3.4 Clustering #Clustering

3.4.1 Intro

With **Clusterig**, a set of **observations** (dataset) is **split** into different **segments** that have the following properties:

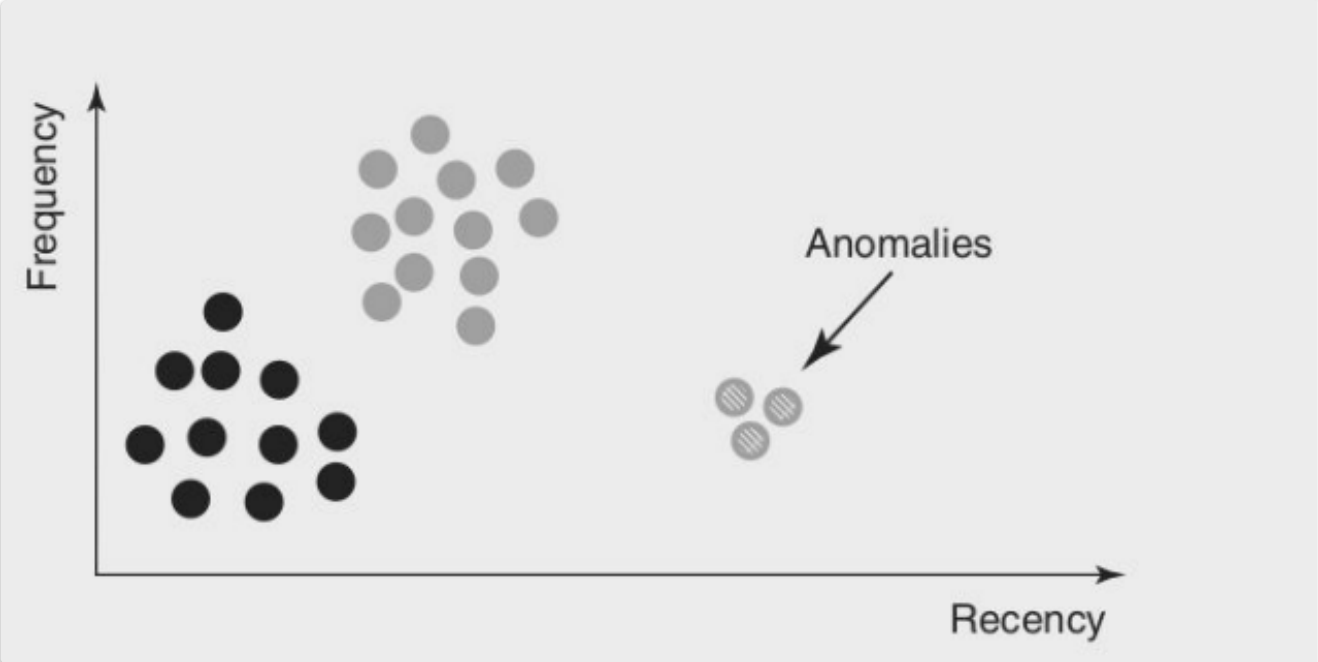
- **Homogeneity** within a **segment** is **maximized**;
- **Heterogeneity** between **segments** is **maximized**.

The data can be:

- Structured
- Unstructured

> ***Avoid*** *excessive amounts of* ***correlated data*** *by applying feature selection methods.*

3.4.2 Clustering for Fraud Detection



When using clustering for fraud detection, these techniques can be identified:

- Hierarchical:
  1. Agglomerative;
  2. Divisive.
- Non-hierarchical:
  1. k-means;
  2. Self Organizing Map (SOM).

Distance Metrics #DistanceMetrics

Since the aim of clustering is grouping observations based on similarity, a distance metric is needed to quantify this similarity.

Simple distance metrics:

- Euclidean Distance
- Minkowski distance
  Formula: $D(x_i; x_j) = (\sum_{k=1}^{n} |x_{ik} - x_{jk}|^p)^{\frac{1}{p}}$
  if p=1 ==> Manhattan or City block distance
  if p=2 ==> Euclidean distance

Techniques based on types of variables:

- **Continuous Variables:**
  1. Euclidean metric;
  2. Pearson correlation;
  3. Cosine measure.
- **Categorical variables:** With binary variables two techniques:
  1. Simple Matching Coefficient (SMC) = calculates the number of identical matches between the variable values (equally important).
  2. Jaccard index = measures the similarity between both claims across those red flags that were raised at least once.
- **Continuous and Categorical Mix:** complicates the computation of the distance. Two options:
  1. Code the categorical variables as 0/1 dummies and then use a Continuous distance measure;
  2. Use a weighted combination of distance measures, which is less straightforward less frequently used

3.4.3 Hierarchical Techniques #HierarchicalTechniques

Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Two types can be identified:

- #Agglomerative **Agglomerative**: Bottom-Up approach, each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy;
- #Divisive **Divisive** : Top-Down approach, all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

Linkeage #Linkeage

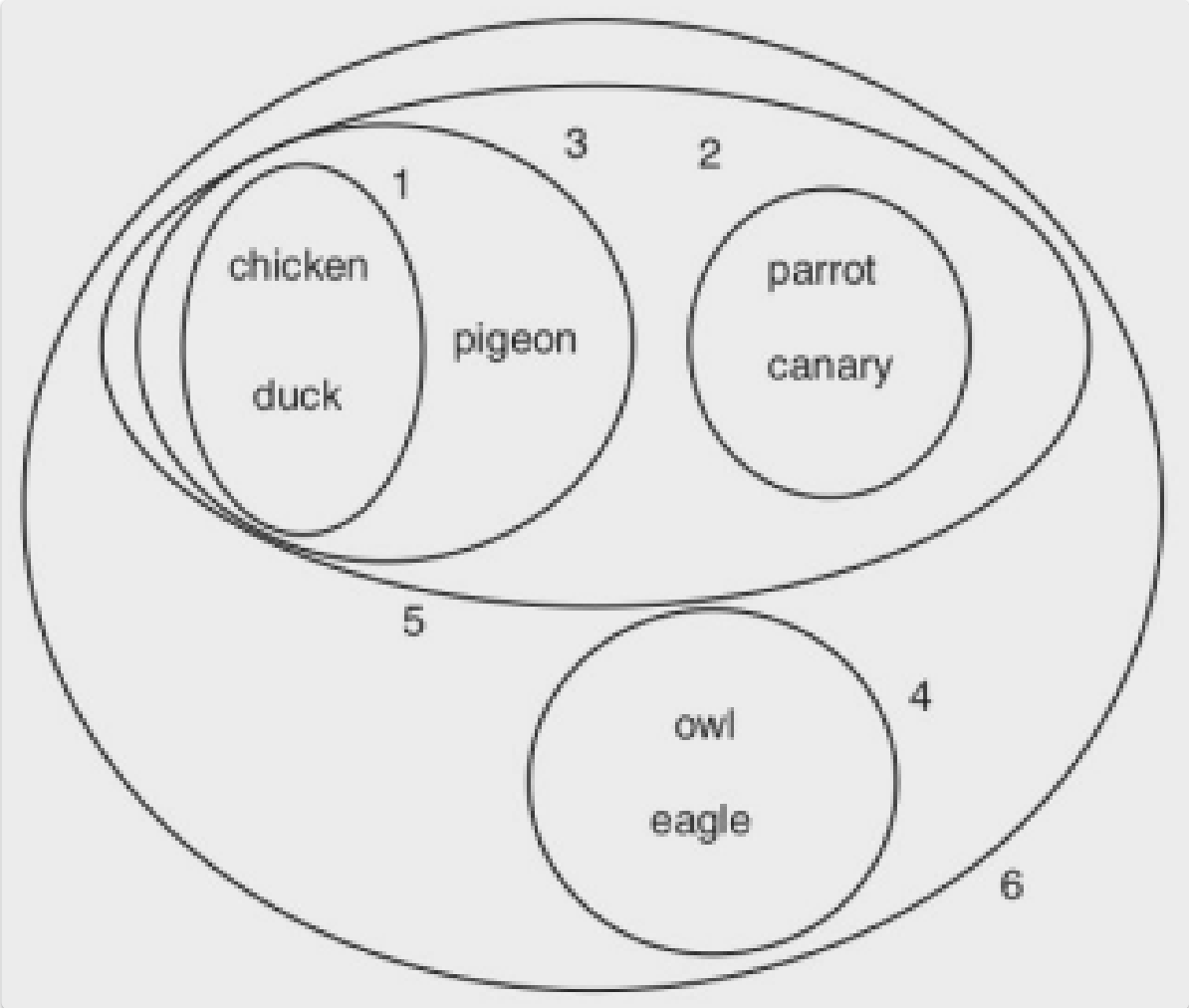
Distance between clusters can be measured in different ways, such as:

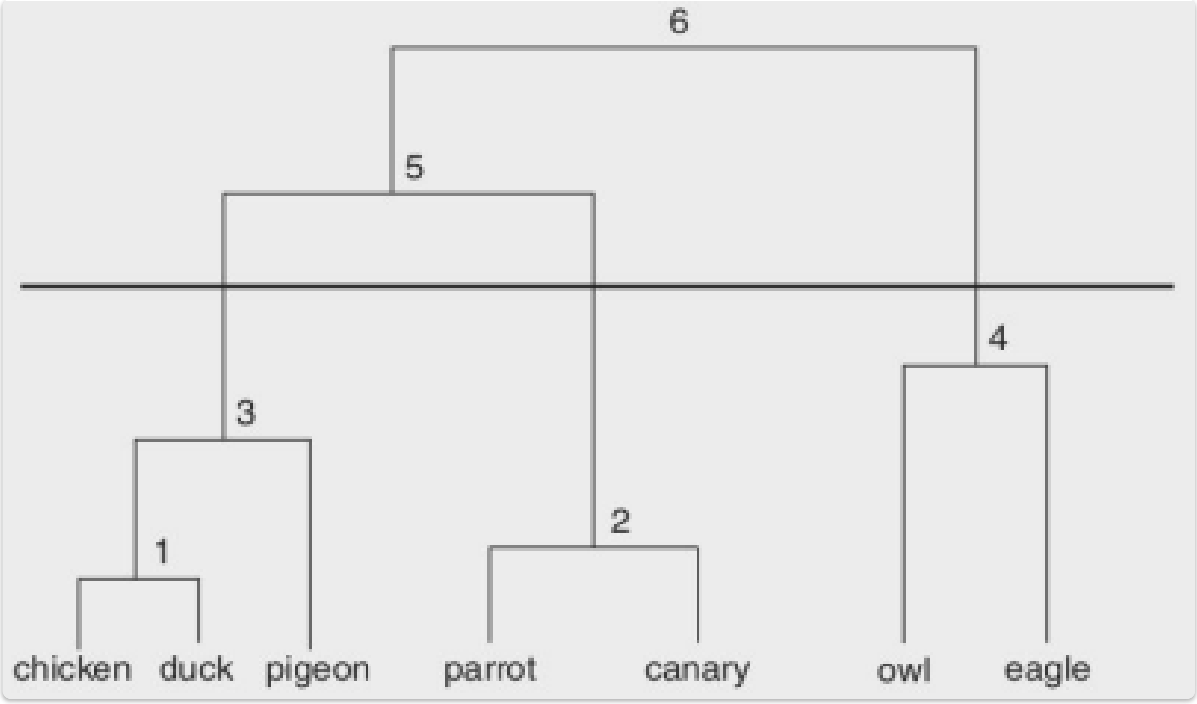
- #SingleLinkeage **Single linkeage:** Minimum distance between two points in the two clusters;
- #CompleteLinkeage **Complete linkeage:** Maximum distance between two points in the two clusters;
- #AverageLinkeage **Average linkeage:** Average distance between all the points in the two clusters;
- #CentroidLinkeage **Centroid linkeage:** Distance between the two center points in the two clusters;
- #WardCriterion **Ward's criterion:**
  $D_{Ward}(C_i; C_j) = \sum_{x \in C_i}(x - c_i)^2 + \sum_{x \in C_j}(x - c_j)^2 - \sum_{x \in C_ij}(x - c_ij)^2$;

Number Of Clusters #NumberOfClusters

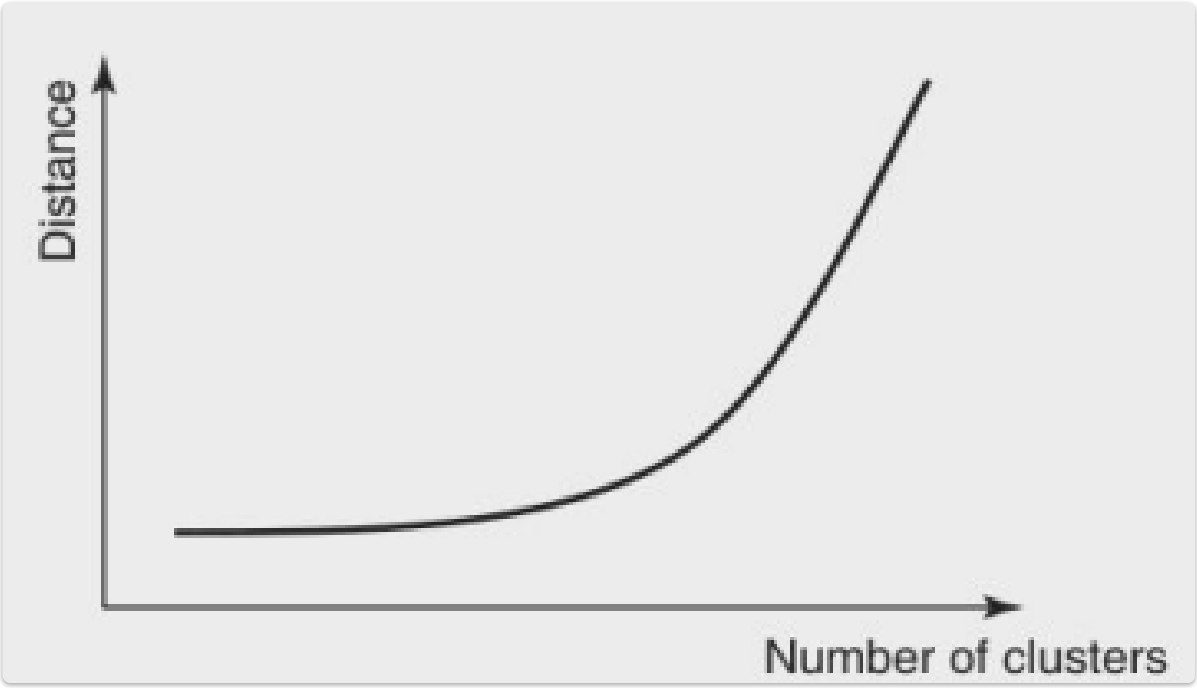
To decide on the optimal number of clusters:

- #Dendrogram **Dendrogram:**
  1. Tree-like diagram that records the sequences of merges;
  2. Vertical (or horizontal scale) gives the distance between two clusters amalgamated;
  3. Cut the dendrogram at the desired level to find the optimal clustering.





- #ScreenPlot **Screen plot:**
  1. Plot of the distance at which clusters are merged;
  2. The elbow point then indicates the optimal clustering.



Conclusions

🟢 Advantages:

1. The *number of clusters* does *not* need to be *specified prior* to the analysis.

🔴 Disadvantages:

1. *Doesn't scale* well with large datasets;
2. The *interpretation* of the clusters is often *subjective* and depends on the business expert and/or data scientist.

3.4.4 Non-Hierarchical Techniques #Non-HierarchicalTechniques

Non-Hierarchical Clustering can be subdivided in two techniques:

- #K-Means ***k*-means**: method that aims to *partition n observations into k clusters* in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid).
- #SOM **Self Organizing Map (SOM):**

3.4.4.1 K-Means

K-Means technique consists of four steps:

1. Select ***k* observations** as **initial** cluster **centroids** (seeds);
2. **Assign** each **observation** to the **cluster** that has the **closest centroid** (e.g., Euclidean distance);
3. When **all observations** have been **assigned**, **recalculate** the **positions** of the ***k* centroids** (mean);
4. **Repeat until** the cluster **centroids no** longer **change or** a **fixed number** of iterations is **reached**.

LIMITATIONS

A k-means approach presents a number of limitations:

🔴 The number of clusters k needs to be specified before the start of the analysis. The number of clusters can be determined in four ways:

1. Expert based input;
2. Result of another (e.g., hierarchical) clustering procedure
3. Multiple values of *k* are tried out and the resulting clusters evaluated;
4. Try out different seeds to verify the stability of the clustering solution.

🔴 *k-means* is sensitive to outliers, which are especially relevant in a fraud detection setting.
More robust alternatives:

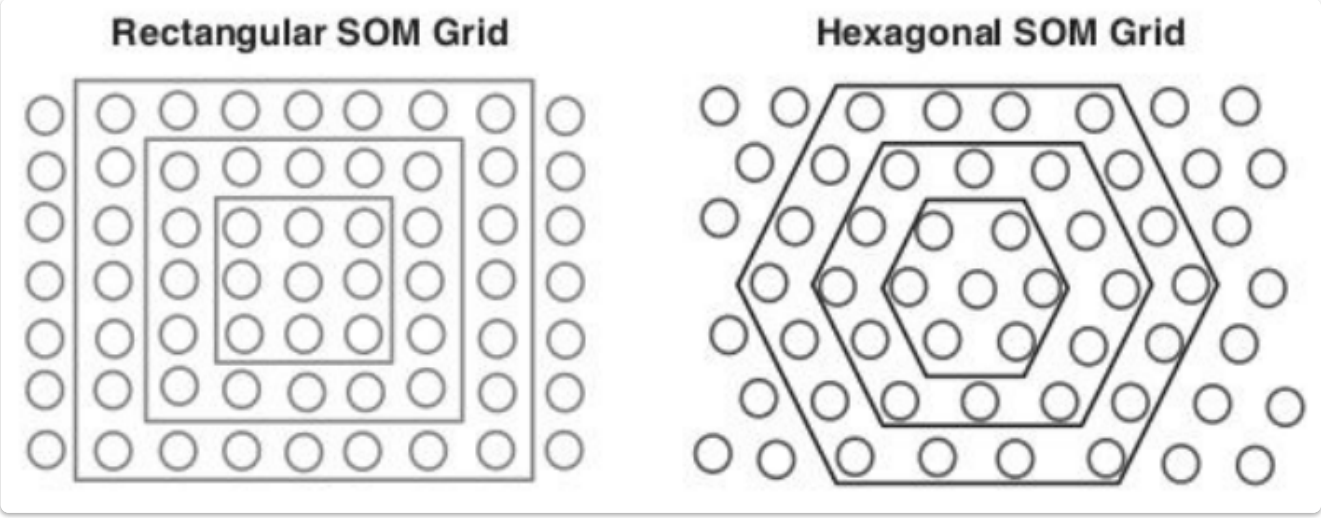
1. Use the median (*k-medoid clustering*);
2. For categorical variables, the mode can be used (*k-mode clustering*).

3.4.4.2 Self Organizing Map

#DEF **SOM** is a clustering technique that **employs** an **unsupervised learning algorithm** that allows users **to visualize** and **cluster high-dimensional data on** a **low-dimensional grid** of **neurons**.

Two types of grids:

- Rectangular SOM Grid;
- Hexagonal SOM Grid.



Functioning

Each input is connected to all neurons in the output layer with weights $w = [w_1, \ldots, w_N]$, $N =$number of variables. All weights are randomly initialized.

When a training vector x is presented, the weight vector of each neuron is compared with x by calculating the distance.
==> The neuron that is most similar to x in Euclidean sense is called the **B**est **M**atching **U**nit (**BMU**).

Euclidean distance: $d(x, w_i) = \sqrt{\sum_{i=1}^{N}(x_i - w_{ci})^2}$

The weight vector of the **BMU** and its neighbors in the grid are then adapted using the following learning rule:

> $w_i(t+1) = w_i(t) + h_{ci} \cdot [x(t) - w_i(t)]$

where

- t represents the time index during training;
- $h_{ci}(t)$ defines the neighborhood of the BMU c.

Next chapter: Semi-supervised clustering