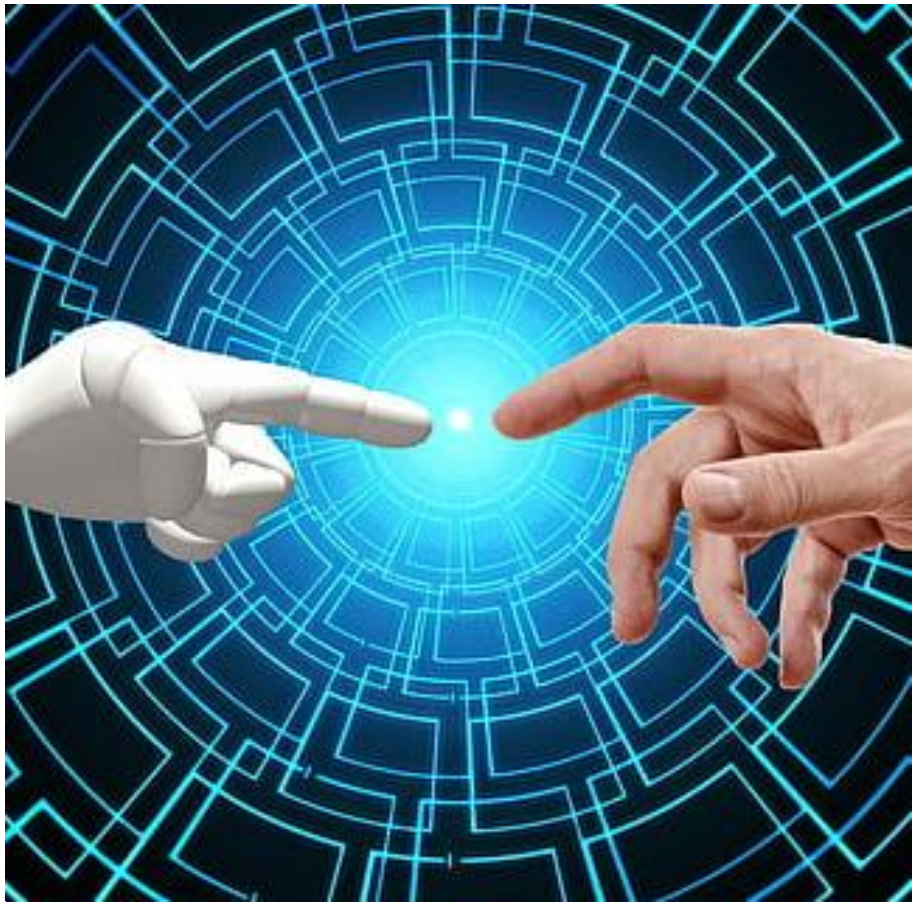
# Artificial Intelligence (AI) for Engineering



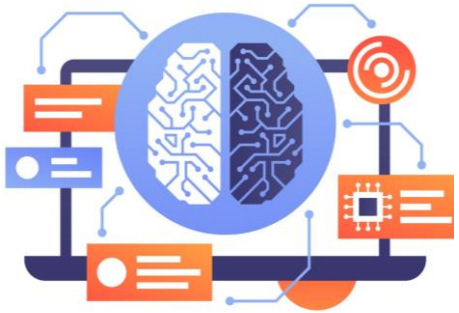
COS40007

**Dr. Abdur Forkan**

**Senior Research Fellow, AI and Machine Learning**

**Digital Innovation Lab**

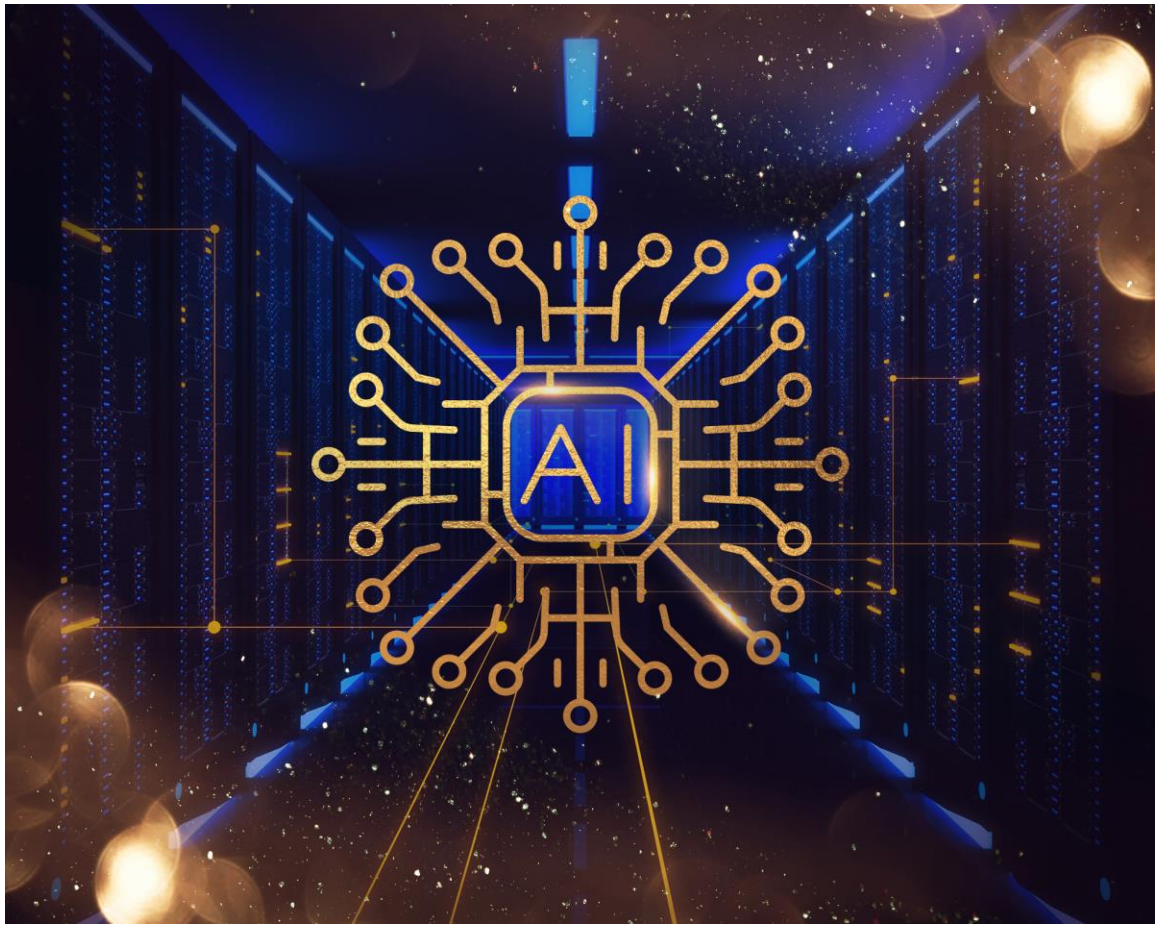
Seminar 8: 23rd September 2024



SWINBURNE UNIVERSITY OF TECHNOLOGY

# Overview

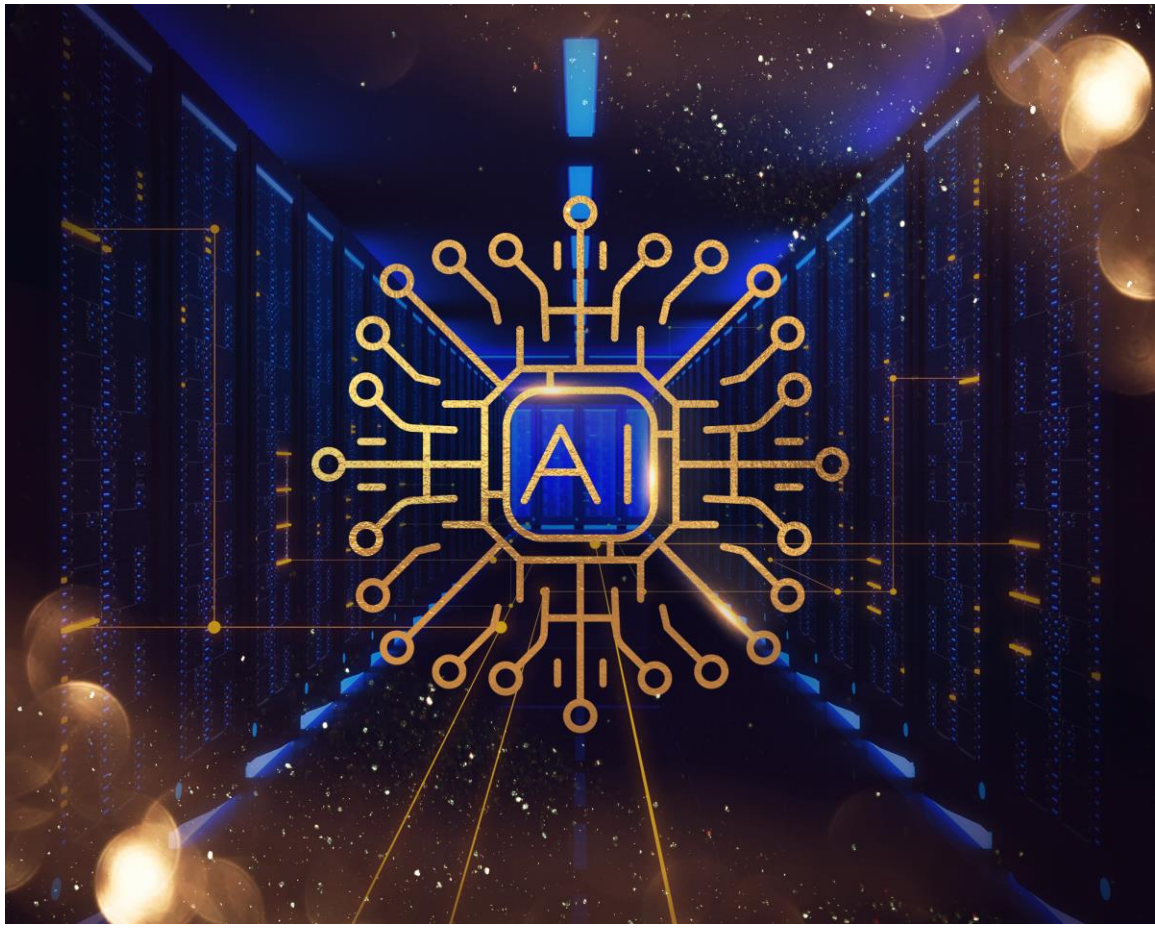
- ❑Basics of Clustering
- ❑Clustering techniques
- ❑Clustering examples
- ❑Distance measures for clustering
- ❑Deep learning in Clustering
- ❑Clustering applications




SWINBURNE UNIVERSITY OF TECHNOLOGY

# Required Reading

- Chapter 1 of “Applied Machine Learning and AI for Engineers”
- Chapter 10 of “Machine Learning with PyTorch and Scikit-Learn”



# At the end of this you should be able to

- Understand about unsupervised learning
- Understand about data for clustering
- Understand different clustering algorithms
- Understand clustering applications

# Clustering

# Unsupervised Learning

- Data is not labelled
- Discover hidden structures in data where we do not know the right answer upfront
- find a natural grouping in data without human intervention
- same cluster are more similar to each other than to those from different clusters
- Clustering is an unsupervised learning approach

# Clustering

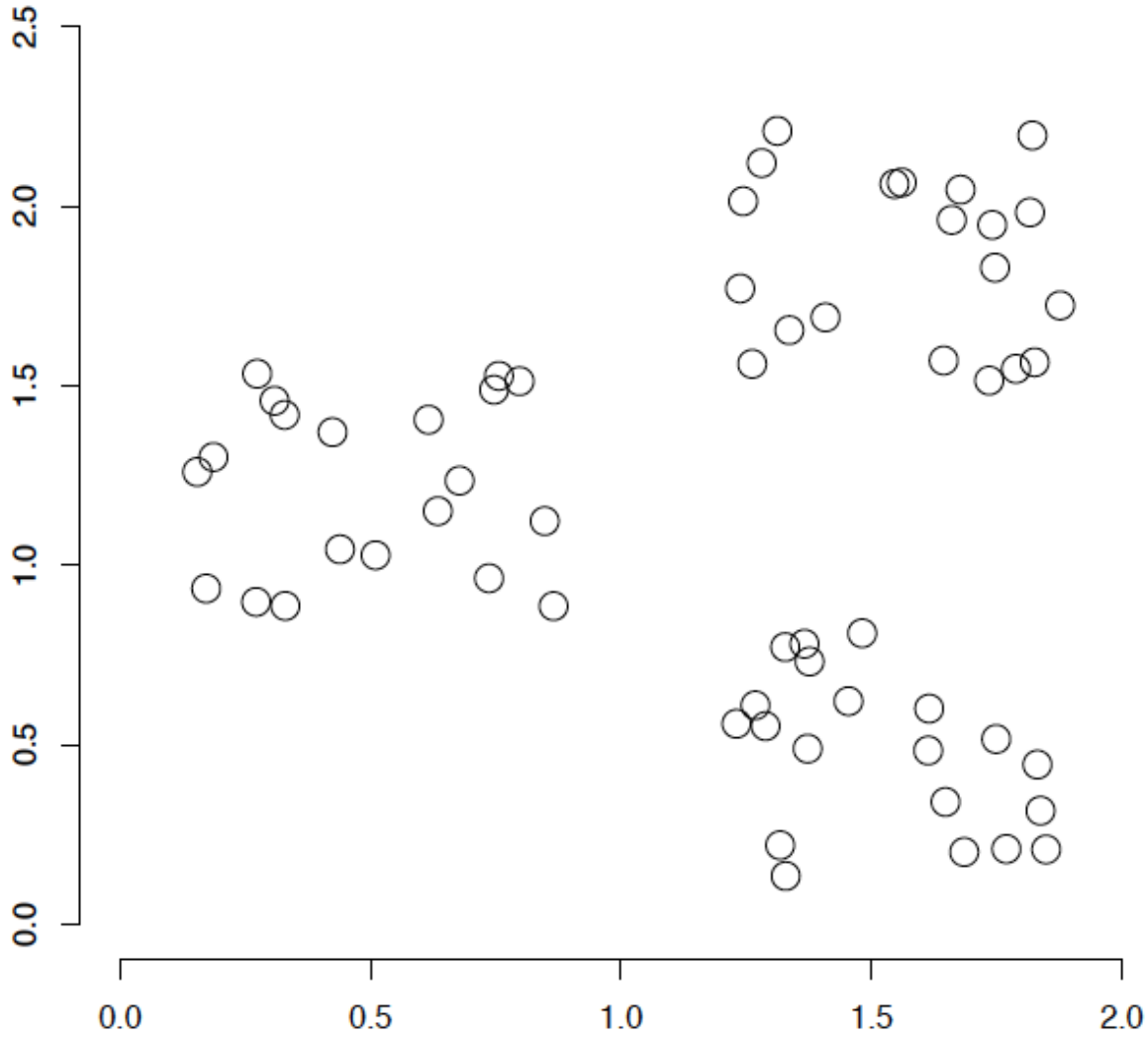
Clustering (or cluster analysis) is a technique that allows us to find groups of similar objects that are more related to each other than to objects in other groups

Example:

finding customers that share similar interests based on common purchase behaviors

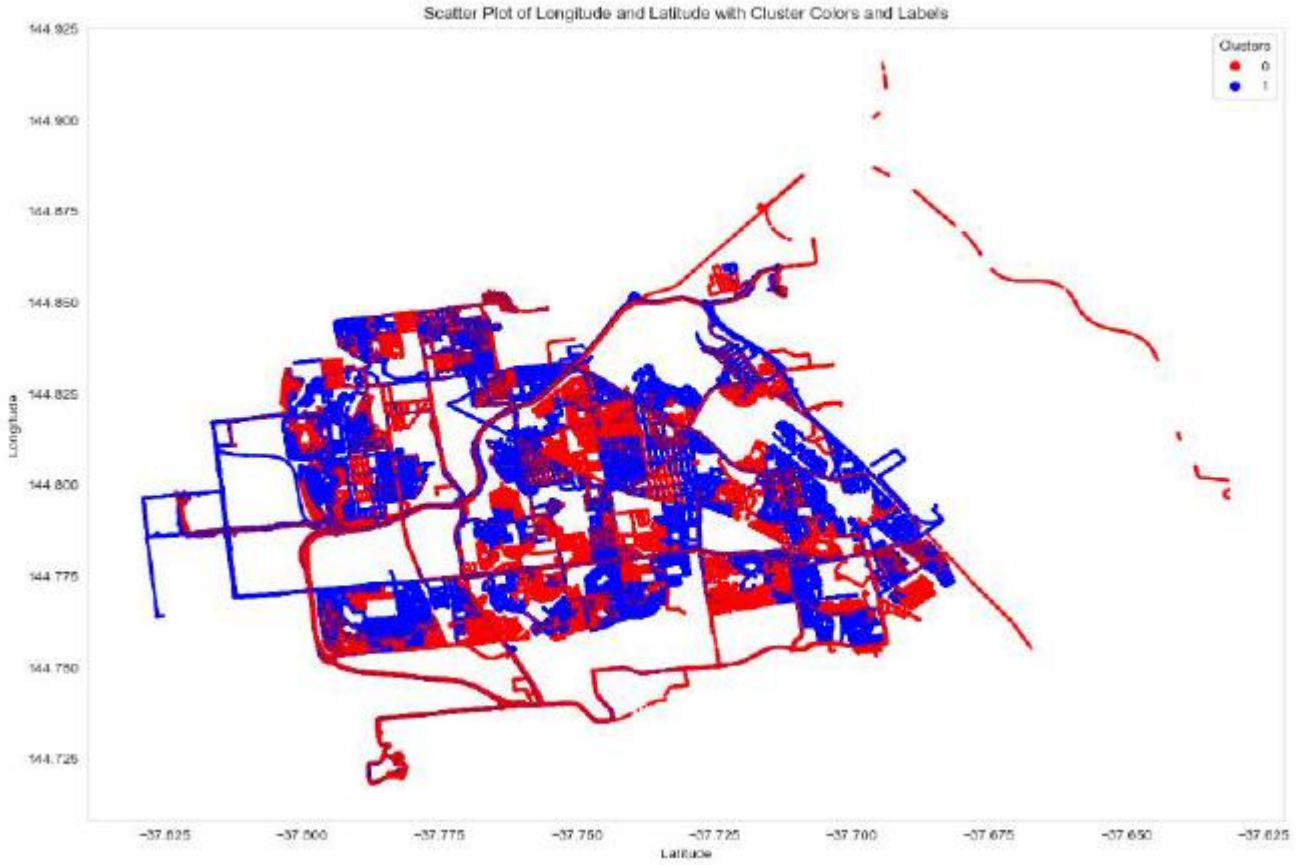
Historical and quality data from a die-casting machine in a production environment are grouped into clusters based on similarity of the data vectors. Members in each cluster are then analysed to determine operation modes or categories (e.g. ‘high probability of defects’ or ‘nominal’)

# Data partitioning technique



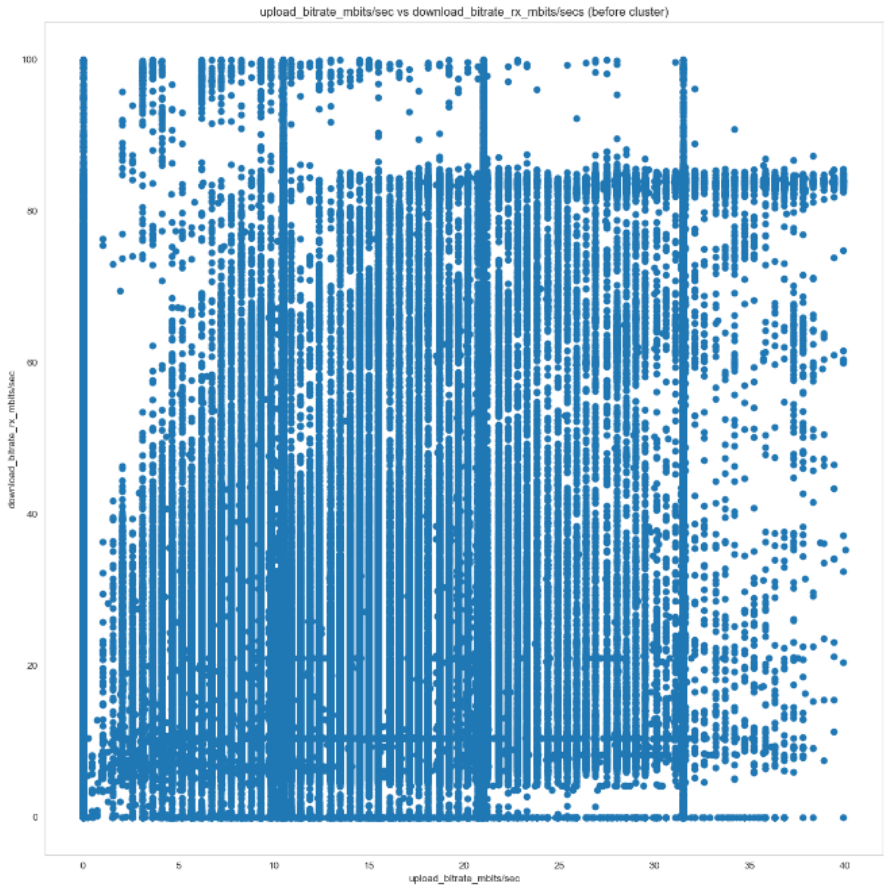

How would you design an algorithm for finding the three clusters in this case?

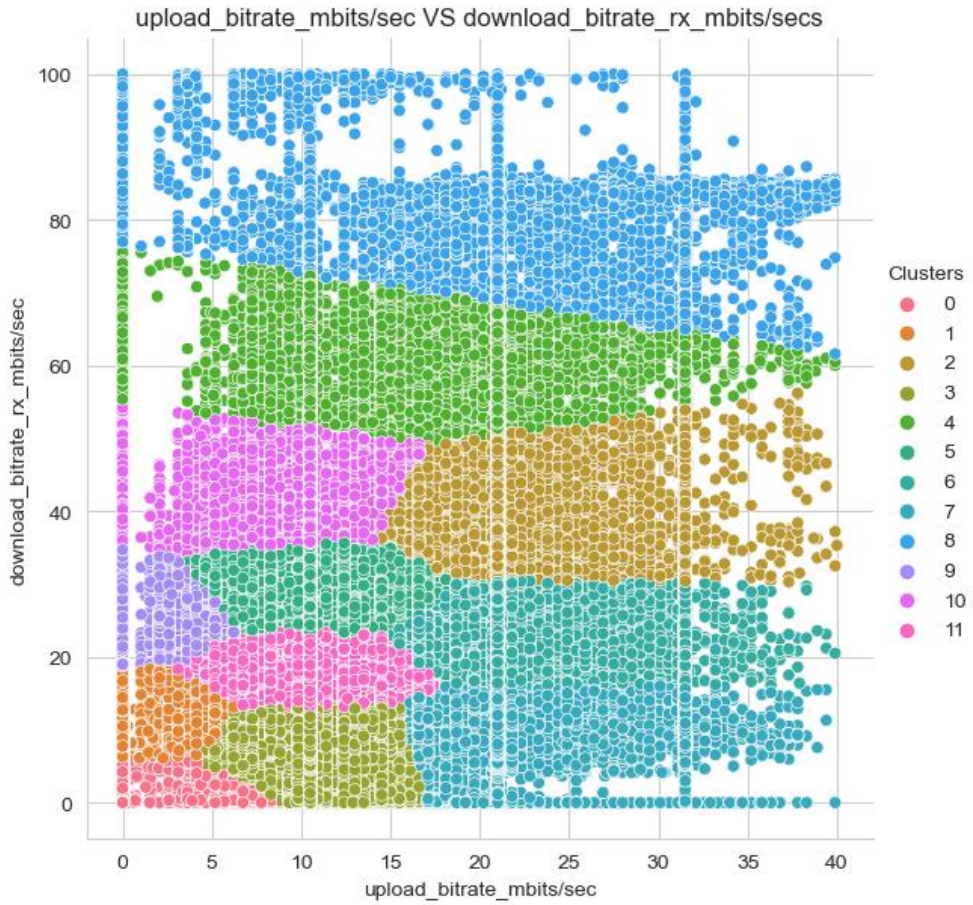
# Use of clustering for visualisation




SWINBURNE UNIVERSITY OF TECHNOLOGY

# Clustering




SWINBURNE UNIVERSITY OF TECHNOLOGY

# Clustering algorithms

## Flat algorithms

- Usually start with a random (partial) partitioning
- Refine it iteratively
  - *K* means clustering
  - (Model based clustering)

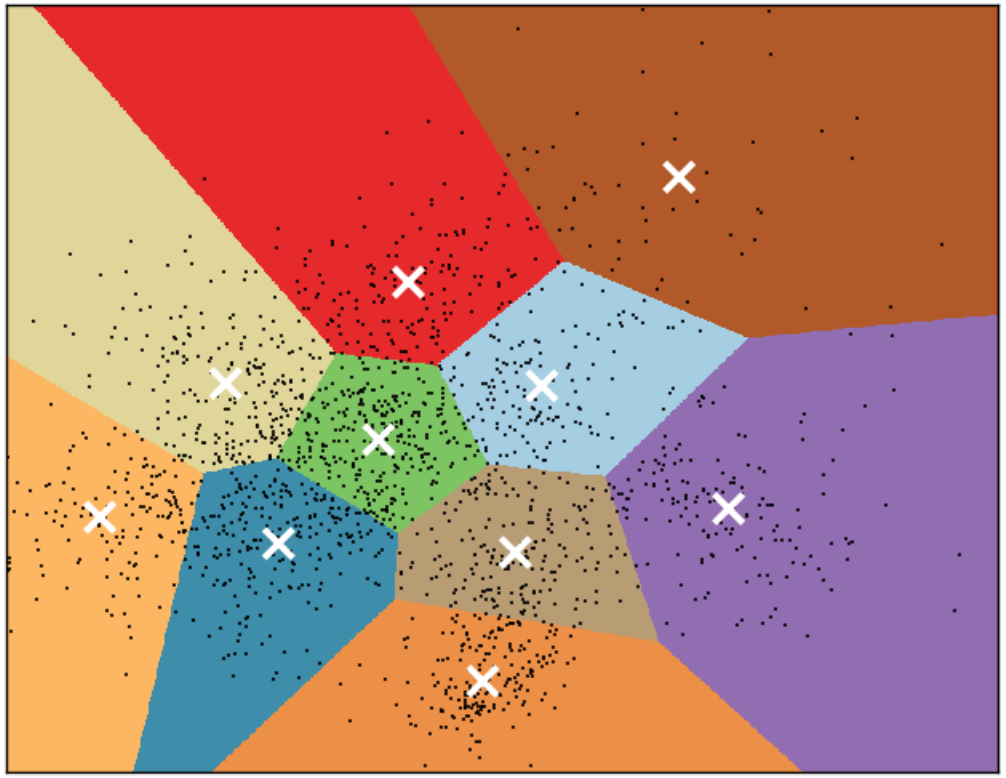
## Hierarchical algorithms

- Bottom-up, agglomerative
- (Top-down, divisive)

## Density-based algorithms

# K-Means Clustering

- Formally: a method of vector quantization
- Partition space into Voronoi cells
- Separate samples into *n* groups of equal variance
- Uses the Euclidean distance metric



# K Means

- Assumes documents are real-valued vectors.
- Clusters based on *centroids* (aka the *center of gravity* or mean) of points in a cluster, *c*:

- Reassignment of instances to clusters is based on distance to the current cluster centroids.
  - (Or one can equivalently phrase it in terms of similarities)

# How Many Clusters?

**Number of clusters *K* is given**

- Partition *n* docs into predetermined number of clusters

**Finding the "right" number of clusters is part of the problem**

- Given docs/points, partition into an "appropriate" number of subsets.
- E.g., for query results - ideal value of *K* not known up front - though UI may impose limits.

**Can usually take an algorithm for one flavor and convert to the other.**

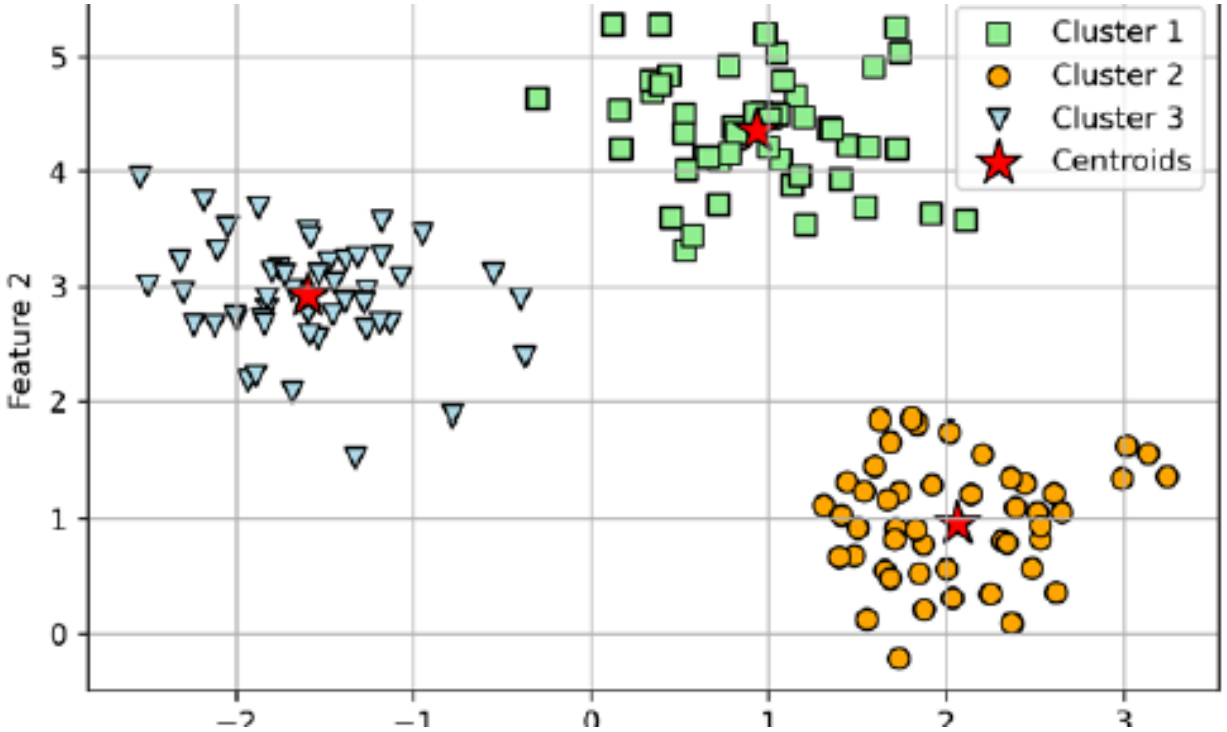
# K-means in sklearn

```
>>> from sklearn.cluster import KMeans
>>> km = KMeans(n_clusters=3,
... init='random',
... n_init=10,
... max_iter=300,
... tol=1e-04,
... random_state=0)

y_km = km.fit_predict(X)
```

# K-means

**Cluster can be visulisaed  using matplotlib**



# Basics of K-means

**Iterative refinement**

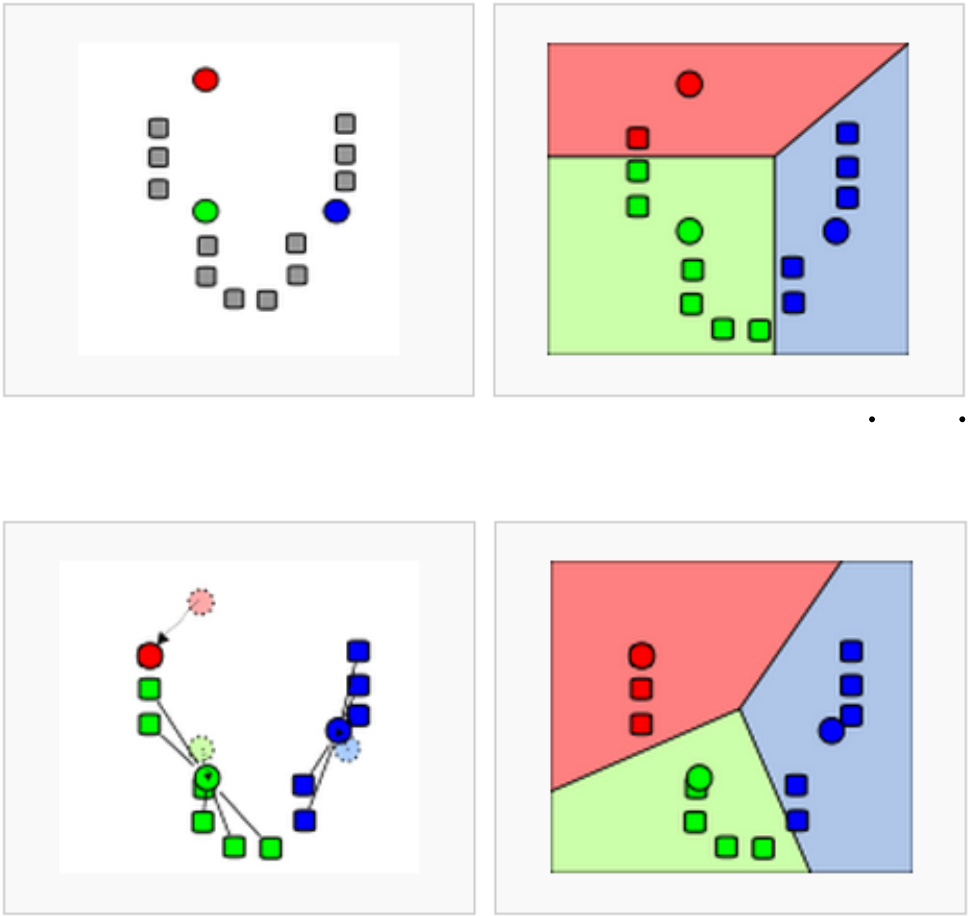
**Three basic steps**

**Step 1: Choose *k***

**Iterate over:**

- Step 2: Assignment
- Step 3: Update

**Repeats until convergence has been reached**

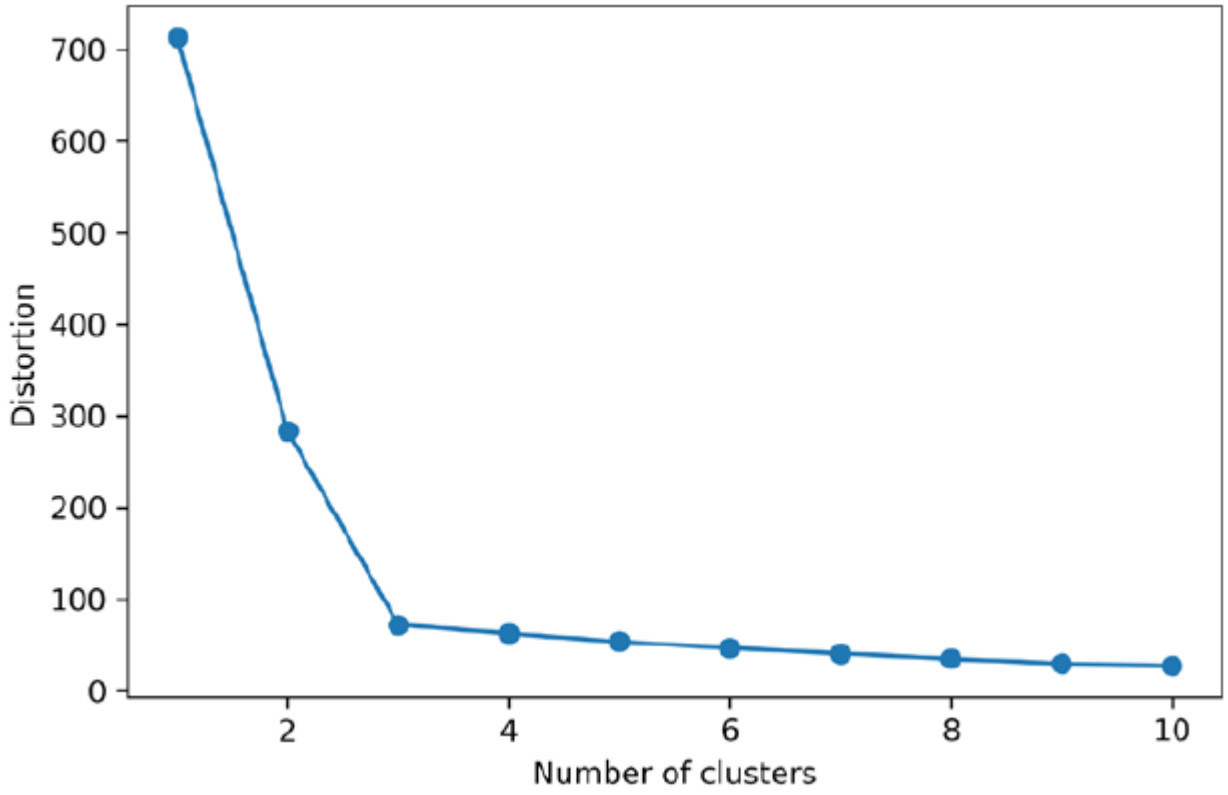


# When to use K-Means

- Normally distributed data
- Large number of samples
- Not too many clusters
- Distance can be measured in a linear fashion

# How to find Optimal Number of Clusters?

- Elbow method is used to find optimal number of clusters, k for a given dataset
- we need to use intrinsic metrics—such as the within-cluster SSE (distortion)—to compare the performance of different k-means clustering models.
- If k increases, the distortion will decrease.
- identify the value of k where the distortion begins to increase most rapidly



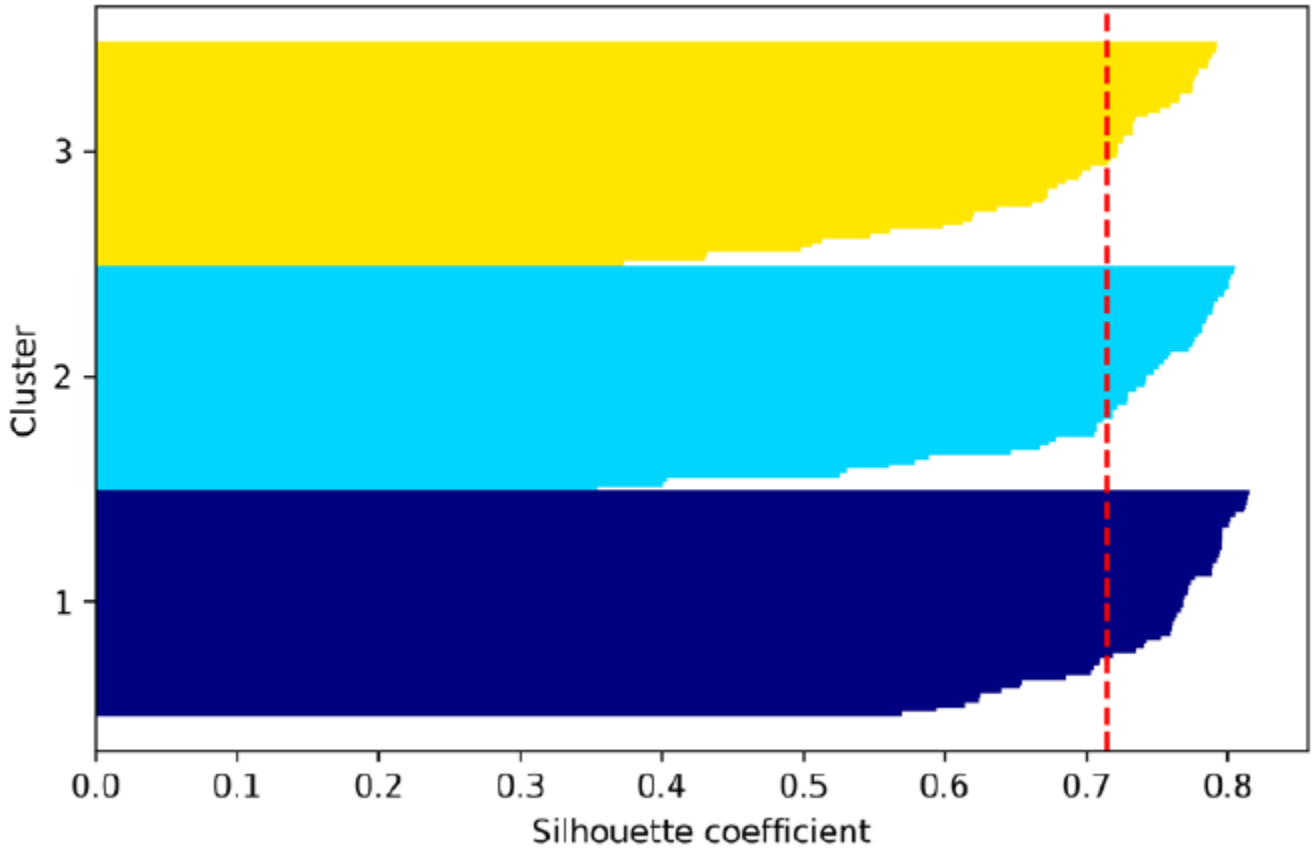
# What is a good clustering?

- Internal criterion: A good clustering will produce high quality clusters in which:
  - the intra-class (that is, intra-cluster) similarity is high
  - the inter-class similarity is low
  - The measured quality of a clustering depends on both the document representation and the similarity measure used

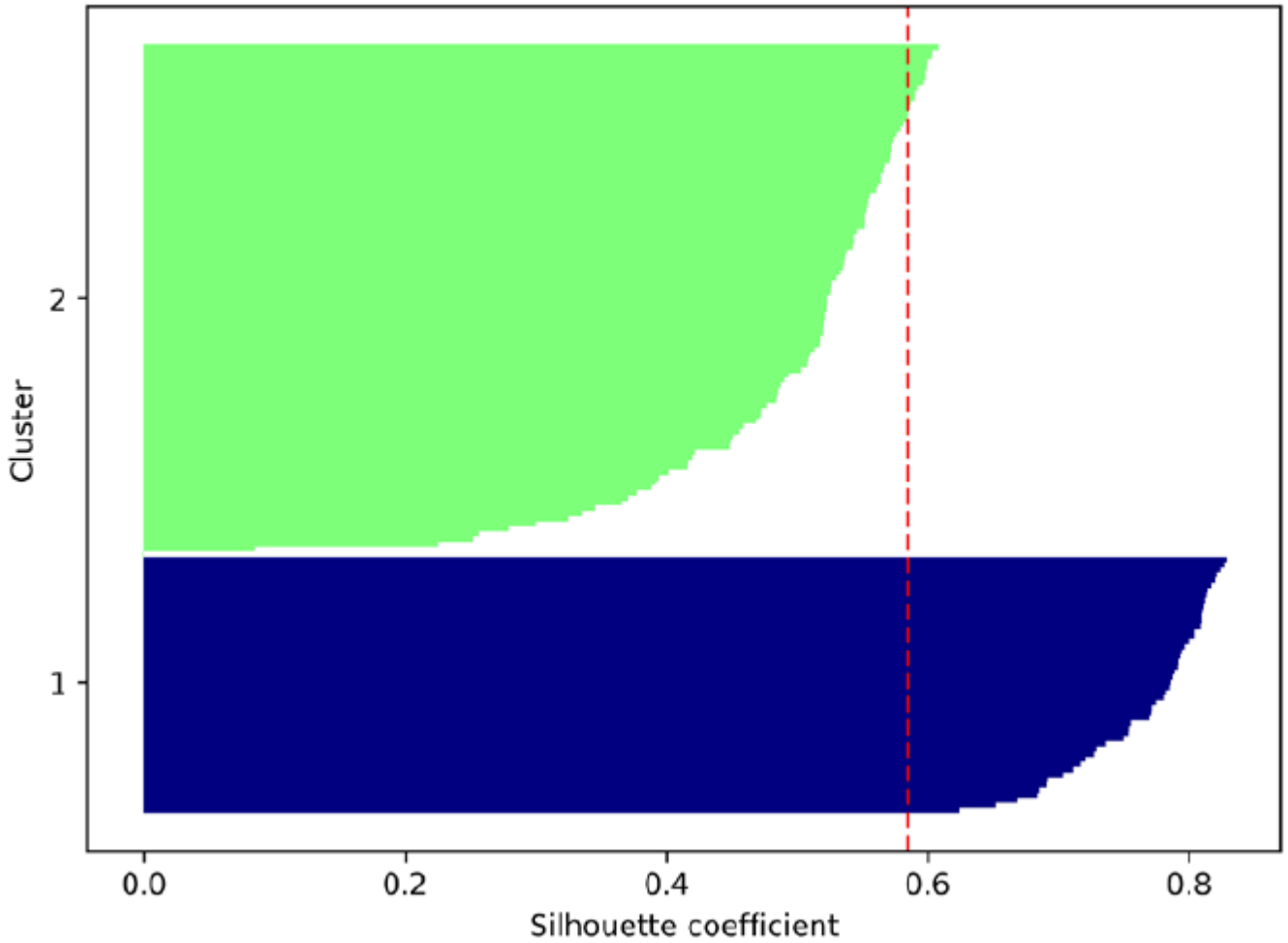
# Measuring clustering quality: silhouette score

- Silhouette coefficient
  - No ground truth
  - Mean distance between an observation and all other points in its cluster
  - Mean distance between an observation and all other points in the **next nearest** cluster
- Silhouette score in scikit-learn
  - Mean of silhouette coefficient for all of the observations
  - Closer to 1, the better the fit
  - Large dataset == long time

# Silhouette score: Example of good clustering

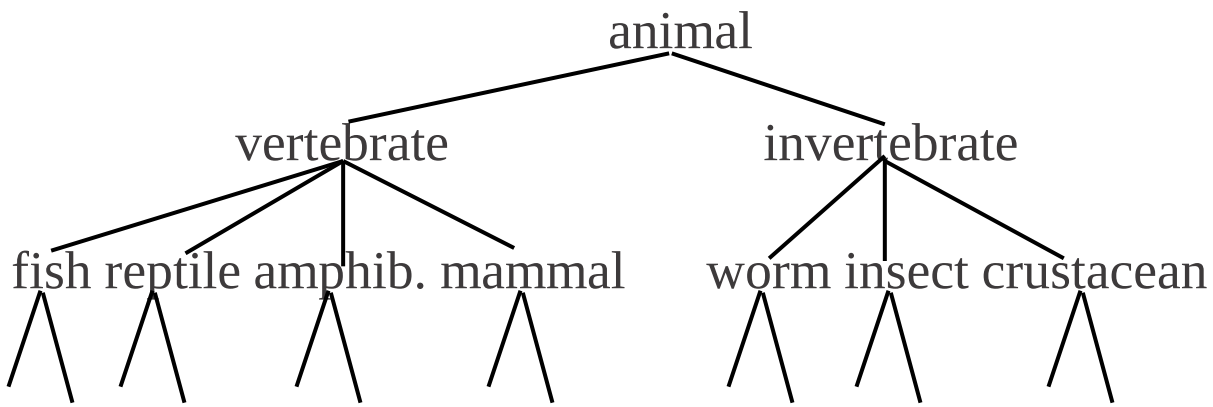


# Silhouette score: Example of bad clustering



# Hierarchical Clustering

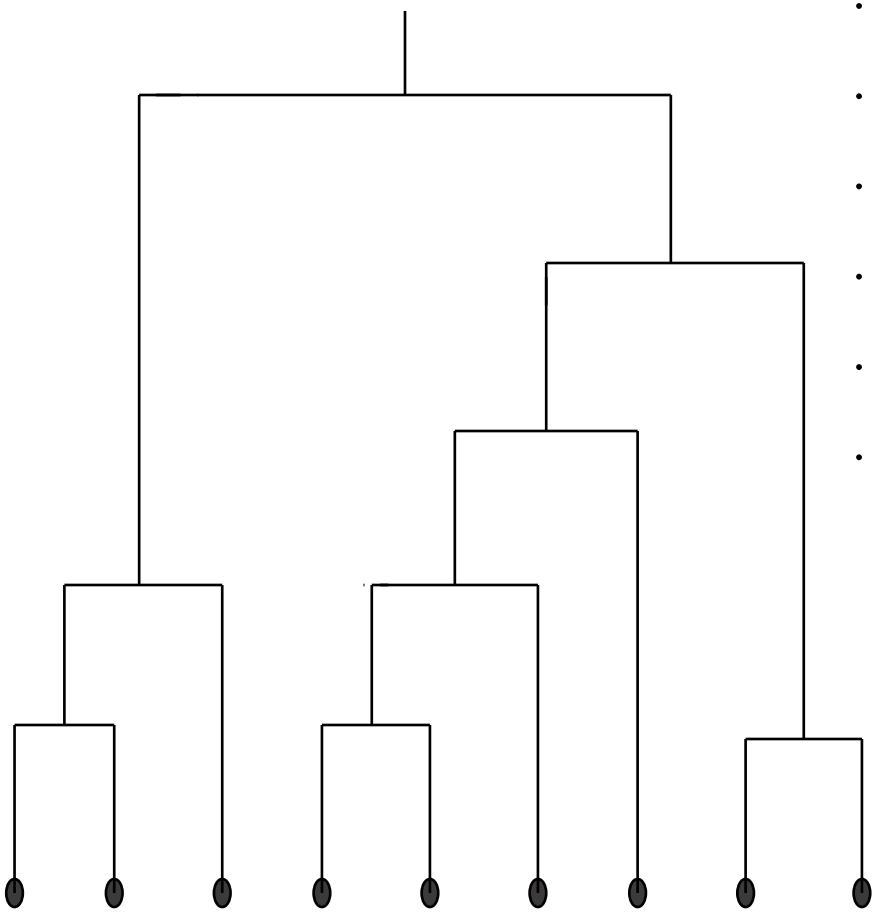
Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.




One approach: recursive application of a partitional clustering algorithm.

# Dendrogram: Hierarchical Clustering

Clustering obtained by cutting the dendrogram at a desired level: each connected component forms a cluster.

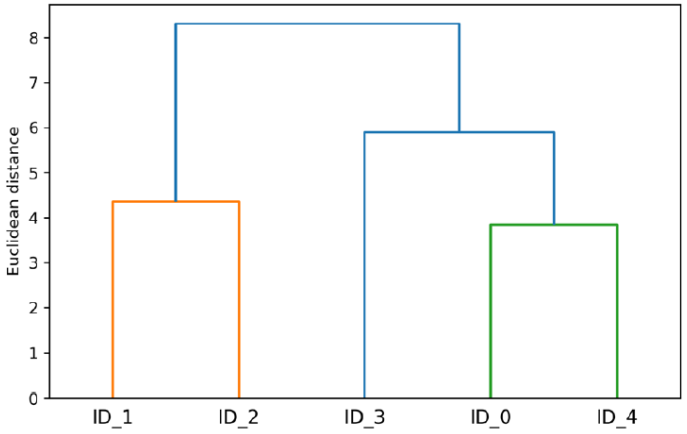


# Hierarchical Clustering

Distance matrix

| | X | Y | Z |
|---|---|---|---|
| ID_0 | 6.964692 | 2.861393 | 2.268515 |
| ID_1 | 5.513148 | 7.194690 | 4.231065 |
| ID_2 | 9.807642 | 6.848297 | 4.809319 |
| ID_3 | 3.921175 | 3.431780 | 7.290497 |
| ID_4 | 4.385722 | 0.596779 | 3.980443 |

```
from scipy.spatial.distance import pdist, squareform
row_dist = pd.DataFrame(squareform(
pdist(df, metric='euclidean')),
columns=labels, index=labels)
```

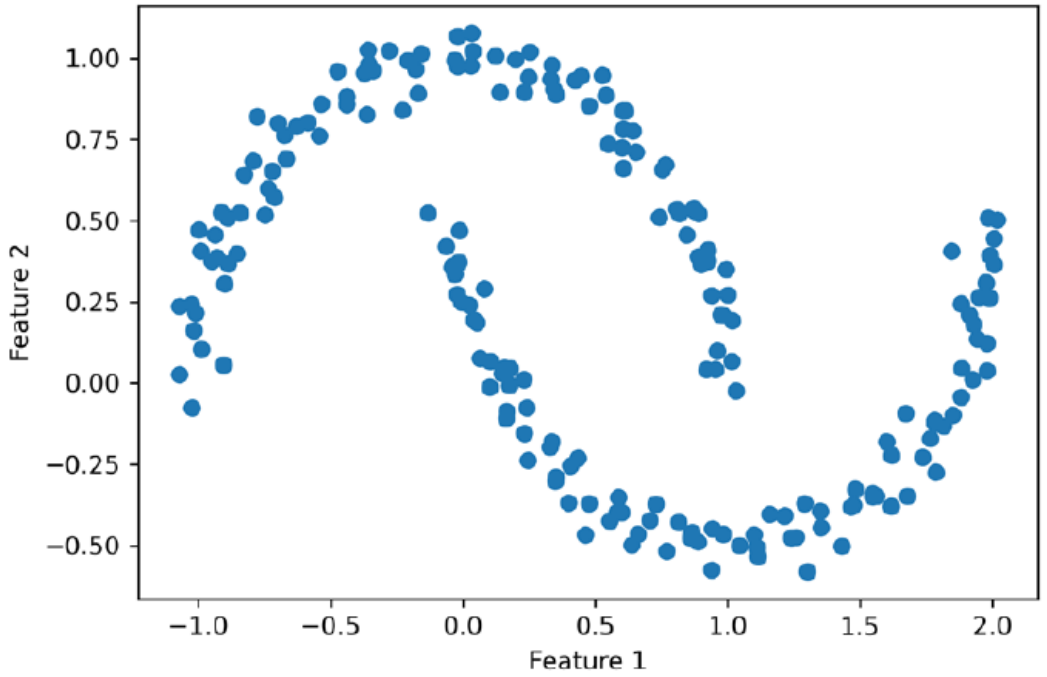


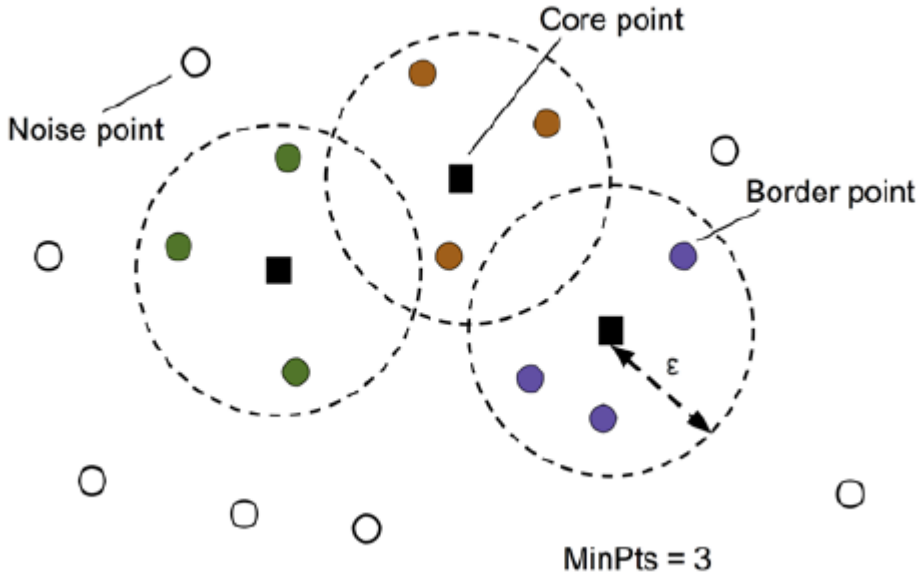

```
>>> from scipy.cluster.hierarchy import dendrogram
>>> # make dendrogram black (part 1/2)
>>> # from scipy.cluster.hierarchy import set_link_color_palette
>>> # set_link_color_palette(['black'])
>>> row_dendr = dendrogram(
...     row_clusters,
...     labels=labels,
...     # make dendrogram black (part 2/2)
...     # color_threshold=np.inf
... )
>>> plt.tight_layout()
>>> plt.ylabel('Euclidean distance')
>>> plt.show()
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

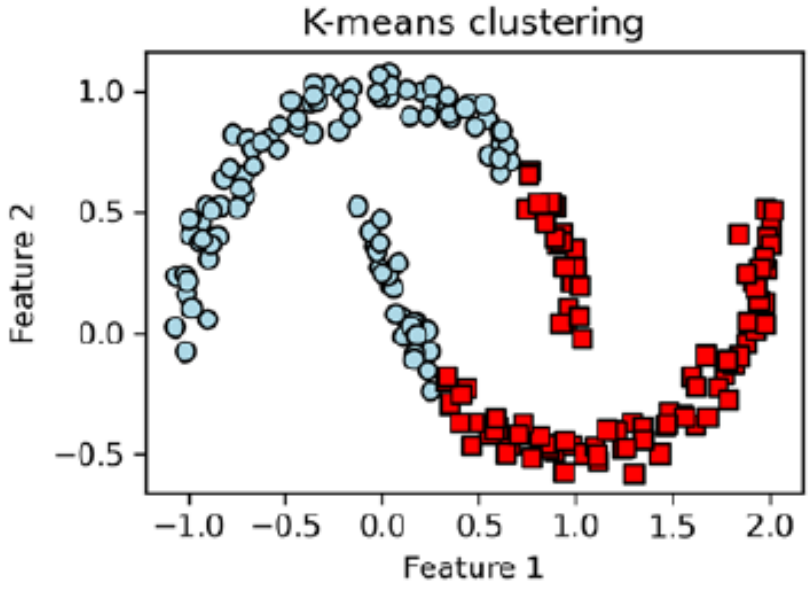
# Density-based clustering: DBSCAN

- density-based clustering assigns cluster labels based on dense regions of points

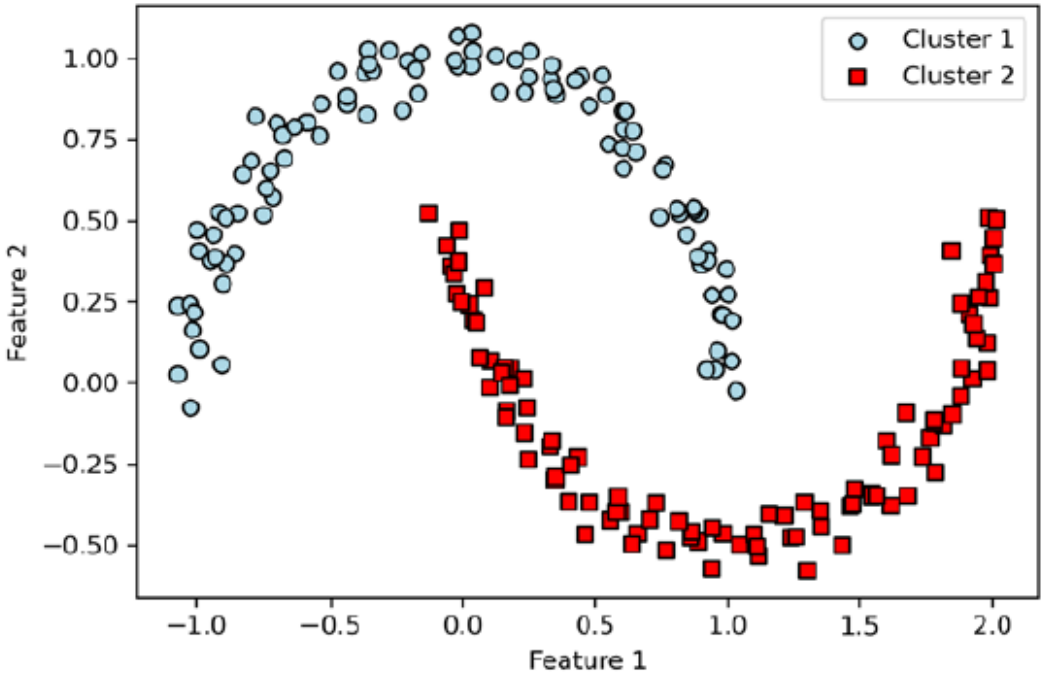


# DBSCAN




```
>>> from sklearn.cluster import DBSCAN
>>> db = DBSCAN(eps=0.2,
... min_samples=5,
... metric='euclidean')
>>> y_db = db.fit_predict(X)
>>> plt.scatter(X[y_db == 0,
```



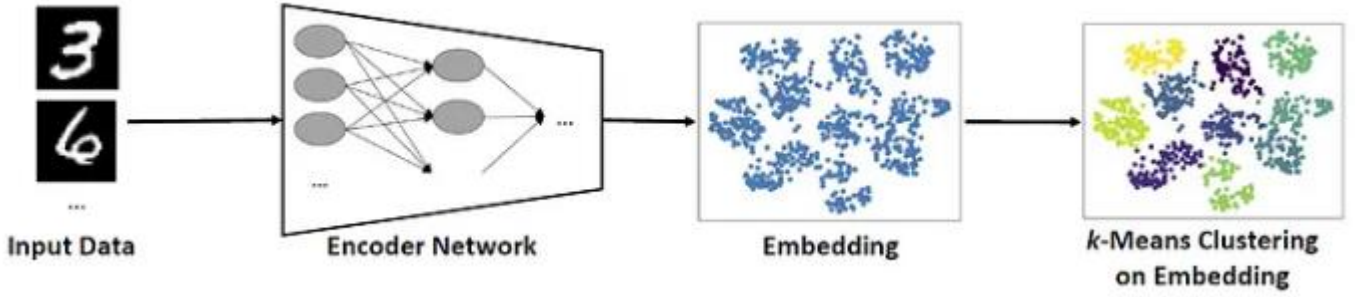

SWINBURNE UNIVERSITY OF TECHNOLOGY

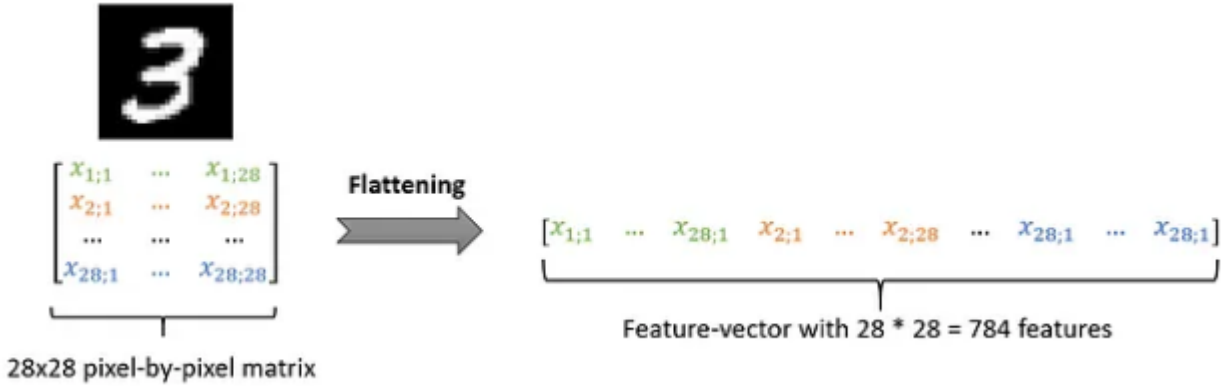
# Distance measure for clustering

- Euclidean Distance: most commonly used, alculates the straight-line distance between two points in n-dimensional space.
- Manhattan Distance: the total of the absolute differences between their Cartesian coordinates
- Cosine Similarity: calculates the cosine of the angle between two data points, with a higher cosine value indicating greater similarity
- Minkowski Distance: a generalized form of both Euclidean and Manhattan distances.
- Jaccard Index: calculates the ratio of the number of features shared by two data points to the total number of features.

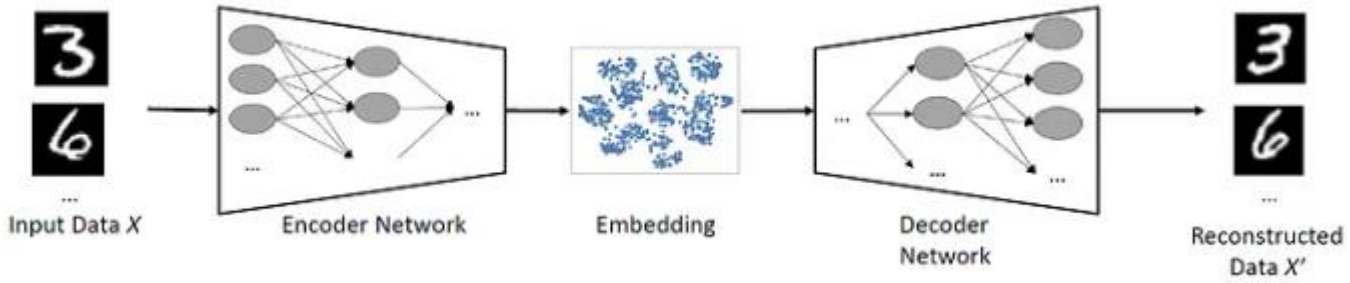
# Clustering high-dimensional data: Auto-Encoder

- Auto-Encoders can effectively reduce the dimensionality of the data to improve the accuracy of the subsequent clustering
- Example:  automatically group similar images in the same clusters



# Auto-Encoder




```
n_input_features = 784 # Dimension of MNIST dataset
model = AutoEncoder(input_size=n_input_features,
hiden_layers=[500, 500, 2000, # neurons in the hidden layers
10 # dimension of the embedding
],
# Prevent overfitting by deactivating 20% of the neurons during training
dropout_rate=0.2
)
```

# Autoencoder

```
from torchvision.datasets import MNIST
from torch.utils.data import ConcatDataset
from torchvision import transforms
# Transform into a tensor and apply normalization
transform = transforms.Compose([transforms.ToTensor(),
transforms.Normalize((0.5,), (0.5,)),
])
# Load Training Data
trainset = MNIST('./', download=True,
train=True,
transform=transform)# Load Test data
testset = MNIST('./', download=True,
train=False,
transform=transform)# Combine both training and test data
dataset = ConcatDataset([trainset, testset])# Use a Data loader, this does not
load the whole dataset at once
# but we can traverse it in batches for training
dataloader = torch.utils.data.DataLoader(dataset,
batch_size=256,
shuffle=True,
num_workers=10)
```

```
X_train = trainset.data.numpy().reshape(60000, 784)
X_test = testset.data.numpy().reshape(10000, 784)
y_train = np.array(trainset.targets)
y_test = np.array(testset.targets)
# Concatenate training and test data
y = np.concatenate([y_train, y_test])
X = np.concatenate([X_train, X_test])
# Convert X to Tensor object
X_tensor = Tensor(X).to(device)
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

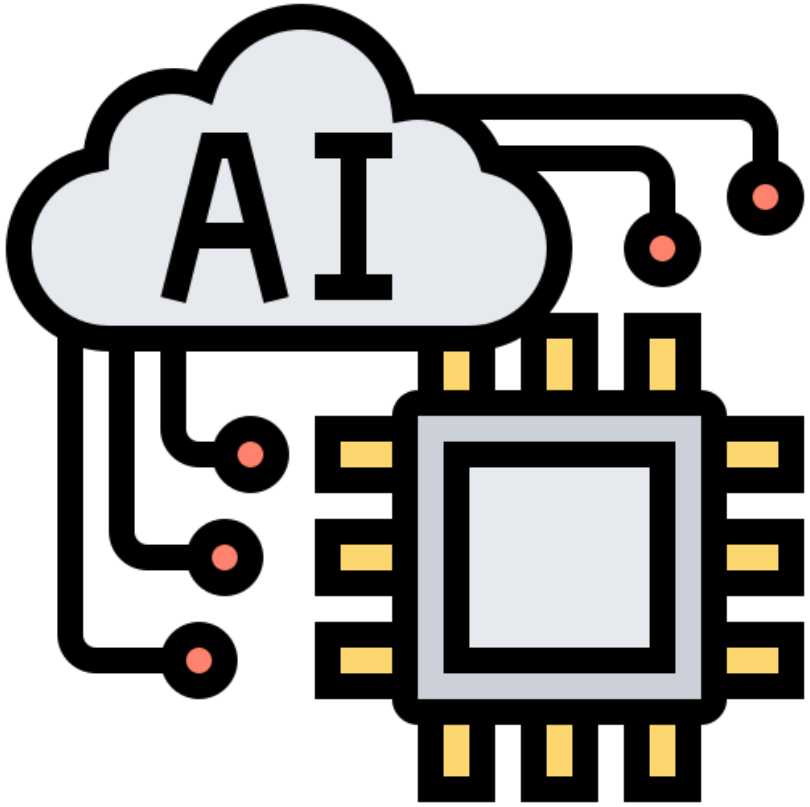
# Auto-Encoder

- Auto-Encoders are powerful unsupervised deep learning networks to learn a lower-dimensional representation
- Use specifically in image data
- Auto-encoder improves the performance of k-means

# Clustering applications

- Understanding different types of defects/anomalies in production
- Clustering customers/consumers
- Understand similar trends in data
- Grouping locations in map based on features
- Clustering users in a social network
- Clustering articles based on topics
- Clustering photographs by content, grouping medical images based on visual features
- Clustering music tracks, video footages

# Learn, Practice and Enjoy the AI journey




SWINBURNE UNIVERSITY OF TECHNOLOGY