APPLIED BUSINESS ANALYTICS

# Mining Relationships Among Records

NOVA SCHOOL OF BUSINESS & ECONOMICS



PATRÍCIA XUFRE

# Cluster Analysis

# What is Cluster Analysis?

Cluster analysis is a data analysis technique that **explores the naturally occurring groups within a data set** (the clusters).
Cluster analysis does not need to group data points into any predefined groups - it is an unsupervised learning method.
In unsupervised learning, insights are derived from the data without any predefined labels or classes.
A good clustering algorithm **ensures high intra-cluster similarity and low inter-cluster similarity**.

# Dissimilarity Measure

A distance between two points $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{y} = (y_1, \dots, y_n)$ is a function $d(\mathbf{x}, \mathbf{y})$ that satisfies the following properties:

i) $d(\mathbf{x}, \mathbf{y}) = 0 \; if \; \mathbf{x} = \mathbf{y}$ and $d(\mathbf{x}, \mathbf{y}) > 0 \; if \; \mathbf{x} \neq \mathbf{y}$

ii) $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$

iii) $d(\mathbf{x}, \mathbf{z}) \leq d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z})$

What is being measured? How are the different measurements related? What scale should each measurement be treated as (numerical, ordinal, or nominal)? Are there outliers? Depending on the goal of the analysis, should the clusters be distinguished mostly by a small set of measurements, or should they be separated by multiple measurements that weight moderately?

# Distance Measures for Numerical Data

## Euclidian Distance

$$d_{ij} = \sqrt{(x_{i1} - x_{j1})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

- highly scale dependent
- ignores the relationship between the measurements
- is sensitive to outliers

## Correlation-based dissimilarity

$$d_{ij} = 1 - r_{ij}^2$$

# Distance Measures for Numerical Data

## Statistical Distance (Mahalanobis Distance)

$$d_{ij} = \sqrt{(\mathbf{x}_i - \mathbf{x}_j)^T S^{-1} (\mathbf{x}_i - \mathbf{x}_j)}$$

- takes into account the correlation between measurements

## Manhattan Distance

$$d_{ij} = |x_{i1} - x_{j1}| + \cdots + |x_{ip} - x_{jp}|$$

# Distance Measures for Numerical Data

## Maximum Coordinate Distance

$$d_{ij} = \max_{k=1,\ldots,p} |x_{ik} - x_{jk}|$$

Most of these measures are highly scale dependent.
There are two possible ways to overcome this fact:
1) Normalizing the variables
2) Unequal weighting should be considered if we want the clusters to depend more on certain variables and less on others

# Similarity Measures for Categorical Data

| | 0 | 1 | |
|---|---|---|---|
| 0 | $a$ | $b$ | $a+b$ |
| 1 | $c$ | $d$ | $c+d$ |
| | $a+c$ | $b+d$ | |

## Matching coefficient

$$s_{ij} = \frac{a + d}{a + b + c + d}$$

## Jaquard's coefficient

$$s_{ij} = \frac{d}{b + c + d}$$

- ignores zero matches

# Similarity Measures for Mixed Data

## Gower's similarity measure

$$s_{ij} = \frac{\sum_{m=1}^{p} w_{ijm} s_{ijm}}{\sum_{m=1}^{p} w_{ijm}}$$

where $s_{ijm}$ is the similarity between records *i* and *j* on variable *m*, and $w_{ijm}$ is a binary weight given to the corresponding distance.

- Variables must be first standardized ([0, 1])

- For **continuous variables**, $s_{ijm} = \frac{|x_{im} - x_{jm}|}{\max(x_m) - \min(x_m)}$ and $w_{ijm} = 1$ unless the value for variable *m* is unknown for one or both of the records, in which case $w_{ijm} = 0$.
- For **binary variables**, $s_{ijm} = 1$ if $x_{im} = x_{jm} = 1$ and 0 otherwise. $w_{ijm} = 1$ unless $x_{im} = x_{jm} = 0$.
- For **nonbinary categorical variables**, $s_{ijm} = 1$ if both records are in the same category, and otherwise $s_{ijm} = 0$. The binary weight is $w_{ijm} = 1$ unless the category for variable *m* is unknown for one or both of the records, in which case $w_{ijm} = 0$.

# Measuring Distance Between Two Clusters

**Cluster:** set of one or more records

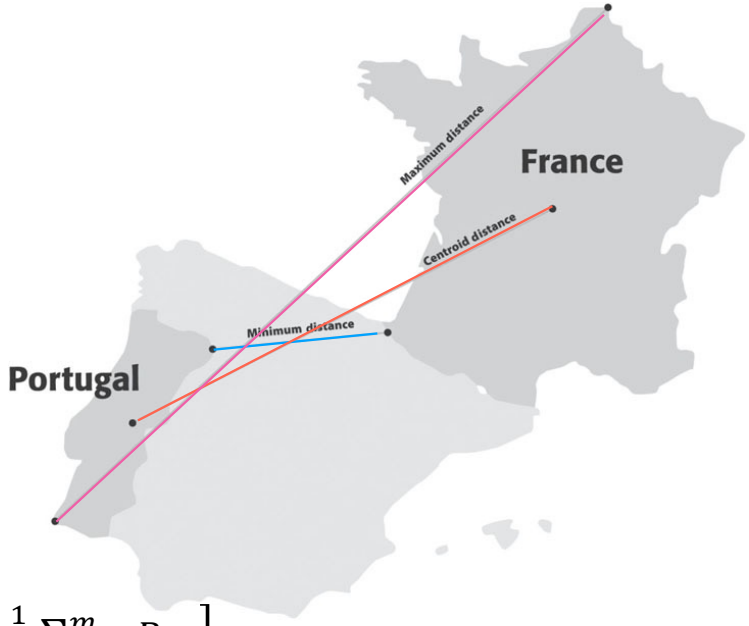
Consider **cluster** $\boldsymbol{A} = \{A_1, \dots, A_m\}$ and **cluster** $\boldsymbol{B} = \{B_1, \dots, B_n\}$, the distance between these two clusters can be defined as:

**Minimum Distance** $\min_{i,j} \left( distance(\boldsymbol{A_i}, \boldsymbol{B_j}) \right)$

**Maximum Distance** $\max_{i,j} \left( distance(\boldsymbol{A_i}, \boldsymbol{B_j}) \right)$

**Average Distance** $\text{average} \left( distance(\boldsymbol{A_i}, \boldsymbol{B_j}) \right)$

**Centroid Distance** $distance(\bar{\boldsymbol{x}}_{\boldsymbol{A}}, \bar{\boldsymbol{x}}_{\boldsymbol{B}})$

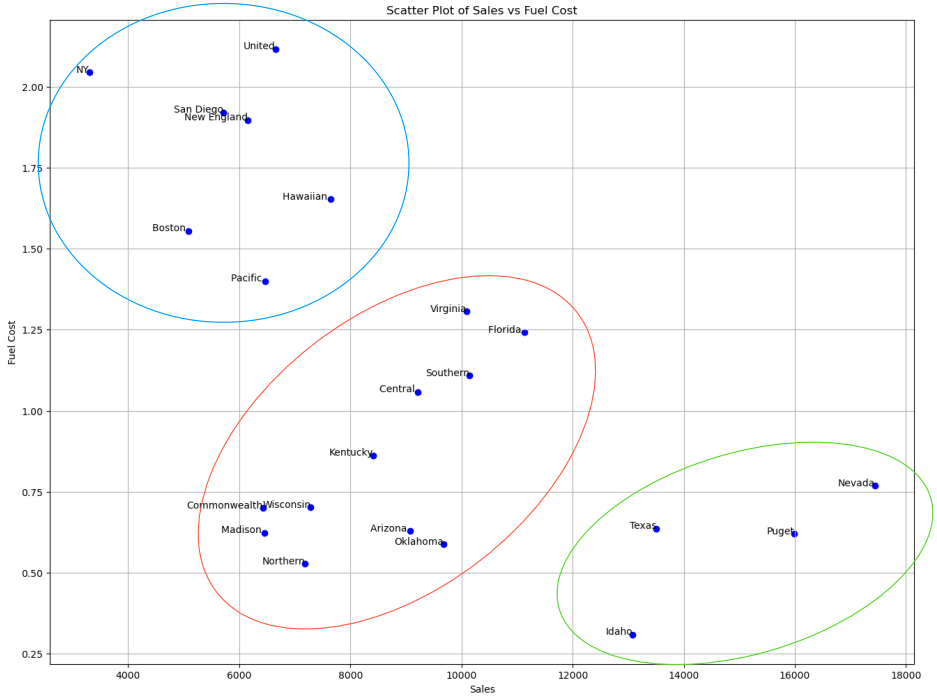



where $\bar{\boldsymbol{x}}_{\boldsymbol{A}} = \left[ \frac{1}{m} \sum_{k=1}^{m} A_{k1}, \dots, \frac{1}{m} \sum_{k=1}^{m} A_{kp} \right]$ and $\bar{\boldsymbol{x}}_{\boldsymbol{B}} = \left[ \frac{1}{n} \sum_{k=1}^{n} B_{k1}, \dots, \frac{1}{m} \sum_{k=1}^{m} B_{kp} \right]$

# Public Utilities

| Company | Fixed_charge | RoR | Cost | Load_factor | Demand_growth | Sales | Nuclear | Fuel_Cost |
|---|---|---|---|---|---|---|---|---|
| **Arizona** | 1.06 | 9.2 | 151 | 54.4 | 1.6 | 9077 | 0.0 | 0.628 |
| **Boston** | 0.89 | 10.3 | 202 | 57.9 | 2.2 | 5088 | 25.3 | 1.555 |
| **Central** | 1.43 | 15.4 | 113 | 53.0 | 3.4 | 9212 | 0.0 | 1.058 |
| **Commonwealth** | 1.02 | 11.2 | 168 | 56.0 | 0.3 | 6423 | 34.3 | 0.700 |
| **NY** | 1.49 | 8.8 | 192 | 51.2 | 1.0 | 3300 | 15.6 | 2.044 |

Fixed = fixed-charge covering ratio (income/debt)
RoR = rate of return on capital
Cost = cost per kilowatt capacity in place
Load = annual load factor
Demand = peak kilowatthour demand growth from 1974 to 1975
Sales = sales (kilowatthour use per year)
Nuclear = percent nuclear
Fuel Cost = total fuel costs (cents per kilowatthour)



# Public Utilities

| Company | Arizona | Boston | Central | Commonwealth | NY |
|---|---|---|---|---|---|
| Company | | | | | |
| Arizona | 0.000000 | 2.010329 | 0.774179 | 0.758738 | 3.021907 |
| Boston | 2.010329 | 0.000000 | 1.465703 | 1.582821 | 1.013370 |
| Central | 0.774179 | 1.465703 | 0.000000 | 1.015710 | 2.432528 |
| Commonwealth | 0.758738 | 1.582821 | 1.015710 | 0.000000 | 2.571969 |
| NY | 3.021907 | 1.013370 | 2.432528 | 2.571969 | 0.000000 |

**Minimum Distance:** 0.76
**Maximum Distance:** 3.02
**Average Distance:** $\frac{0.77+0.76+3.02+1.47+1.58+1.01}{6} = 1.44$

Consider only the two variables Sales and Fuel Cost:
**Centroid A** $\left(\frac{0.0459-1.0778}{2}, \frac{-0.8537+0.8133}{2}\right) = (-0.516, -0.020)$
**Centroid B** $\left(\frac{0.0839-0.7017-1.5814}{3}, \frac{-0.0804-0.7242+1.6926}{3}\right) = (-0.733, 0.296)$
Centroid Distance: $\sqrt{(-0.516+0.733)^2 + (-0.020-0.296)^2} = 0.38$

# Clustering Algorithms

## Hierarchial Methods

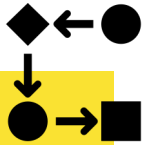
- Agglomerative or Divisive Methods
- Useful when the goal is to arrange the clusters into a natural hierarchy

## Non-hierarchical Methods

- Using a prespecified number of clusters, the methods assigns records to each cluster
- Less computationally intensive

HIERARCHICAL

# Agglomerative Clustering Algorithm



1. Start with n clusters (each record = cluster).
2. The two closest records are merged into one cluster.
3. At every step, the two clusters with the smallest distance are merged. This means that either single records are added to existing clusters or two existing clusters are combined.

# Linkage Methods

## **Single-linkage** (Minimum Linkage)

- Distance between two clusters is defined as the **shortest distance** between any two points in the two clusters.
- It tends to form long, elongated clusters, and it's sensitive to noise and outliers.
- It's computationally efficient but can produce chains of points.

## **Complete-linkage** (Maximum Linkage

- Distance between two clusters is defined as the **maximum distance** between any two points in the two clusters.
- It tends to form compact, spherical clusters and is less sensitive to noise and outliers compared to single linkage.
- It can capture compact clusters well but may struggle with elongated or irregularly shaped clusters.

# Linkage Methods

**Average Linkage** (Unweighted Pair Group Method with Arithmetic Mean)

- Distance between two clusters is defined as the **average distance** between all pairs of points from the two clusters.
- It strikes a balance between single and complete linkage and can handle some degree of noise and outliers.
- It tends to produce clusters with more balanced sizes and is less sensitive to outliers compared to single linkage.

**Centroid Linkage** (Unweighted Pair Group Method with Centroid Mean)

- Distance between two clusters is defined as the **distance between the centroids** of the two clusters.
- It can produce clusters of varying shapes and sizes and is less sensitive to noise compared to single linkage.
- It may merge clusters with large differences in size or density, leading to less interpretable clusters.

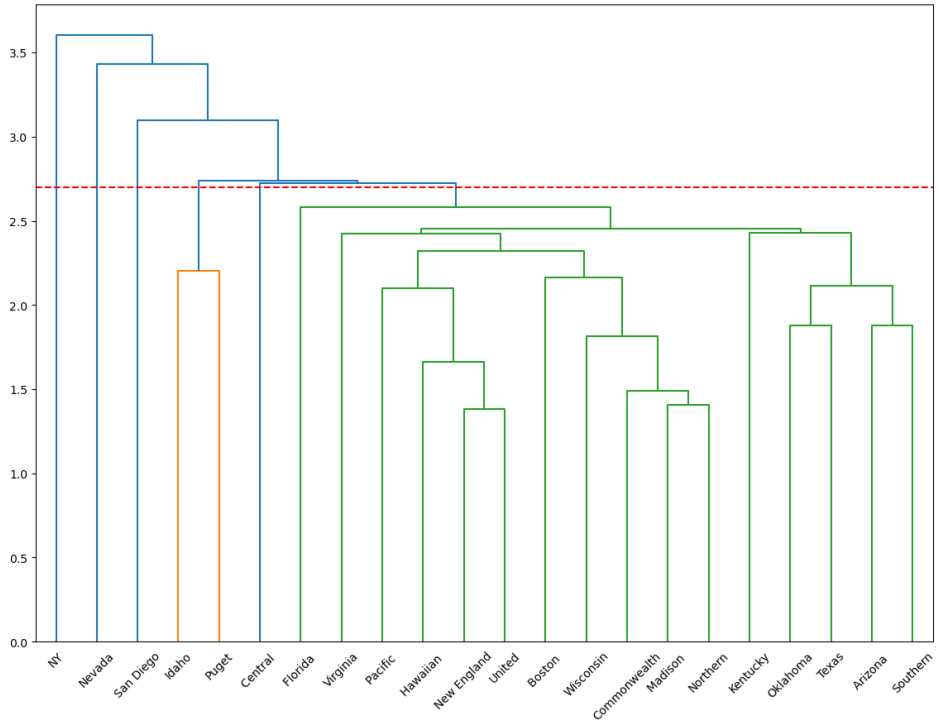
# Linkage Methods

## Ward's Linkage

- This method minimizes the variance when merging clusters.
- It tends to produce clusters of relatively equal size and compactness.
- It is sensitive to noise and outliers, but it can handle large datasets well and is often used for exploratory data analysis.

The choice of method depends on the nature of the data and the goals of the analysis. Experimentation and validation are often necessary to determine the most appropriate linkage method for a given dataset.

# Public Utilities

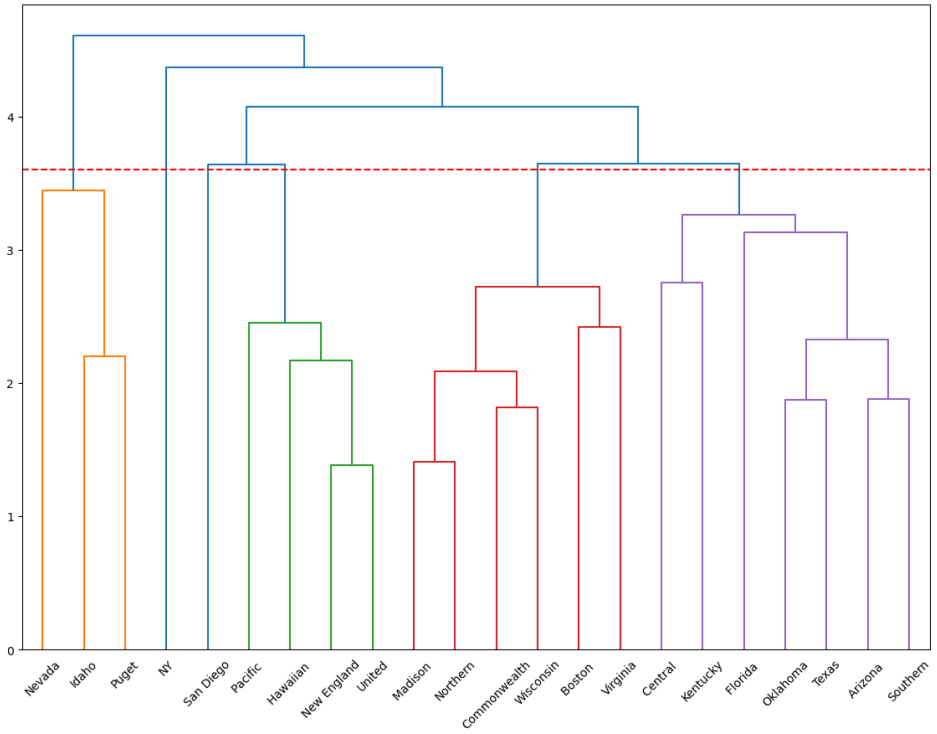



The dendrogram is a tree diagram that displays the groups that are formed by clustering observations at each step and their dissimilarity levels. The distance level is measured along the vertical axis and the different observations are listed along the horizontal axis.

```
# in linkage() set argument method =
# 'single', 'complete', 'average', 'weighted', centroid', 'median', 'ward'
plt.figure(figsize=(14, 10))
Z = linkage(df_norm, method='single')
dendrogram(Z, labels=df_norm.index, color_threshold=2.7)
plt.axhline(y=2.7, color='r', linestyle='--')
plt.show()
```

# Public Utilities

```
plt.figure(figsize=(14, 10))
Z = linkage(df_norm, method='average')
dendrogram(Z, labels=df_norm.index, color_threshold=3.6)
plt.axhline(y=3.6, color='r', linestyle='--')
plt.show()
```




```
memb = fcluster(linkage(df_norm, method='average'), 6, criterion='maxclust')
memb = pd.Series(memb, index=df_norm.index)
for key, item in memb.groupby(memb):
    print(key, ': ', ', '.join(item.index))
```

```
1 :  Idaho, Nevada, Puget
2 :  Hawaiian , New England, Pacific , United
3 :  San Diego
4 :  Boston , Commonwealth, Madison , Northern, Wisconsin, Virginia
5 :  Arizona , Central , Florida , Kentucky, Oklahoma, Southern, Texas
6 :  NY
```

# Validating Clusters

Meaningful Clusters

**Cluster interpretability**

**Summary measures for each variable in each cluster**

Examining the clusters for separation along some common variable that was not used in the cluster analysis

Labeling the clusters: Based on the interpretation, trying to assign a name or label to each cluster

**Cluster stability**

Cluster partition A.

Use the cluster centroids from A to assign each record in partition B (each record is assigned to the cluster with the closest centroid).

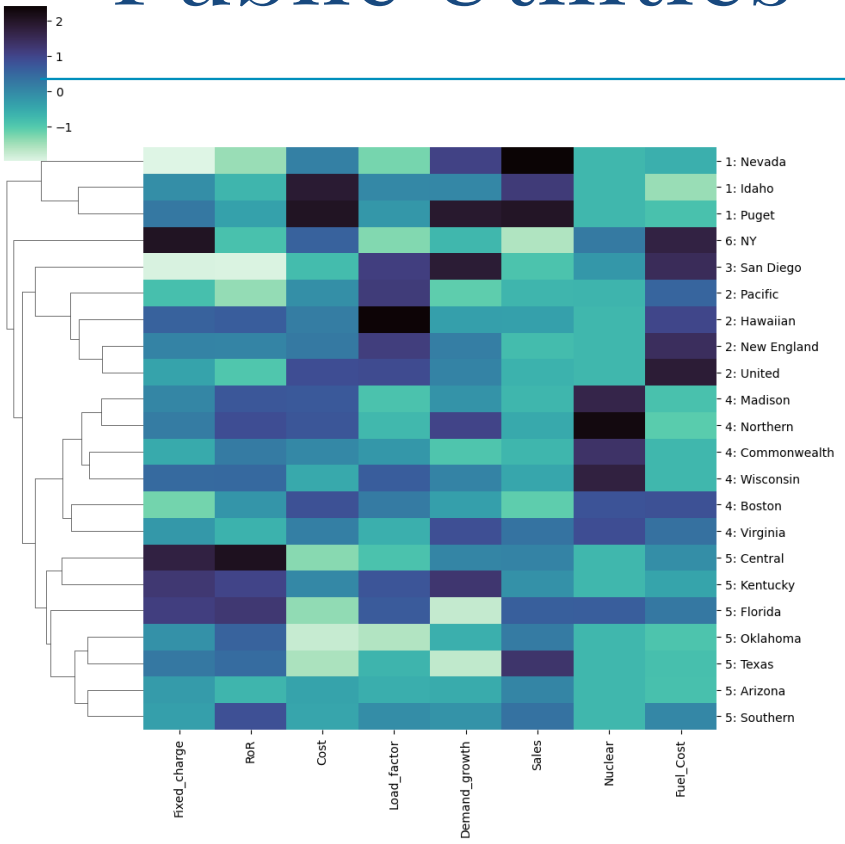
Assess how consistent the cluster assignments are compared to the assignments based on all the data.

**Cluster separation**

Examine the ratio of between-cluster variation to within-cluster variation to see whether the separation is reasonable.

**Number of clusters**

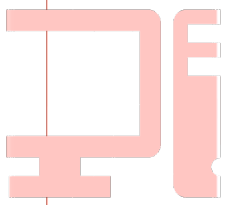
# Public Utilities




```
# set labels as cluster membership and utility name
df_norm.index = ['{}: {}'.format(cluster, state)
                                for cluster, state in zip(memb, df_norm.index)]
# plot heatmap
sns.clustermap(df_norm, method='average', col_cluster=False, cmap='mako_r')
```

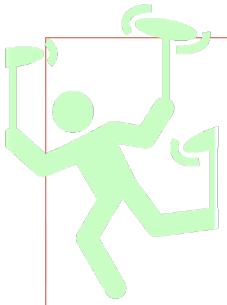
- cluster 4 is characterized by utilities with a high percent of nuclear power
- cluster 5 is characterized by high fixed charge and RoR
- cluster 2 has high fuel costs.

# Limitations



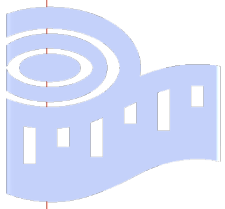
## High computational and storage power

- It requires the computational and storage of a $n \times n$ distance matrix. For very large dataset, this can be expensive and slow.



## Low stability

- The hierarchical algorithm makes only one pass through the data. This means that records that are allocated incorrectly early in the process cannot be reallocated subsequently. Reordering data or dropping a few records can lead to a different solution.



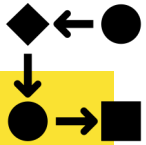
## Choice of the distance measure

- Single and complete linkage are robust to changes in the distance metric as long as the relative ordering is kept. In contrast, average linkage is more influenced by the choice of distance metric, and might lead to completely different clusters when the metric is changed.

## Sensitive to outliers

NON-HIERARCHICAL

# *K*-means Clustering Algorithm



1. Start with *k* initial clusters (user chooses *k*).
2. At every step, each record is reassigned to the cluster with the “closest” centroid.
3. Recompute the centroids of clusters that lost or gained a record, and repeat Step 2.
4. Stop when moving any more records between clusters increases cluster dispersion.

# Public Utilities

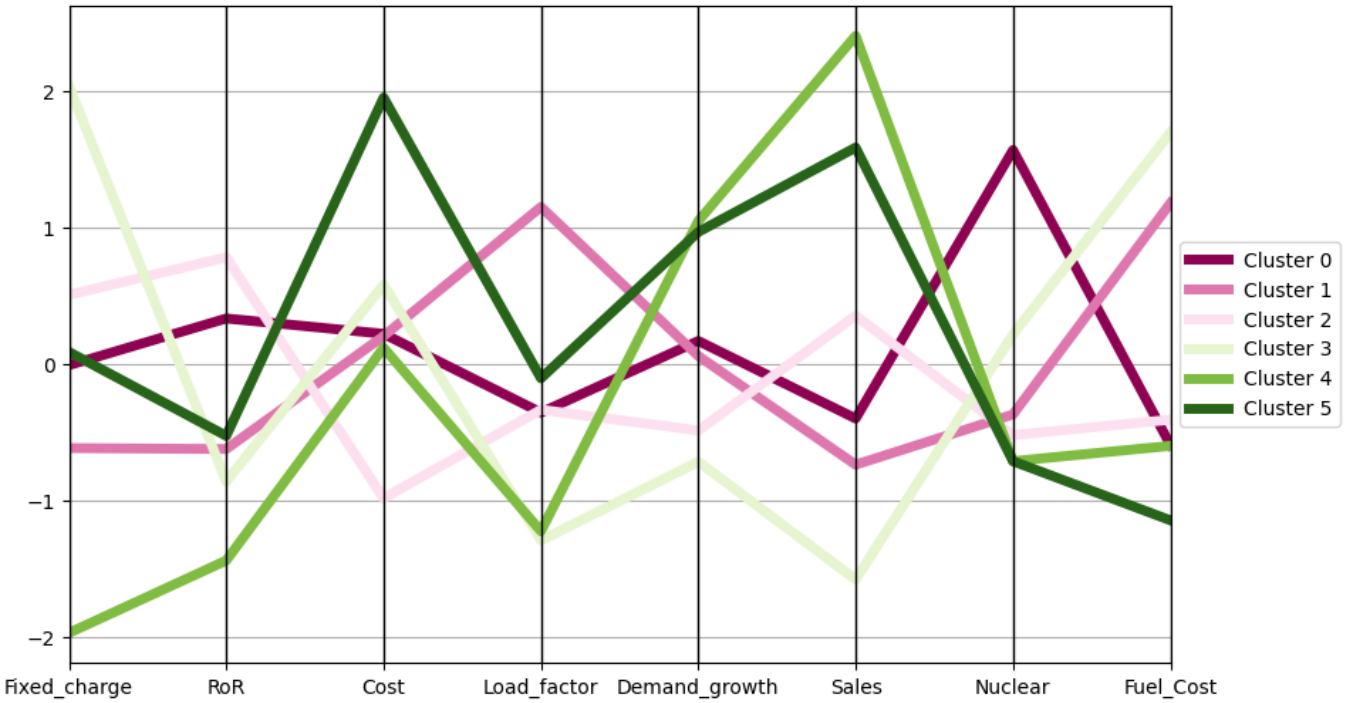
```
kmeans = KMeans(n_clusters=6, random_state=0).fit(df_norm)
# Cluster membership
memb_ = pd.Series(kmeans.labels_, index=df_norm.index)
for key, item in memb_.groupby(memb_):
    print(key, ': ', ', '.join(item.index))
```

```
0 :  Commonwealth, Madison , Northern, Wisconsin, Virginia
1 :  Boston , Hawaiian , New England, Pacific , San Diego, United
2 :  Arizona , Central , Florida , Kentucky, Oklahoma, Southern, Texas
3 :  NY
4 :  Nevada
5 :  Idaho, Puget
```

```
# calculate the distances of each data point to the cluster centers
distances = kmeans.transform(df_norm)
# find closest cluster for each data point
minSquaredDistances = distances.min(axis=1) ** 2
# combine with cluster labels into a data frame
df = pd.DataFrame({'squaredDistance': minSquaredDistances, 'cluster': kmeans.labels_},
    index=df_norm.index)
# group by cluster and print information
for cluster, data in df.groupby('cluster'):
    count = len(data)
    withinClustSS = data.squaredDistance.sum()
    print(f'Cluster {cluster} ({count} members): {withinClustSS:.2f} within cluster ')
```

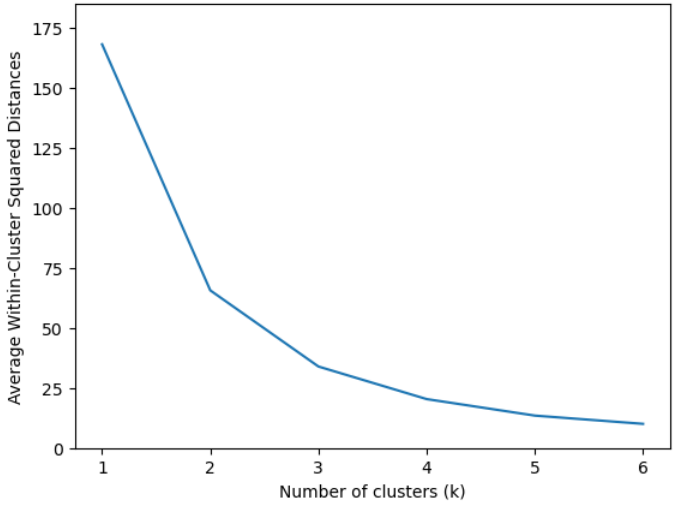
```
Cluster 0 (5 members): 10.18 within cluster
Cluster 1 (6 members): 21.19 within cluster
Cluster 2 (7 members): 26.51 within cluster
Cluster 3 (1 members): 0.00 within cluster
Cluster 4 (1 members): 0.00 within cluster
Cluster 5 (2 members): 2.42 within cluster
```

# Public Utilities

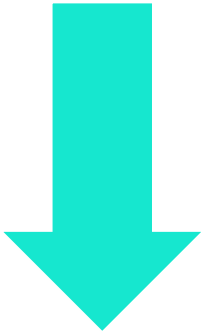


# Public Utilities

```
inertia = []
for n_clusters in range(1, 7):
    kmeans = KMeans(n_clusters=n_clusters, random_state=0).fit(df_norm)
    inertia.append(kmeans.inertia_ / n_clusters)
inertias = pd.DataFrame({'n_clusters': range(1, 7), 'inertia': inertia})
ax = inertias.plot(x='n_clusters', y='inertia')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Average Within-Cluster Squared Distances')
plt.ylim((0, 1.1 * inertias.inertia.max()))
ax.legend().set_visible(False)
plt.show()
```



# How to choose the best *k*?



## Elbow Method

Calculate the **Within-Cluster-Sum of Squared** Errors (WSS) for **different values of *k***, and choose the *k* for which WSS becomes first starts to diminish.

## Silhouette Method

The silhouette value measures how similar a point is to its own cluster (**cohesion**) compared to other clusters (**separation**).