Data-intensive space engineering

Lecture 6

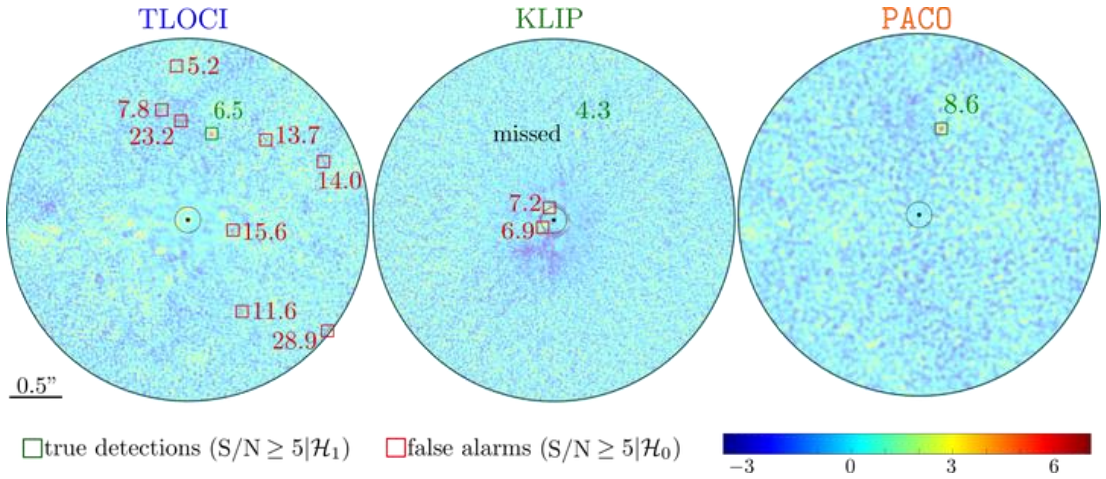
Carlos Sanmiguel Vila

Introduction to Unsupervised

Most available data is unlabeled, i.e., we have the input features X but not the labels y

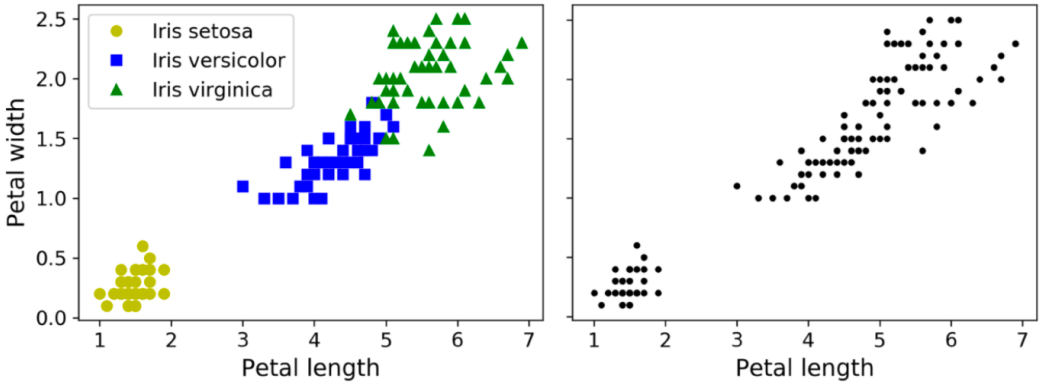
Example of a classical routine:

- Discover new exoplanets by direct imaging
- Take thousands of data every day
- Need to label each observation targeting if an observation is a real candidate for an exoplanet or not to train binary classifiers
- Long, costly, and tedious task



Types of unsupervised learning

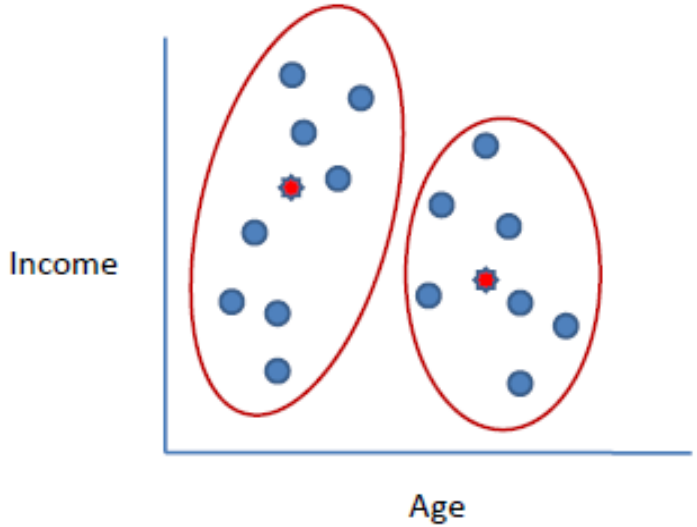
- **Clustering**
  - Group similar instances together into clusters



- **Anomaly detection**
  - Learn “what” normal data looks like and then use it to detect abnormal instances
- **Density estimation**
  - Estimating the probability density function of the random process that generated the data set
- **Dimensionality Reduction**
  - Learning to reduce the dimension of our dataset in an efficient manner

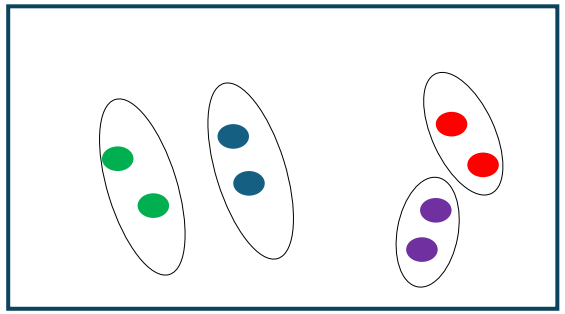
Different clustering algorithms

- There is no universal definition of what a cluster is
  - It depends on the context and different algorithms will capture different kinds of clusters
  - Most typical approach is to look for instances centered around a particular point, called a centroid



Clustering Types

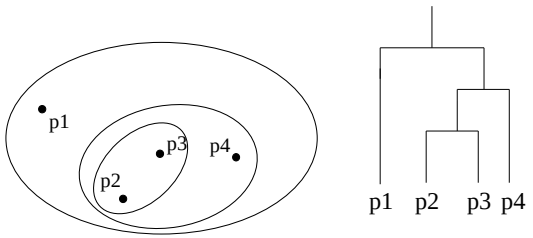
- Partitional clustering: no overlap between clusters



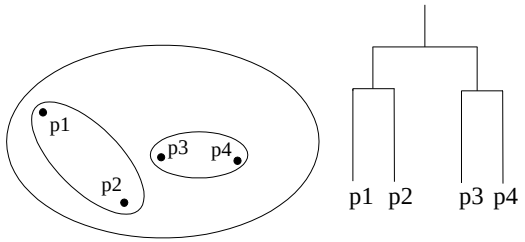
K=4

- Hierarchical clustering: clusters are nested

3 clusters: {p1},{p2,p3}, {p4}



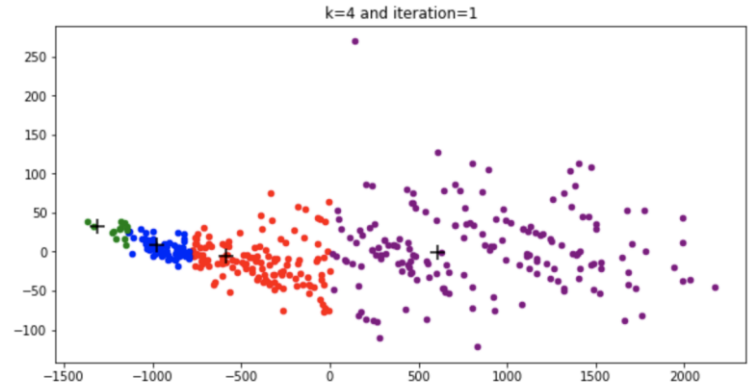
4 clusters: {p1},{p2},{p3}, {p4}



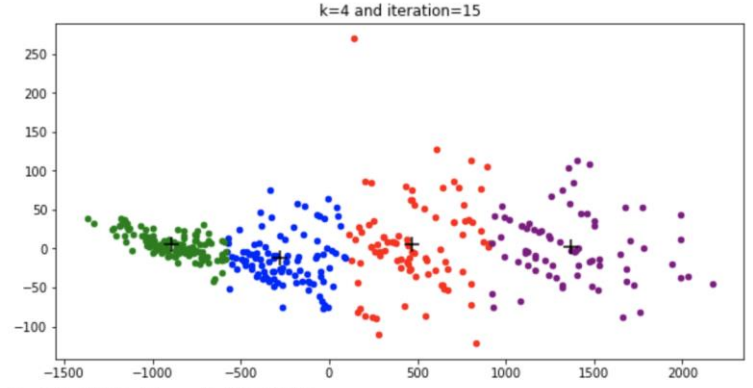
K-means

- Mostly widely used partitional clustering approach
- Simple and efficient
- Design:
  - Input: a collection of data records, the number of clusters, i.e., K
  - Process:
    1. Randomly choose K points as the initial centroids,
    2. Generate K clusters by assigning all points to the closest centroid
    3. Recompute the centroid of each cluster
    4. Repeat steps 2, 3 until the centroid don’t change

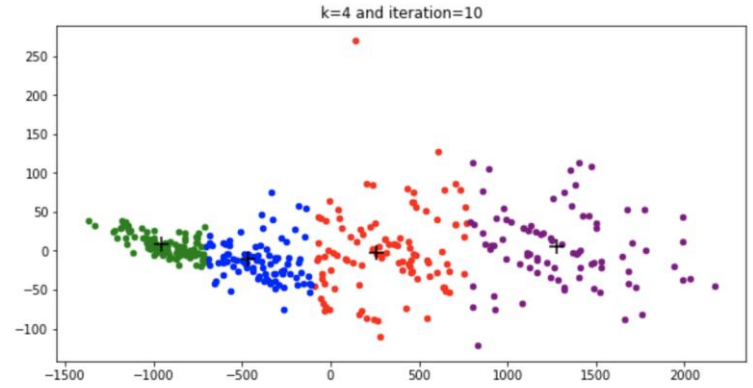
K-means



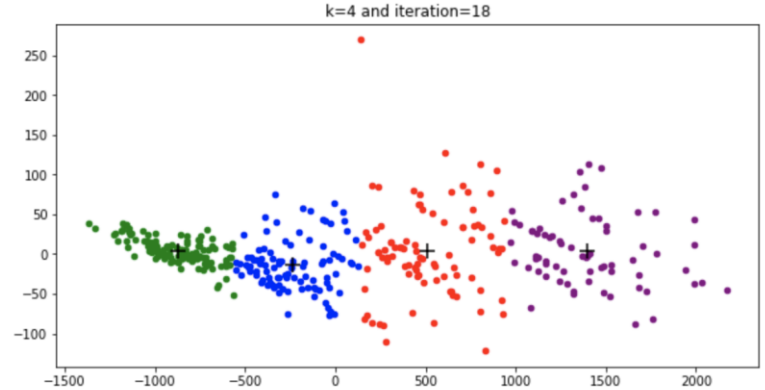

```
[[ -589.63268924    -5.74874363]
 [ -979.32872141     9.14819031]
 [-1311.06603935    31.84574949]
 [  610.759587      -1.36753344]]
```




```
[[ 469.21229583    5.24834915]
 [-277.82761308  -12.70795153]
 [-896.62628601    5.18789866]
 [1367.52838428    2.54325882]]
```




```
[[ 262.03592093   -2.06404016]
 [-471.04181627  -10.052115  ]
 [-957.82841684    8.07467308]
 [1274.80443573    4.83840174]]
```




```
[[ 511.18816227    4.51160068]
 [-234.43461798  -13.04519014]
 [-875.77791537    4.49084539]
 [1399.77742914    3.85711884]]
```

K-means

Advantages:

- Easy to implement and understand.
- Efficient for large datasets with moderate dimensionality.

Disadvantages:

- Sensitive to initial centroid placement (can get stuck in local minima).
- Requires specifying K (number of clusters), which may not always be known beforehand.
- Assumes clusters are spherical and equally sized, which may not hold in real-world applications.

K-means

- **Silhouette Coefficient**:

  • This measures how similar a data point is to its own cluster compared to others. It ranges from **-1 to 1**:

  - **+1**: The point is well-matched to its own cluster and far from others.
  - **0**: The point is on or very close to the decision boundary between two clusters.
  - **-1**: The point might have been misclassified to the wrong cluster.

  The **Silhouette Coefficient** helps evaluate the clustering quality and choose the optimal number of clusters (**K**).

K-means

- The **Elbow Method** is a heuristic used to determine the optimal number of clusters K by measuring the inertia (or within-cluster sum of squared errors (SSE)). Inertia represents how tightly the clusters are packed.

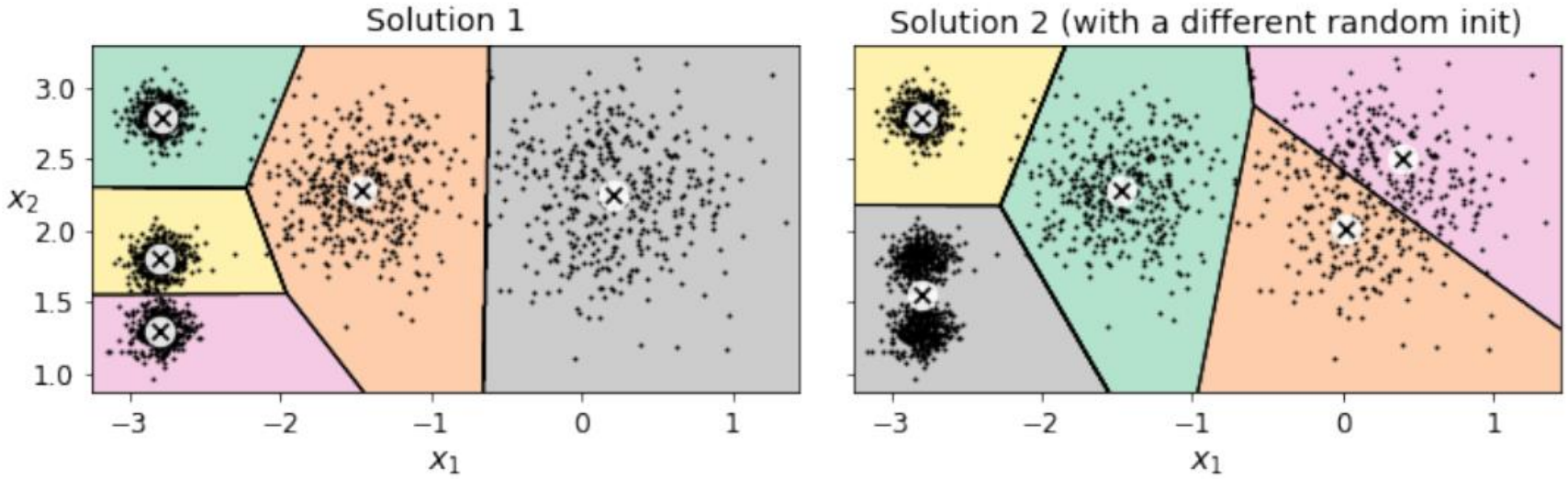
- The idea behind the Elbow Method is to run K-Means (or any clustering algorithm) for different values of K and calculate the total SSE (the sum of squared distances between each point and the centroid of its assigned cluster). As K increases, the SSE decreases because the clusters are smaller and better fitted to the data. However, increasing K results in diminishing returns at a certain point, which can be identified by an "elbow" in the plot.

Steps:

- Run K-Means for a range of K values.
- Calculate the SSE for each K.Plot K vs. SSE.
- Identify the "elbow point" where the rate of decrease slows down, suggesting an optimal K.

K-means

- Although the algorithm is guaranteed to converge, it may not converge to the right solution
  - Highly sensitive to the centroid initialization



K-means

- **How do we deal with this issue?**

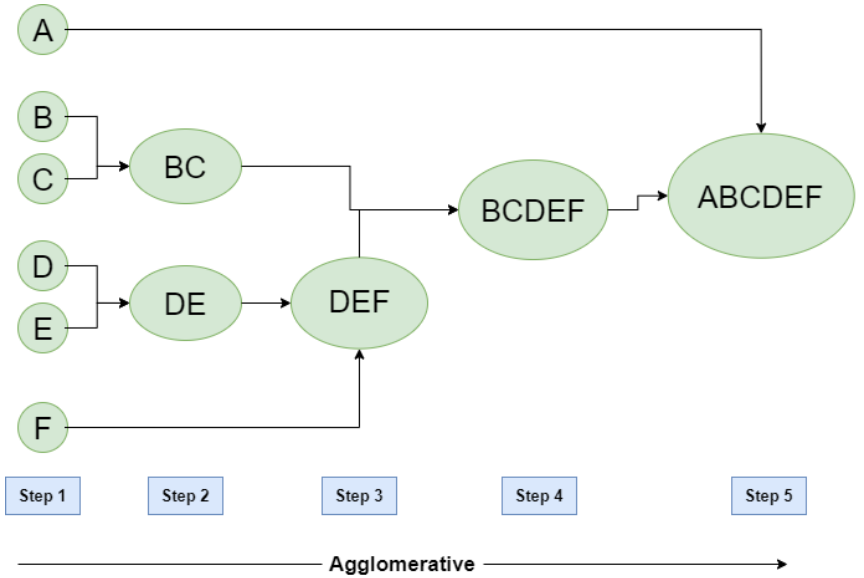
  - **k-means++ Initialization**: A smarter initialization method that selects initial centroids that are far apart, improving convergence and cluster quality. By default, scikit-learn's KMeans uses k-means++ initialization.

  - **Multiple Initializations (n_init)**: Running K-Means multiple times with different random starts and choosing the best result reduces the risk of poor initialization. The algorithm runs K-Means multiple times with different initializations (based on the number of n_init), and then selects the solution with the lowest within-cluster sum of squares (SSE).

  - **Data Scaling:** Ensures that features with larger values do not dominate distance calculations, improving initialization and clustering results.

Hierarchical Clustering

- Hierarchical clustering creates a hierarchy of clusters that can be visualized as a **dendrogram**. It is a bottom-up approach (agglomerative), where each data point starts as its own cluster, and pairs of clusters are merged at each step based on their similarity.
  - Start with each data point as its own cluster.
  - At each iteration, merge the closest clusters.
  - Continue merging until only one cluster remains.
- **Linkage Criteria**:
  - **Single Linkage**: Distance between the closest points of clusters.
  - **Complete Linkage**: Distance between the farthest points of clusters.
  - **Average Linkage**: Average distance between points in clusters.
  - **Ward's Method**: Minimizes the variance within clusters, often leading to better-defined clusters.



Hierarchical Clustering

Advantages:
- No need to specify the number of clusters.
- Provides a full hierarchy of clusters (can choose any level of granularity).

Disadvantages:
- Computationally expensive for large datasets.
- Sensitive to noise and outliers.

Best Types of Linkage Methods:
•**Single Linkage**: Clusters with irregular shapes, but can result in "chaining" where points get stretched into long chains.
•**Complete Linkage**: Compact, well-separated clusters.
•**Average Linkage**: Balanced, spherical clusters.
•**Ward's Method**: Producing clusters of relatively equal size.

Tip:
•**Ward's Method** is often a good default choice, especially for aerospace applications where clusters are expected to be compact and well-separated (e.g., grouping satellite telemetry phases or system states).

DBSCAN

This algorithm defines clusters as continuous regions of high density

• For each instance, it counts how many samples are located within a small distance ϵ from it (ϵ -neighborhood)

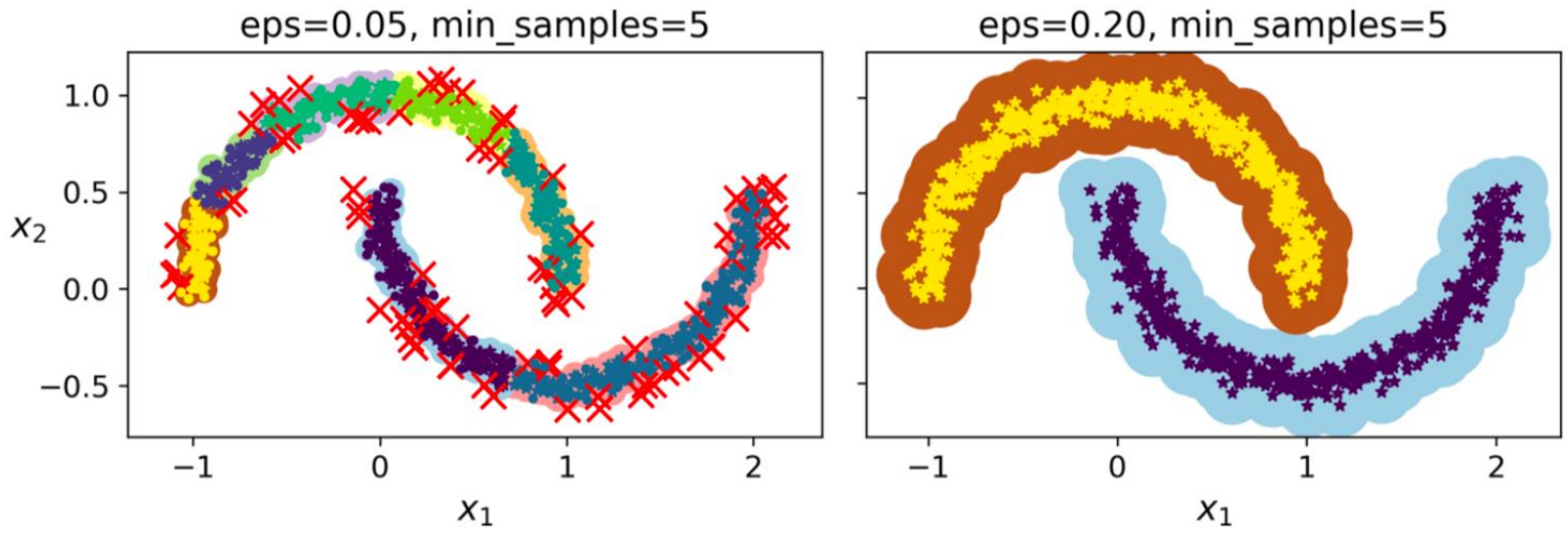
• If an instance has at least min_samples samples in its -neighborhood, then it is considered a “core instance”

• All instances in the neighborhood of a core instance belong to the same cluster

• Any instance that is not a “core instance” and does not have one in its neighborhood is considered an anomaly

DBSCAN

DBSCAN clustering using two different neighborhood radiuses



- DBSCAN can identify any number of clusters of any shape
- Robust to outliers
- If the density varies significantly, it may be impossible to capture all clusters

What Is Principal Component Analysis?

An unsupervised method that takes X from p dimensions down to k dimensions. PCA is a linear dimensionality reduction technique. It identifies the directions (principal components) in which the data varies the most and projects the data into those directions, reducing the number of features.

Why is this helpful?

- Reduces noise in the data for supervised learning
- Reduces data
- Simpler to visualize data (though dimensions may be unintuitive!)

The Covariance Matrix

Consider some feature $\mathbf{X^{(a)}}$, then we can compute the **variance** (*std deviation squared*) of this feature :

$$var\left(X^{(a)}\right) = (s^{(a)})^2 = \frac{\sum_{i=1}^{n}\left(X_i^{(a)} - \overline{X^{(a)}}\right)^2}{(n-1)}$$

and we can generalize for *1* <= *a* <= *p* to get a variance matrix

Similarly, we can define **covariance** to capture how the dimensions vary from the mean with respect to each other

$$cov\left(X^{(a)}, X^{(b)}\right) = \frac{\sum_{i=1}^{n}(X_i^{(a)} - \overline{X^{(a)}})(X_i^{(b)} - \overline{X^{(b)}})}{n-1} = E[\left(X^{(a)} - \overline{X^{(a)}}\right)\left(X^{(b)} - \overline{X^{(b)}}\right)]$$

Principal Component Analysis

How it Works:

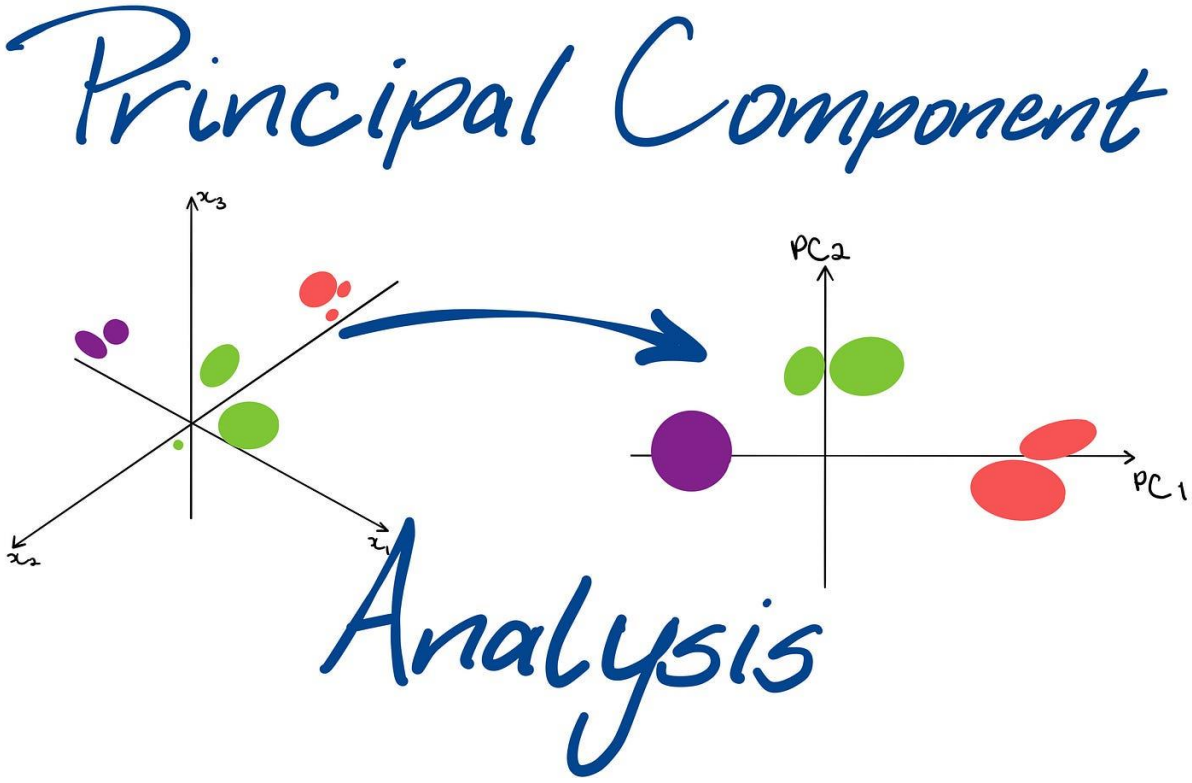
- Compute the covariance matrix of the data.
- Calculate the eigenvectors (principal components) and eigenvalues.
- Sort the eigenvectors by their corresponding eigenvalues in descending order (importance).
- Choose the top k
- Construct a projection matrix W from these k eigenvectors
- Project the original dataset X by multiplying W to obtain a k-dimensional feature subspace Z

$$\boldsymbol{Z_{n\times k} = X_{n\times p} \cdot W_{p\times k}}$$

Principal Component Analysis

Transform to new coordinate system:

- Find directions of maximum variation (covariance)
- Minimize reconstruction error



Principal Component Analysis

Advantages:

- Reduces the dimensionality of data while retaining the most important variance.
- Helps visualize high-dimensional data in 2D or 3D.

Disadvantages:

- PCA is a **linear** method, so it struggles with non-linear datasets.
- The resulting components are often difficult to interpret.

Evaluation:

- **Explained Variance Ratio**: The proportion of the dataset's variance explained by each principal component. A higher variance ratio indicates that more of the data's information is retained.

Isolation Forest

Isolation Forest is an algorithm specifically designed for anomaly detection. Unlike many other algorithms that measure distance or density to detect outliers, it works by isolating observations. It uses the principle that anomalies are few and different, meaning they are easier to isolate than normal points.

How Isolation Forest Works:

•**Isolation**: The basic idea of the Isolation Forest is that anomalies are data points that are "isolated" more quickly than normal points when random cuts are made in the feature space.

•**Random Partitioning**: The algorithm builds an ensemble of **randomly generated decision trees** (called "isolation trees"). At each node in the tree, a feature is randomly selected, and a random split value is chosen between the minimum and maximum value of that feature. This process continues recursively.

•**Path Length**: The number of splits required to isolate a point corresponds to its **path length**. Anomalies are expected to have **shorter path lengths** since they are isolated faster (they are in sparse regions of the data).

•**Ensemble**: The algorithm uses an ensemble of many isolation trees, similar to random forests, to make more robust decisions.

Isolation Forest

Advantages:

•**Effective for Anomaly Detection**: Especially useful for identifying outliers in high-dimensional datasets, making it suitable for aerospace applications like detecting anomalous satellite telemetry data or system malfunctions.

•**No Need for Distance or Density Measures**: Unlike algorithms like DBSCAN or K-Means, Isolation Forest doesn't rely on distance metrics or density estimation, making it effective in scenarios with high-dimensional data.

Disadvantages:

•May require **parameter tuning** (e.g., number of trees, contamination) to optimize performance, although defaults often work well.

•Not as interpretable as some other methods since it's based on random cuts rather than a distance-based approach.

K-Nearest Neighbors (KNN)

- K-Nearest Neighbors (KNN) is a non-parametric and instance-based learning algorithm. It can be used for both classification and regression tasks.
- **For unsupervised learning,** KNN can be adapted for anomaly detection and density estimation. The key idea of KNN is to classify or estimate a data point based on the K nearest neighbours in the feature space.
- **For anomaly detection,** the algorithm identifies points that are far away from their neighbors as outliers.

- **How KNN Works:**
- **Distance-based approach**: The algorithm finds the K nearest neighbors of a given point based on a distance metric, typically Euclidean distance. The distance is used to estimate the density or classify the point.
- **K parameter:** The number of neighbors K plays a crucial role in the performance of the model. A small K is sensitive to noise, while a large K can smooth out important patterns.

K-Nearest Neighbors (KNN)

Advantages

Simple and Intuitive: Easy to understand and implement.Versatile: Works for both classification and anomaly detection in high-dimensional data.

No training phase: KNN is an instance-based learner, meaning the algorithm doesn't need explicit training.

Disadvantages

Computationally expensive: KNN requires calculating distances for every point in the dataset during prediction, which can be slow for large datasets.
Sensitive to scaling: Since KNN is distance-based, it's sensitive to feature scaling. Proper normalization or scaling of the data is essential.