Clustering P1

ENSF 444

Review what unsupervised learning is

Discuss how this can be applied to partition data

What is clustering?

A technique used in unsupervised learning that groups data points into clusters based on similarities. Clustering works with unlabeled data.

Why do we cluster?

Clustering is a fundamental task in unsupervised learning that involves grouping similar data points together into clusters or segments based on their inherent characteristics or patterns. There are several reasons why clustering is valuable in various fields:

1. **Exploratory Data Analysis:** Clustering helps in understanding the underlying structure of the data by identifying natural groupings or patterns that may not be apparent initially. It provides insights into the relationships and similarities among data points, which can guide further analysis and decision-making.
2. **Data Compression and Dimensionality Reduction:** Clustering can be used as a preprocessing step for data compression and dimensionality reduction. By grouping similar data points together, redundant information can be consolidated, leading to a more compact representation of the data without losing essential patterns or trends.
3. **Anomaly Detection:** Clustering can be used to identify outliers or anomalies in the data. Data points that do not belong to any cluster or are distant from the centroids of existing clusters can be considered anomalies, which may indicate unusual behavior, errors, or interesting events that warrant further investigation.
4. **Customer Segmentation:** In marketing and customer analytics, clustering is commonly used to segment customers into homogeneous groups based on their demographics, behaviors, or preferences. This allows businesses to tailor their products, services, and marketing strategies to different customer segments, thereby improving customer satisfaction and retention.

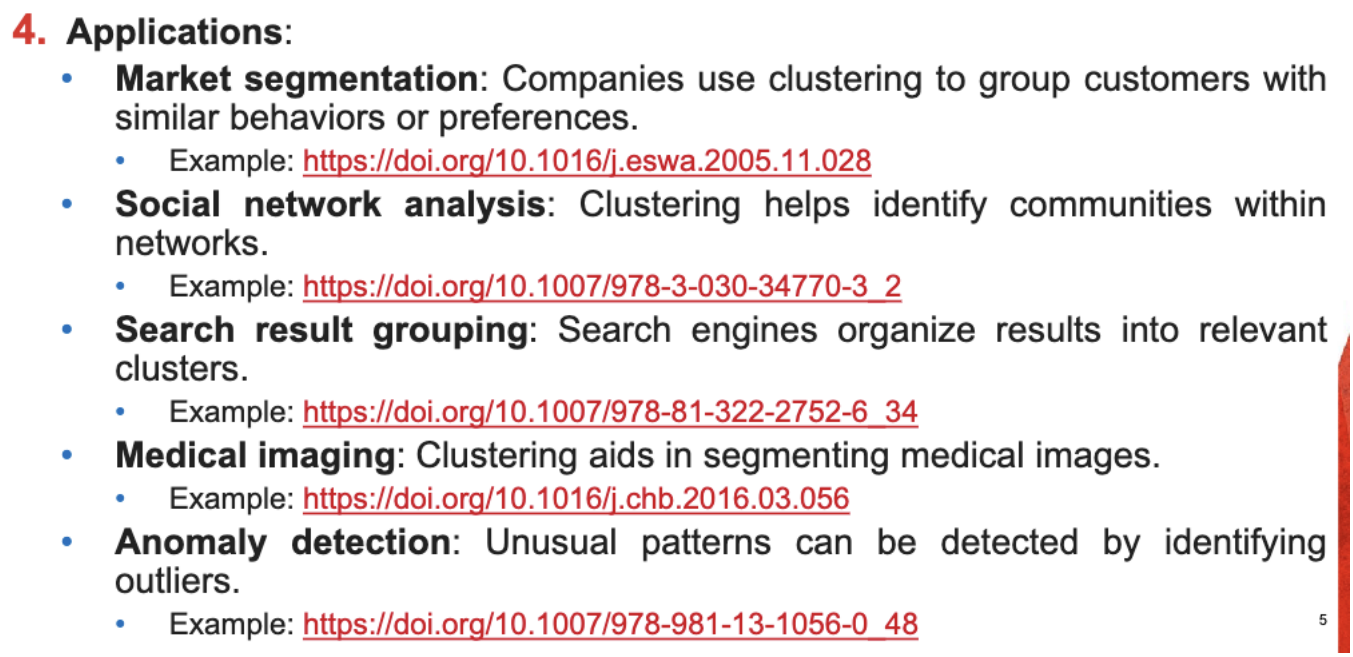
5. **Recommendation Systems:** Clustering can be applied to group similar items or users together in recommendation systems. By identifying clusters of similar items or users, personalized recommendations can be generated for individuals based on the preferences of their respective clusters.
6. **Image and Signal Processing:** In image and signal processing, clustering techniques are used for tasks such as image segmentation, where pixels with similar characteristics are grouped together to identify objects or regions of interest in images, or for clustering similar patterns in time series data.
7. **Biology and Bioinformatics:** Clustering is widely used in biology and bioinformatics for tasks such as gene expression analysis, protein classification, and identifying functionally related genes or proteins.

Overall, clustering is a versatile and powerful tool that can uncover hidden patterns, reduce data complexity, facilitate decision-making, and enable various applications across domains.

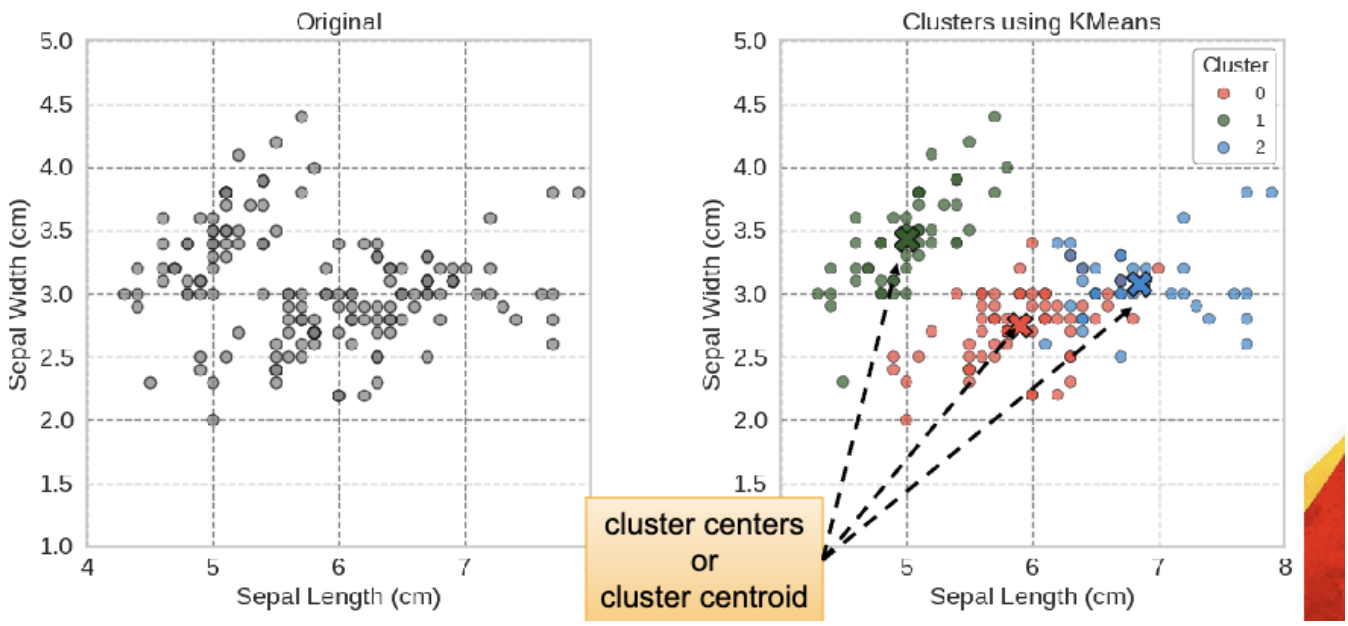
Some key points are;

1. **Definition**: Finds patterns in data and assigns them to groups based on their features.
2. **Purpose**: Reveals hidden structure of data and helps us to analyze it better, Use clustering to make predictions about new data points based on their membership.
3. **Example**: One common example of clustering is the analysis of the Iris dataset, which contains measurements of four attributes for 150 flowers from three species. Clustering can help us classify the flowers into three groups based on their sepal length and width.

Applications



Heres an example of thee Iris data set being placed into three categories based on their cluster membership



What are the different Clustering Methods?

- **K-means**
- **Hierarchical clustering**
- **DBSCAN**
- **Gaussian Mixture Models (GMMs)**
- Affinity Propagation
- Mean Shift
- Spectral clustering …
- https://scikit-learn.org/stable/modules/clustering.html

Clustering Terminology

1. **Cluster Label**: the identifier for which a data point is assigned
2. **Cluster Prediction**: Applying the predict() method to assign cluster labels to new data samples

3. **Label Retrieval**: Accessing the .lebels_ attribute to obtain the cluster labels of the trained model

Distance Metrics

Distance metric	Arguments	Distance Function
Euclidean (L_2)	N/A	$\sqrt{\sum(x-y)^2}$
Manhattan (L_1)	N/A	$\sum\|x-y\|$
Chebyshev	N/A	$\max(\|x-y\|)$
Minkowski	p, w	$\left(\sum w * \|x-y\|^p\right)^{1/p}$
Cosine	N/A	$\frac{x.y}{\sqrt{\sum x^2} * \sqrt{\sum y^2}}$

What is K-means?

K-means is a method to divide a set of data points into k distinct clusters, where k is a predefined number. The goal is to partition the data such that points in the same cluster are similar to each other, while points in different clusters are not.

How does K-means work?

- **Initialization:** Randomly select **k** points from the data as the initial cluster centers (centroids).
- **Assignment:** Assign each data point to the nearest centroid, forming **k** clusters.
- **Update:** Calculate the new centroid of each cluster by taking the **mean** of all points assigned to that cluster.
- **Repeat:** Repeat the assignment and update steps until the centroids no longer change significantly.

What is the math behind it?

The algorithm minimizes the within-cluster variance, which is the sum of squared distances between each point and its corresponding centroid. This is known as the **inertia** or **within-cluster sum of squares (WCSS)** and is given by the formula:

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

where

- k is the number of clusters,
- C_i is the set of points in cluster i,
- x is a data point in cluster C_i,
- μ_i is the centroid of cluster C_i,
- $\| x - \mu_i \|$ is the Euclidean distance between point x and centroid μ_i.

An important point

The algorithm iterates through the assignment and update steps to minimize the WCSS, which effectively makes the clusters as compact and separated as possible.

Brief explanation of each hyperparameter for K-Means in scikit-learn:

- **n_clusters**: *default=8.* The number of clusters and centroids to form. It's the 'k' in k-means.
- **init**: default='k-means++'. Method for initialization of centroids. Options include 'k-means++' for faster convergence, 'random' for random initialization, or a custom method.
- **n_init**: *default='auto'.* Number of times the algorithm will run with different centroid seeds. The best output in terms of inertia is chosen.
- **max_iter**: *default=300*. Maximum number of iterations for a single run of the algorithm.
- **tol**: *default=1e-4.* Tolerance for declaring convergence based on the Frobenius norm of the difference in cluster centers.

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Hyperparameter in K-means above

In the actual lecture slides, K means examples are shown.

Validation Metrics for Clustering

• Silhouette Coefficient • Calinski-Harabasz Index • Davies-Bouldin Index

Silhouette Coefficient

Measures how similar an object is to its own cluster compared to other clusters. A higher silhouette value indicates better-defined clusters.

- **a:** Mean distance between a sample and all other points in the same cluster.
- **b:** Mean distance between a sample and all other points in the next nearest cluster.
- **Formula:** The coefficient for a sample is computed as

$$\frac{b - a}{\max(a, b)}$$

- **Score:** The silhouette score of a dataset is the mean of the silhouette coefficient for each sample, calculated using

```
sklearn.metrics.silhouette_score(X, labels)
```

This metric is useful for assessing the appropriateness of the clustering by measuring how well each object lies within its cluster.

A brief explanation of some of the parameters for silhouette_score

UNIVERSITY OF CALGARY

- **X:** This is either the array of pairwise distances between samples or a feature array from which distances can be computed.
- **Labels:** These are the predicted labels for each sample, indicating which cluster each sample belongs to.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

The Silhouette Coefficient is a measure of how well-separated the clusters are in a clustering result. It takes values between -1 and 1, where:

- A high positive value close to 1 indicates that the clusters are well-separated and the assignment of data points to clusters is appropriate.
- A value around 0 suggests overlapping clusters or clusters that are too close to each other.
- A negative value indicates that data points might have been assigned to the wrong cluster.

In our specific example, a Silhouette Coefficient of 0.5577 suggests a **reasonably good** separation between clusters. It indicates that the clustering result is meaningful and the data points within each cluster are relatively close to each other compared to points in other clusters.

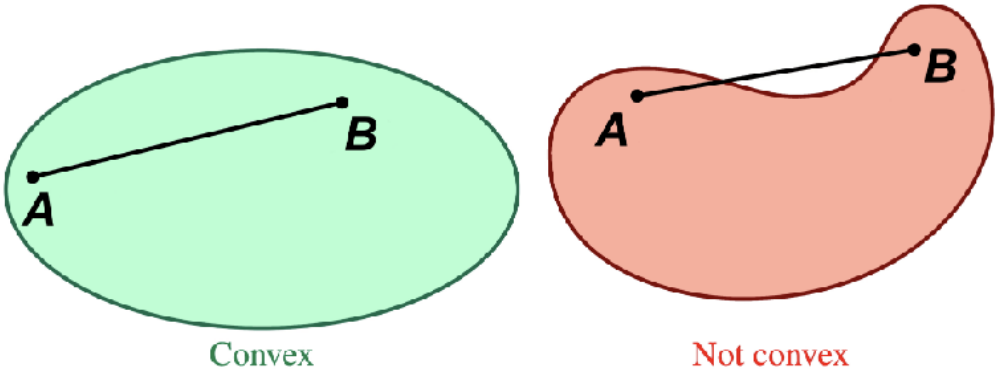
Advantages of Silhouette Coefficient

• Defined Range: The Silhouette Coefficient scores range from -1 to +1. A score of -1 denotes poor clustering, whereas +1 indicates highly compact clusters. • Indication of Overlap: Scores near 0 indicate overlapping clusters, which means that data points might not be clearly categorized. • Measure of Separation and Density: A higher score reflects that clusters are well-separated and densely packed, which is consistent with the standard definition of a cluster.

The Silhouette Coefficient is a metric used to evaluate the effectiveness of clustering outcomes.

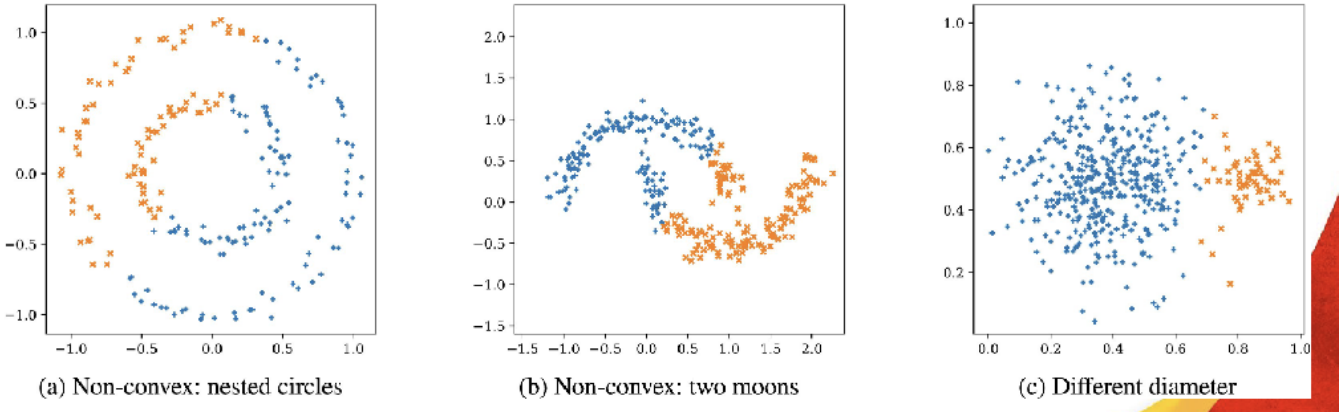
Convex vs Non-Convex

In mathematics, convexity refers to a shape where **any line** segment connecting two points within the shape falls entirely inside the shape itself.



Why is this important? because of datasets that look like this. Which are actually drawbacks of the Silhouette

- **Preference for Convexity:** The Silhouette Coefficient is generally higher for **convex clusters** than for other concepts of clusters, such as density-based clusters like those obtained through DBSCAN



Calinski-Harabasz Index

- The index is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all clusters (where dispersion is defined as the sum of distances squared)
- A higher Calinski-Harabasz index relates to a model with better defined clusters
- It is also known as the ***Variance Ratio Criterion.***

```
sklearn.metrics.calinski_harabasz_score(X, labels)
```

In our specific example, a Calinski-Harabasz Index of 332.5601 suggests a relatively good clustering result. A higher index indicates better-defined and more compact clusters. It implies that the variance between clusters is higher compared to the variance within clusters, reinforcing the quality of our clustering solution.

Advantages of Calinski-Harabasz Index

• The score is fast to compute

• The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

Drawbacks of Calinski-Harabasz Index

• The Calinski-Harabasz index is generally higher for convex clusters than for other concepts of clusters, such as density-based clusters like those obtained through DBSCAN.

Davies-Bouldin Index

- This index signifies the average ‘similarity’ between clusters, where the similarity is a measure that compares the distance between clusters with the size of the clusters themselves
- Zero is the lowest possible score
- Values closer to zero indicate a better partition

```
sklearn.metrics.davies_bouldin_score(X, labels)
```

The clustering result has a Davies-Bouldin Score of 0.6643. This metric shows how good the clustering is, with lower scores meaning better and more distinct clusters. The score of 0.6643 indicates a fair level of cluster density and separation, which quantifies how well the clustering algorithm works on the data.

Advantages of Davies-Bouldin Index

• The computation of Davies-Bouldin is simpler than that of Silhouette scores • The index is solely based on quantities and features inherent to the dataset as its computation only uses point-wise distances

Drawbacks of Davies-Bouldin Index

• The Davies-Boulding index is generally higher for convex clusters than for other concepts of clusters, such as density-based clusters like those obtained from DBSCAN • The usage of centroid distance limits the distance metric to Euclidean space

---

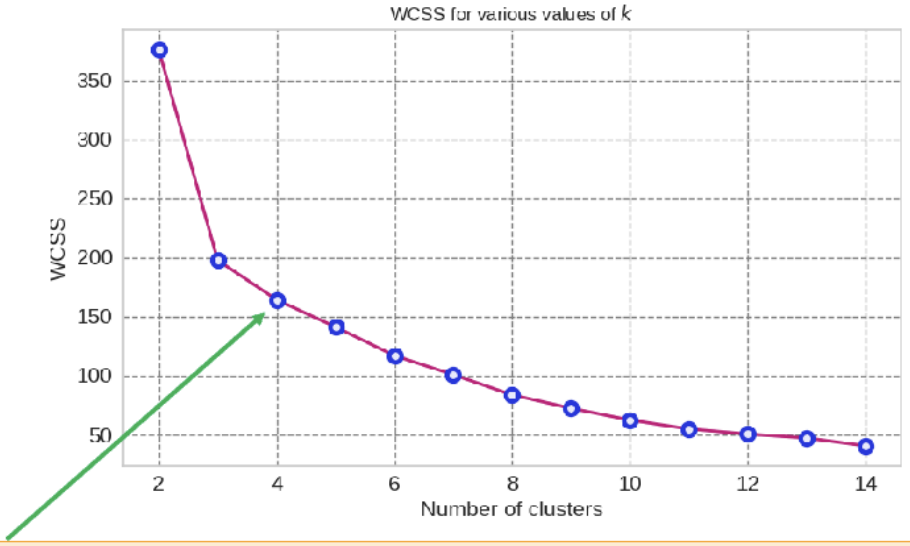
How to estimate the Number of Clusters

- **Elbow Method:** It involves plotting the within-cluster sum of squares (also called inertia) against the number of clusters and identifying the point where the curve changes its slope significantly, creating an "elbow" shape. This point generally indicates the optimal number of clusters, as adding more clusters beyond this point doesn't lead to a significant decrease in inertia.
- **Validation Metrics:** These metrics, such as the Silhouette Coefficient, the Calinski-Harabasz Index, or the Davies-Bouldin Index, measure how well the data points are grouped into clusters. We can plot these metrics against the number of clusters and choose the value that maximizes or minimizes the metric, depending on its definition. Ideally, the optimal value should match the elbow point.

Estimating the Number of Clusters (1/)

UNIVERSITY OF CALGARY

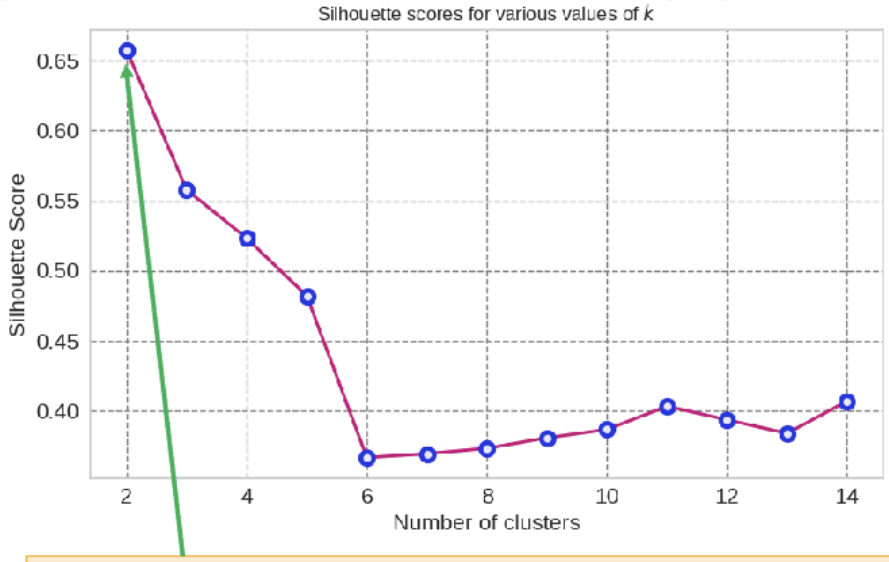
k	WCSS
2	375.911417
3	197.789555
4	164.328437
5	141.301594
6	117.069627
7	100.884894
8	83.837141
9	72.514324
10	62.471259
11	54.943982
12	50.621097
13	47.368983
14	40.688909



k = 4 seems to be a good choice, as the WCSS decreases slowly after that point.

Estimating the Number of Clusters (2/)

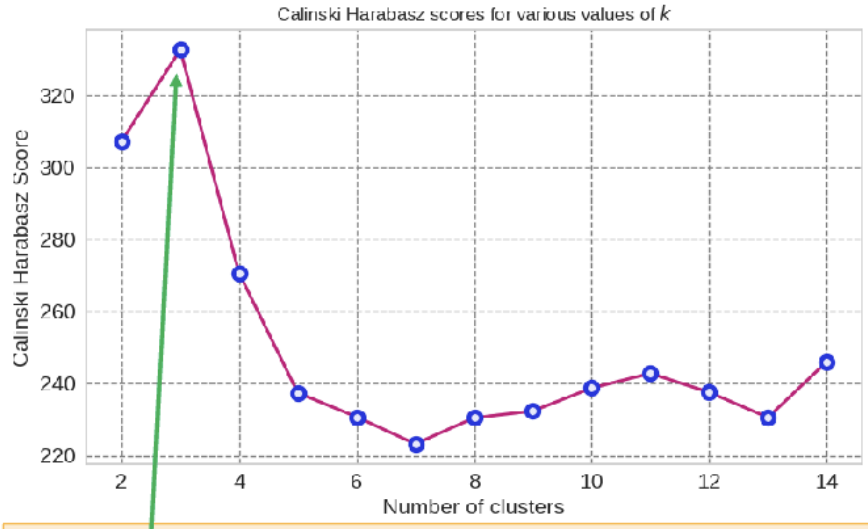
k	Silhouette Score
2	0.657254
3	0.557686
4	0.523192
5	0.482150
6	0.366941
7	0.369662
8	0.373564
9	0.381007
10	0.386978
11	0.403531
12	0.393846
13	0.384175
14	0.407270



If we pick k based on the highest Silhouette Score, then it would be 2.

Estimating the Number of Clusters (3/)

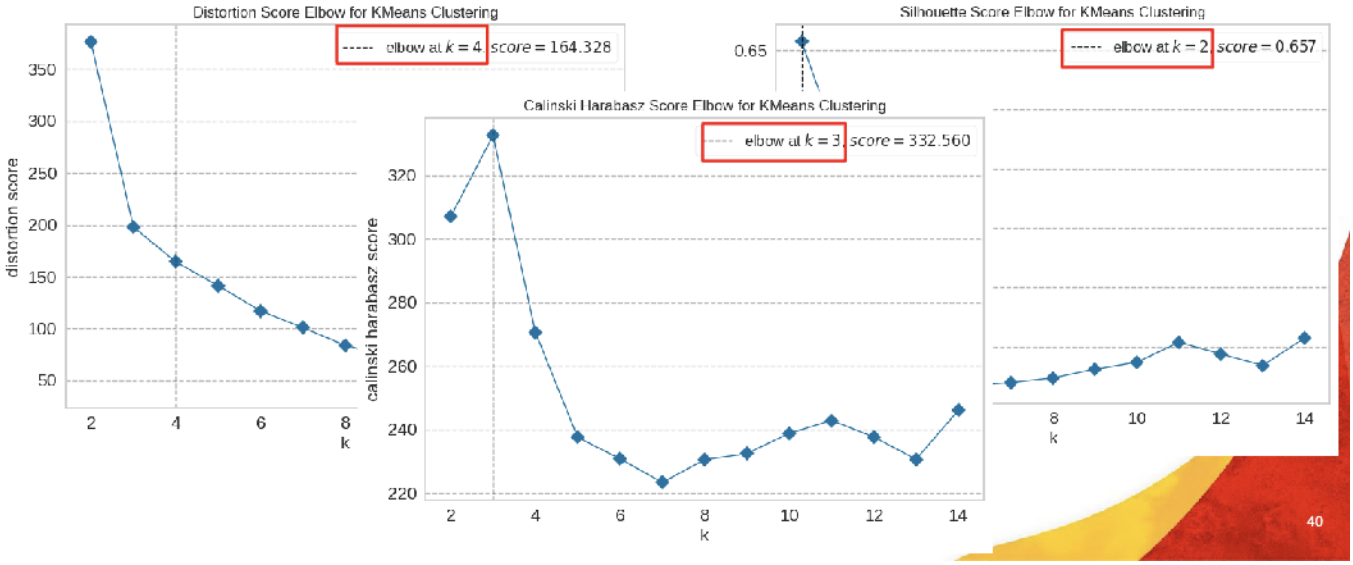
k	Calinski Harabasz Score
2	307.131184
3	332.560118
4	270.616335
5	237.449094
6	230.756409
7	223.259505
8	230.474621
9	232.396379
10	238.756737
11	242.824213
12	237.591621
13	230.597759
14	246.043261



If we pick k based on the highest Calinski Harabasz Score, then it would be 3.

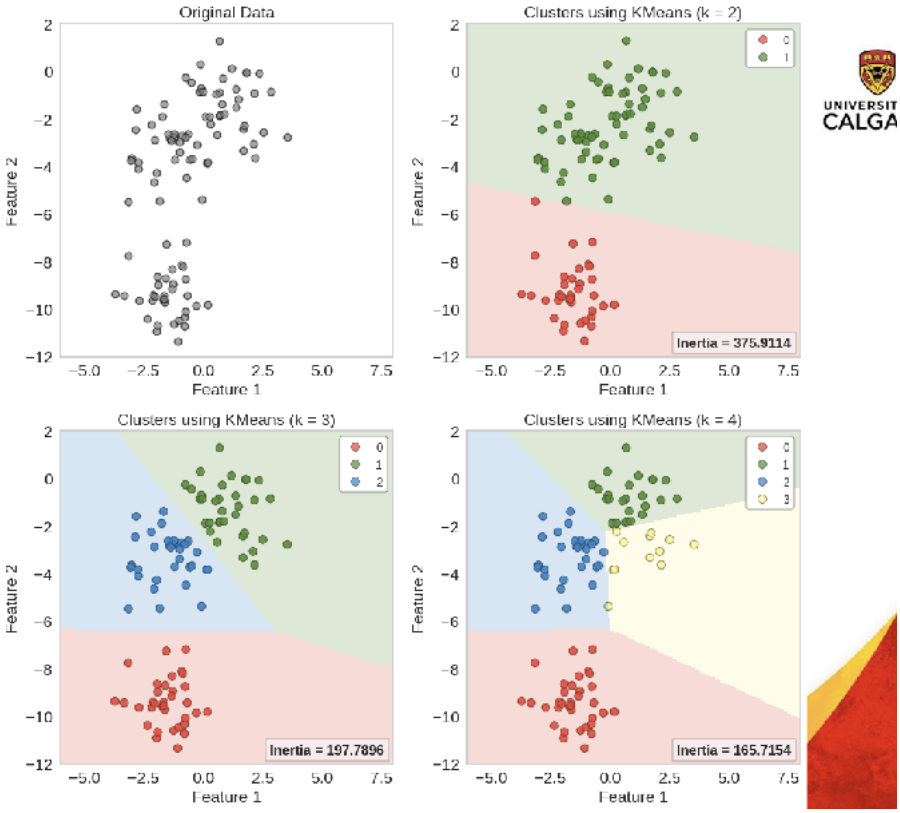
Estimating the Number of Clusters (4/)

We also could this using kelbow_visualizer from Yellowbrick liberary.



Estimating the Number of Clusters (5/)

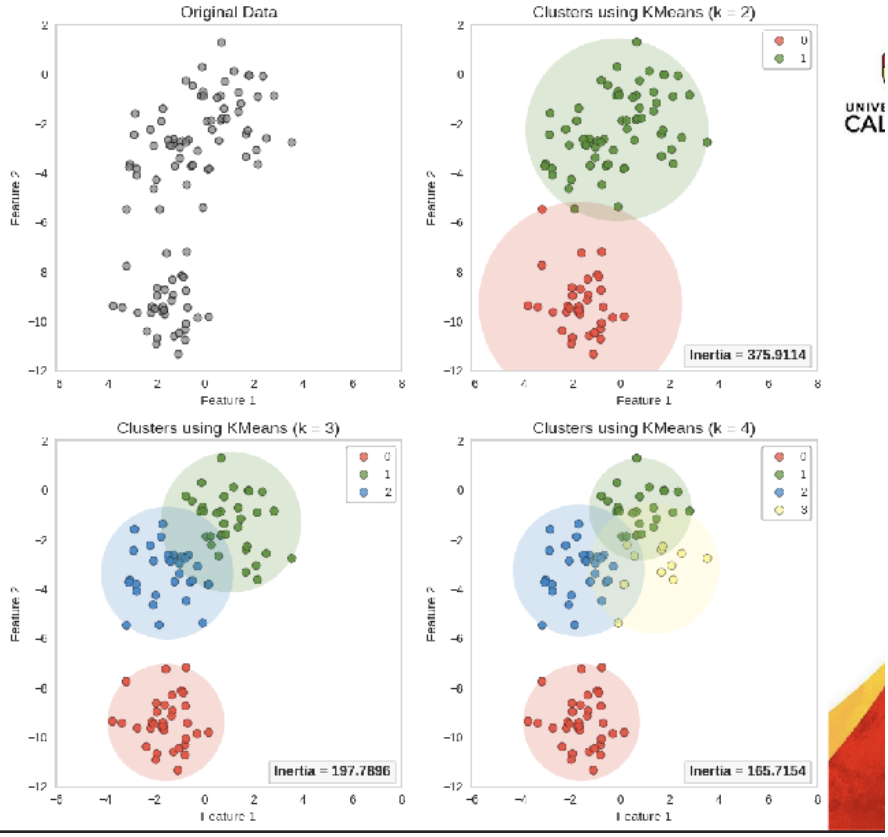
Let's use the above suggested values for **k**.



Estimating the Number of Clusters (6/)

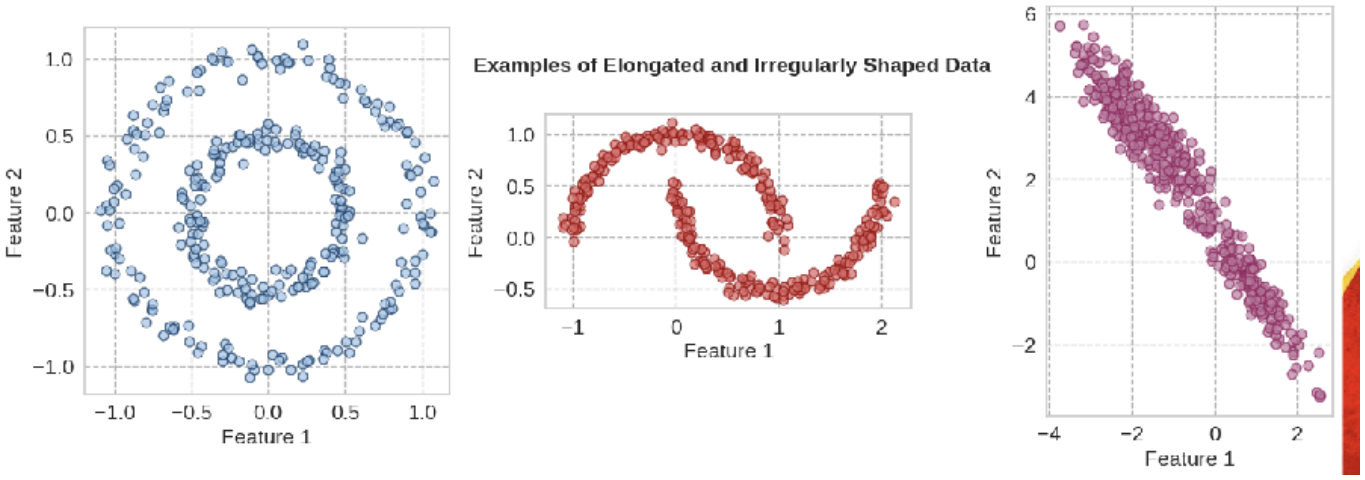
Let's use the above suggested values for **k**.

Alternative presentations: The circles' radii represent the distances from cluster centroids to the furthest points.



Drawbacks of Inertia

• Inertia assumes that clusters are convex, which is not always the case • It responds poorly to elongated clusters, or manifolds with irregular shapes



• Inertia is not a normalized metric: we just know that lower values are better and zero is optimal
• In very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called "curse of dimensionality")
• Running a dimensionality reduction algorithm such as Principal component analysis (PCA) prior to k-means clustering can alleviate this problem and speed up the computations

1. **Inertia:** Inertia is a measure of how spread out the clusters are in k-means clustering. Lower values are better, and zero is the best. We don't have a fixed scale for inertia, but we aim to minimize it.

2. **Curse of Dimensionality:** In high-dimensional spaces, distances between points can become distorted, making it harder to find meaningful clusters. This makes clustering less reliable and more computationally intensive.

3. **PCA Dimensionality Reduction:** PCA is a technique that reduces the number of features (dimensions) in the data while preserving its important information. Using PCA before clustering can help mitigate the effects of the curse of dimensionality and speed up the clustering process.