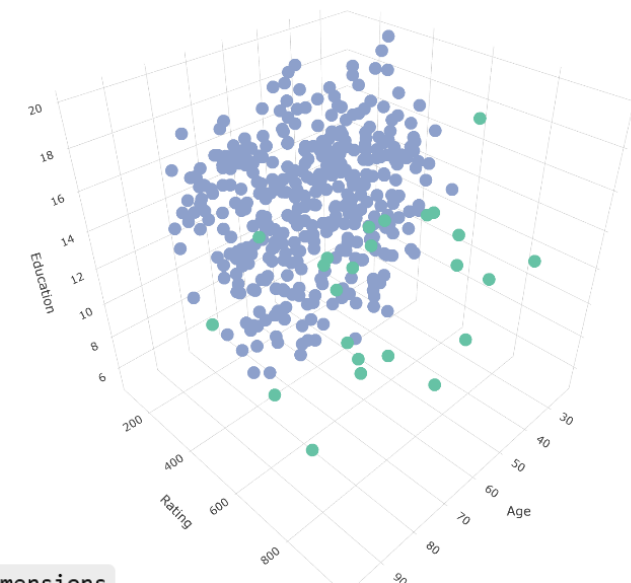
3 Features DB-SCAN

```

145 #3 Features DB-SCAN Clustering---|
146 # Load necessary libraries
147 #install.packages("dbscan")
148 library(ISLR)
149 library(dbscan)
150
151 # Step 1: Load and prepare the data
152 data(Credit)
153 selected_features <- Credit[, c("Age", "Rating", "Education")]
154
155 # Step 2: Standardize the features (recommended for DBSCAN)
156 scaled_features <- scale(selected_features)
157
158 # Step 3: Apply DBSCAN
159 # Set eps (neighborhood size) and minPts (minimum points for a cluster)
160 dbscan_result <- dbscan(scaled_features, eps = 0.8, minPts = 6)
161
162 # Step 4: Inspect the clustering result
163 print(dbscan_result)
164
165 # Add cluster labels back to the original dataset
166 Credit$Cluster <- dbscan_result$cluster
167
168 # View the updated dataset
169 View(Credit)
170
171 # Step 5: Visualize the clusters
172 # 3D Scatter plot of clusters
173 library(plotly)
174 #install.packages("plotly")
175
176 plot_ly(Credit, x = ~Age, y = ~Rating, z = ~Education,
177         color = ~factor(Cluster), type = 'scatter3d', mode = 'markers')
178

```

Cluster

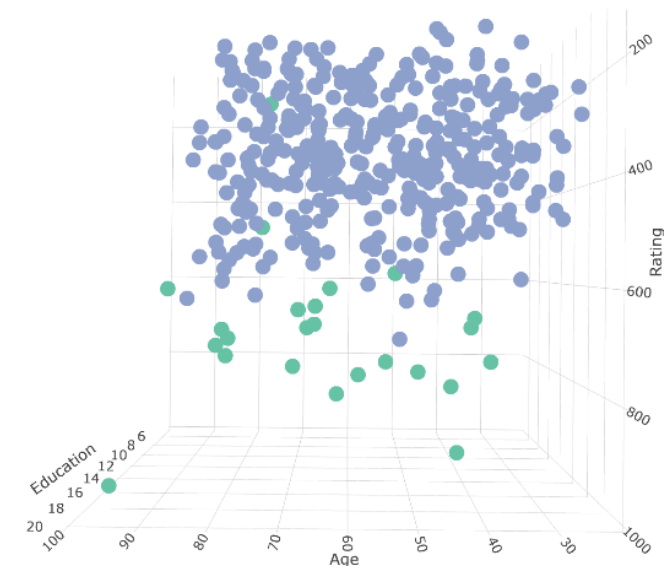


The rule of thumb for `minPts` :


- `minPts = 2 * number of dimensions`

For a 3-dimensional dataset:

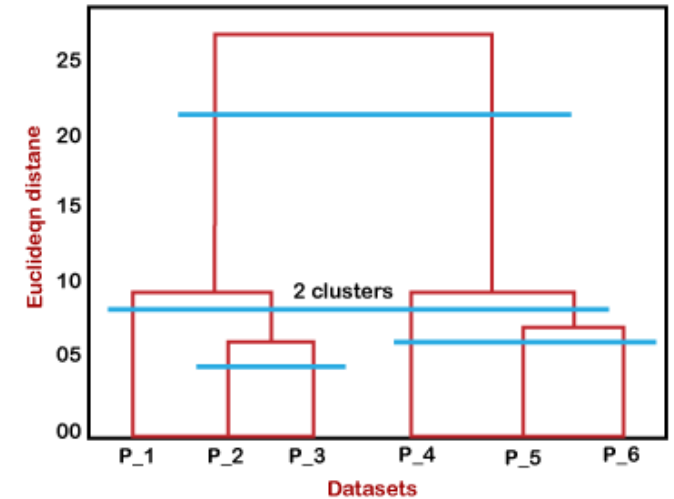
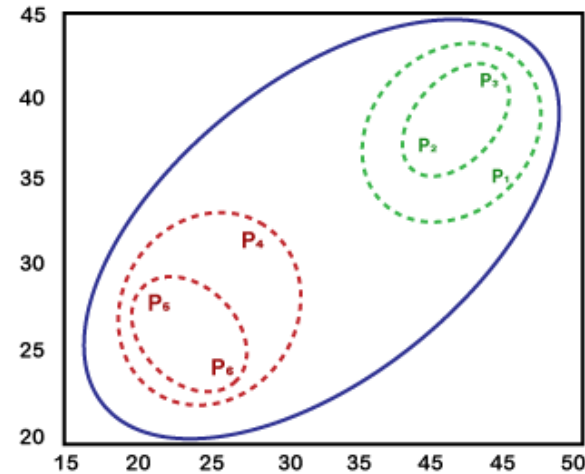
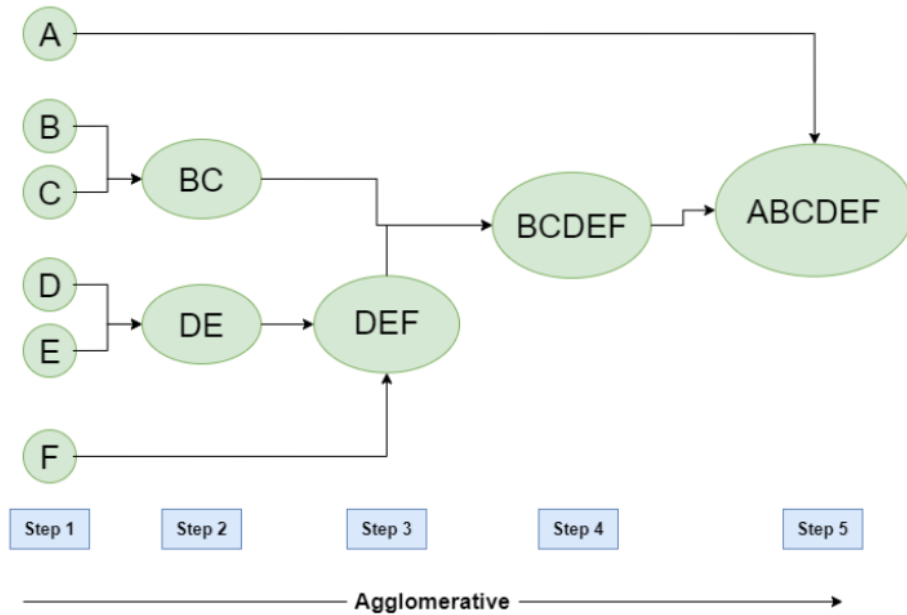
- `minPts = 2 * 3 = 6`



Hierarchical Clustering

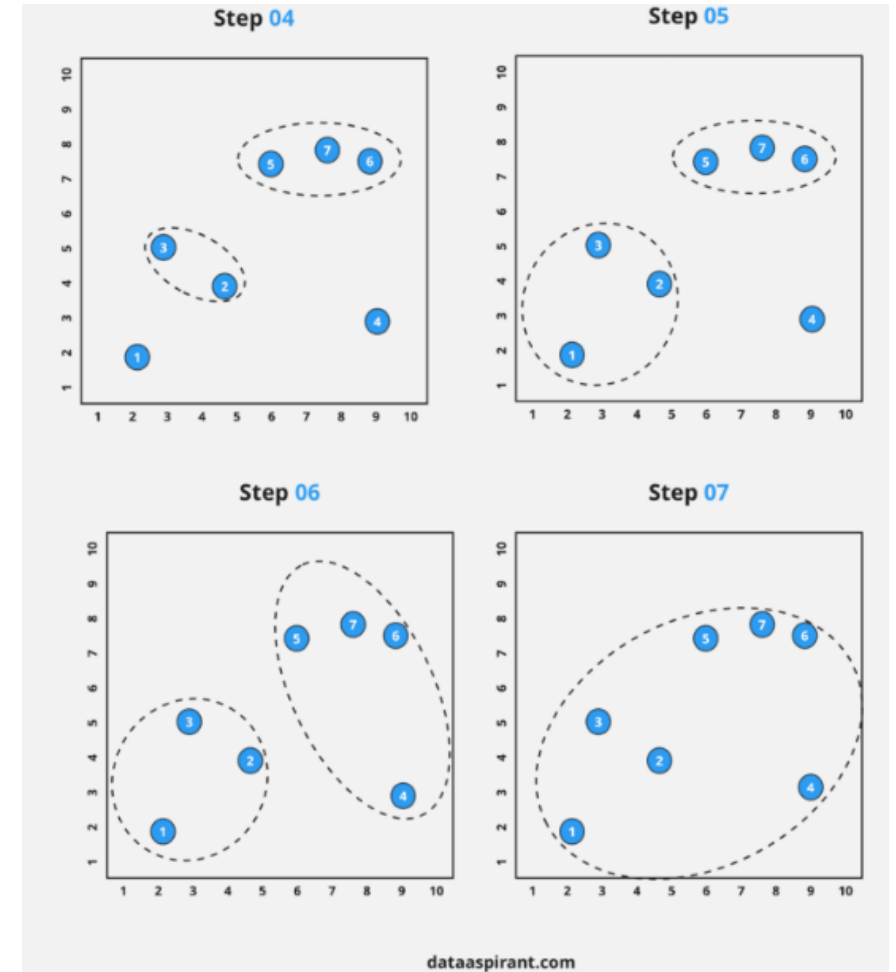
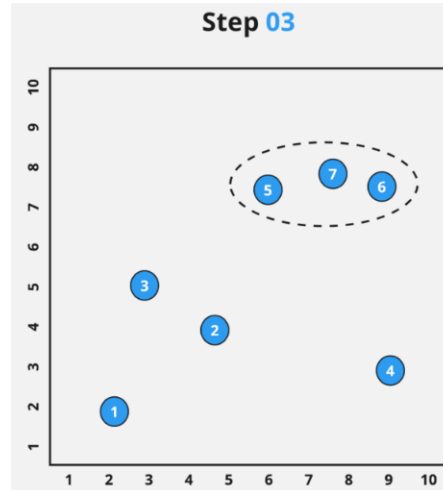
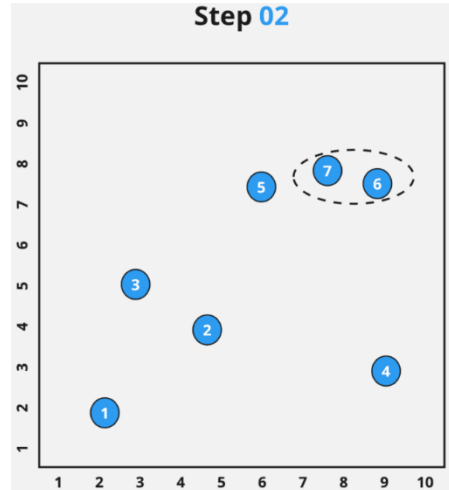
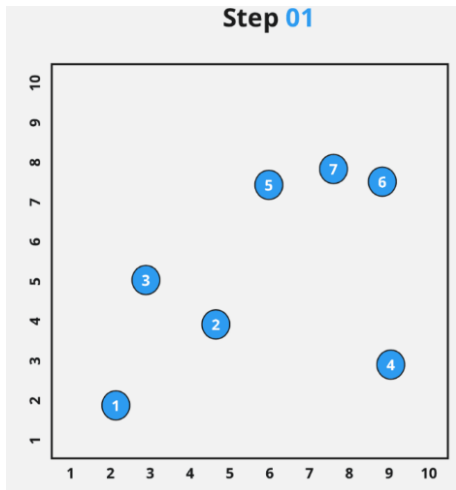


Hierarchical Clustering



It builds a hierarchy (tree structure or dendrogram) of clusters, which can then be divided into clusters at various levels

Agglomerative Hierarchical Clustering



Divisive method is essentially the **reverse of the Agglomerative method**

Advantages of Hierarchical Clustering

No Need to Predefine k

Unlike K-Means, you don't need to specify the number of clusters upfront.

Dendrogram for Exploration:

Provides a visual representation of how data points group at different levels.

Works with Any Distance Metric:

Can use various similarity/distance measures, offering flexibility.

Hierarchical Insight:

Gives insight into the structure and relationships of data.

Disadvantages of Hierarchical Clustering

1. Computationally Intensive:

For large datasets, the algorithm is slow due to the need to calculate pairwise distances.

2. Scalability Issues:

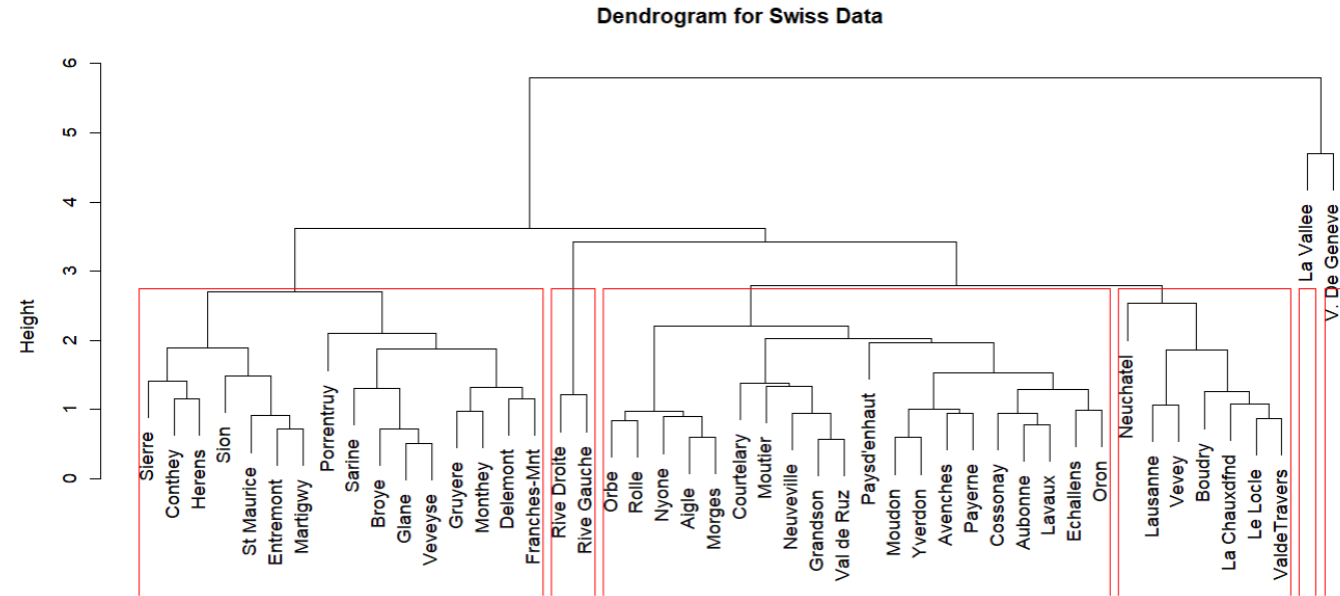
Struggles with very large datasets because of memory and computational demands.

3. Not Robust to Noise and Outliers:

Sensitive to noisy or irrelevant features, which can distort cluster formation.

Hierarchical Clustering in R

```
179 # Hierarchical clustering----
180 data(swiss)
181 View(swiss)
182 hc <- hclust(dist(scale(swiss)), method = "average")
183
184 # Plot the dendrogram
185 plot(hc, main = "Dendrogram for Swiss Data")
186
187 # Highlight the clusters (e.g., 3 clusters)
188 rect.hclust(hc, k = 6, border = "red")
189
190 # Assign clusters using cutree
191 clusters <- cutree(hc, k = 6)
192
193 # Add the cluster assignments as a new column
194 swiss$Cluster <- clusters
195
```



Performance Comparison

Performance Comparison

Criterion	DBSCAN	K-Means	Hierarchical Clustering
Cluster Shape	Arbitrary (e.g., circular, elongated)	Spherical	Arbitrary
Handles Noise and Outliers	Yes, effectively isolates noise	No, outliers distort centroids	No, outliers can affect dendrogram
Number of Clusters (k)	Determined automatically based on density	Must be predefined	Can decide based on dendrogram
Scalability (Dataset Size)	Efficient for large datasets (with indexing)	Highly scalable for large datasets	Less scalable, computationally expensive
Density Variations	Struggles with varying densities	Not suitable for density-based clusters	Handles density variations to some extent
High Dimensionality	Struggles due to distance metric limitations	Performs well in high dimensions	Computationally expensive in high dimensions

Recommendation

Scenario	Best Algorithm	Reason
Clusters with arbitrary shapes and noise	DBSCAN	Handles non-spherical clusters and outliers effectively.
Well-separated, spherical clusters with no noise	K-Means	Works efficiently for simple and structured datasets.
Exploratory clustering with no predefined k	Hierarchical Clustering	Dendrogram helps explore clusters at different levels.
Dataset contains noise or outliers	DBSCAN	Detects and isolates sparse data points.
High-dimensional data	K-Means	Scales well in high-dimensional spaces.
Small datasets with hierarchical relationships	Hierarchical Clustering	Provides hierarchical insights and is computationally feasible for small datasets.
Spatial or geographic data analysis	DBSCAN	Groups dense areas effectively and isolates sparse regions as noise.

Thank you

