Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant

Unsupervised Learning

Clustering, Dimensionality Reduction

AGENDA

1. Intro to Unsupervised Learning
2. Clustering Techniques
3. Dimensionality Reduction
4. Code Exercise

What's the best advice you've ever received, and who gave it to you?

1. Intro to Unsupervised Learning

A type of machine learning where the model learns patterns from unlabeled data, finding hidden structures or groupings without supervision.

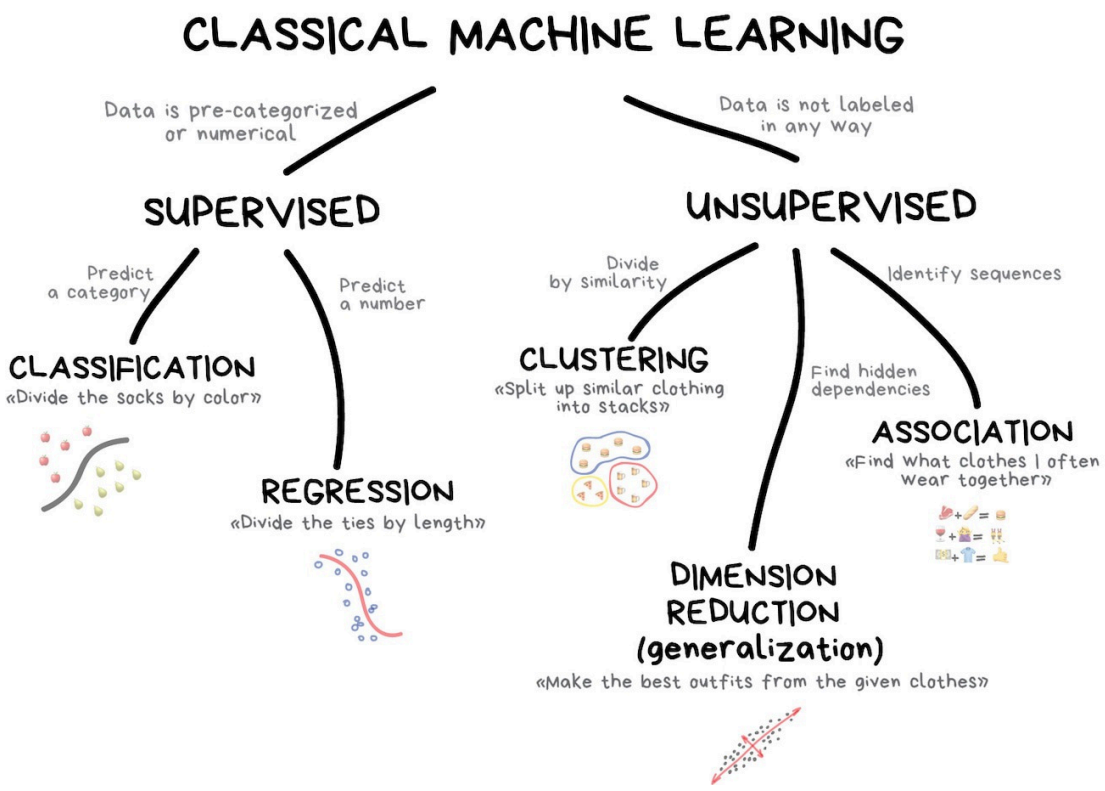
- **Key Differences from Supervised Learning**:
  - No labeled data (no target or outcome variable).
  - Focuses on finding patterns and structures within data.

Real-World Applications of Unsupervised Learning

- **Customer Segmentation**: Grouping customers based on purchasing behavior for targeted marketing.
- **Anomaly Detection**: Identifying unusual patterns or outliers in network security or credit card transactions.
- **Document Clustering**: Organizing documents into thematic groups for topic analysis.

Types of Unsupervised Learning

1. **Clustering**: Grouping similar data points together.
2. **Dimensionality Reduction**: Reducing the number of features while retaining essential information.



Clustering

- **Definition**: Clustering is a technique to organize data into groups, or clusters, based on similarity, where data points within each cluster are more similar to each other than to those in other clusters.
- **Goals**:
  - **Identify Patterns**: Find natural groupings within data.
  - **Simplify Data**: Make complex datasets more manageable and interpretable.

Popular Clustering Algorithms

1. **K-Means Clustering**: Partitions data into K clusters by minimizing the variance within each cluster.
2. **Hierarchical Clustering**: Builds a tree of clusters based on similarity, either merging smaller clusters or dividing a large cluster.
3. **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)**: Groups data points based on density, capable of handling noise and clusters of varying shapes.

K-Means Clustering

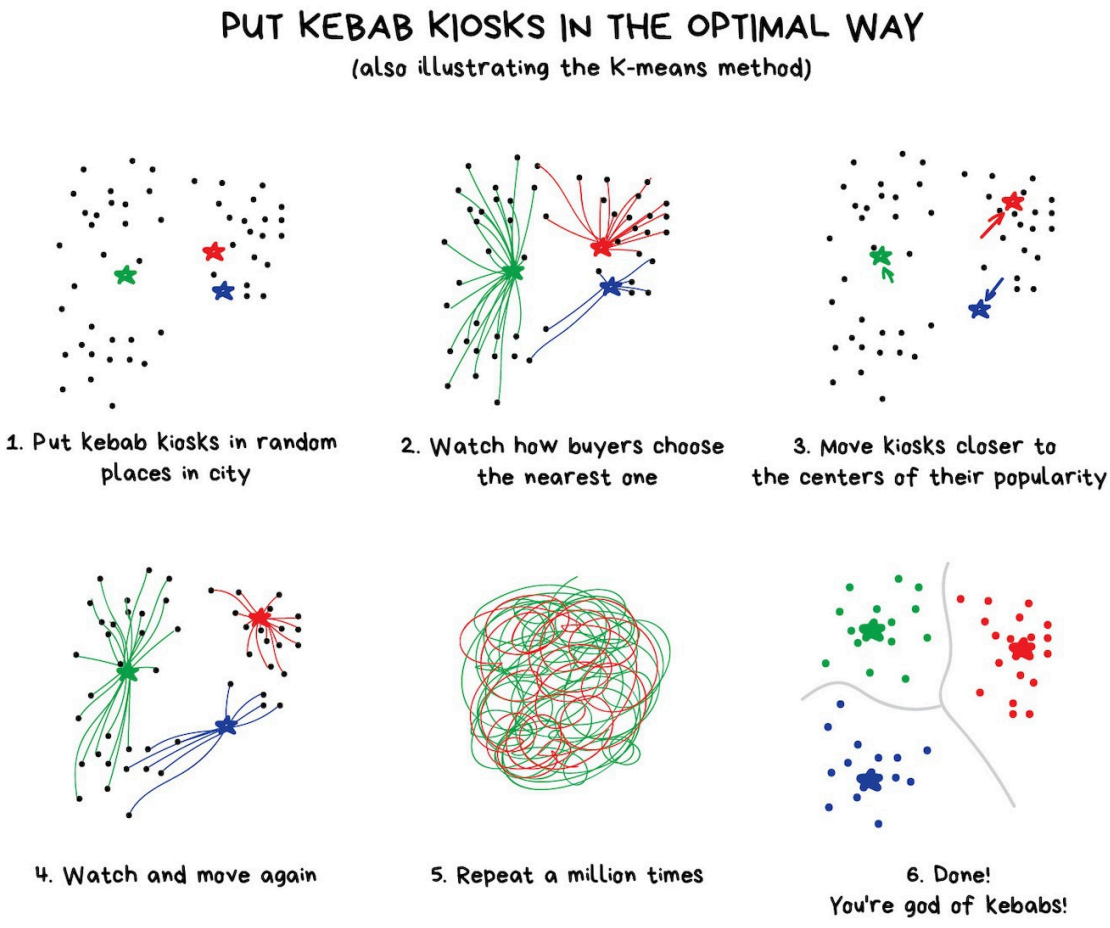
1. **Initialization**: Randomly select K points as initial cluster centroids.
2. **Assignment**: Assign each data point to the nearest centroid.
3. **Update**: Recalculate the centroids by averaging the points in each cluster.
4. **Convergence**: Repeat assignment and update steps until centroids stabilize.

Choosing K (Number of Clusters)

- **Elbow Method**: Plot the within-cluster sum of squares (WCSS) against the number of clusters and identify the "elbow" point, where additional clusters do not significantly reduce WCSS.
- **Silhouette Score**: Measures cohesion within clusters and separation between clusters, with a higher score indicating better-defined clusters.

Practical Considerations

- **Initialization Methods**: Using methods like `k-means++` to choose better initial centroids and improve convergence.
- **Handling Outliers**: Outliers can skew centroids; consider data preprocessing to manage outliers.



Demo & Small Activity (15-20 mins)

1. **Dataset**: Load a small dataset (e.g., Iris dataset or a synthetic dataset).
2. **Implementation**: Apply K-Means using Python's `scikit-learn` library.
3. **Visualization**:
   - Plot the clusters to visualize the groups.
   - Discuss the interpretation of each cluster and what it reveals about the data.

Example Code:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

# Load dataset
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)

# Apply K-Means
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=42)
df['Cluster'] = kmeans.fit_predict(df)

# Visualize clusters
plt.scatter(df.iloc[:, 0], df.iloc[:, 1], c=df['Cluster'],
cmap='viridis')
plt.xlabel(data.feature_names[0])
plt.ylabel(data.feature_names[1])
plt.title("K-Means Clustering on Iris Dataset")
plt.show()
```

2. Clustering Techniques and Evaluation

Hierarchical Clustering

Hierarchical clustering is a method of cluster analysis that seeks to build a hierarchy of clusters. Unlike partitioning methods like K-Means, which divide data into a fixed number of clusters, hierarchical clustering creates a tree-like structure (dendrogram) that shows the arrangement of clusters at various levels of granularity.

Key Features of Hierarchical Clustering

1. **Types of Hierarchical Clustering**:

   - **Agglomerative Clustering**: This is a bottom-up approach where each data point starts as its own cluster. The algorithm iteratively merges the closest pairs of clusters until only one cluster remains or a desired number of clusters is reached.

- **Divisive Clustering**: This is a top-down approach, starting with one all-encompassing cluster and recursively splitting it into smaller clusters.

2. **Dendrogram**: A visual representation of the clustering process. The dendrogram shows the arrangement of clusters and allows you to choose a level at which to cut the tree to obtain a desired number of clusters.

3. **Linkage Criteria**: The method used to calculate the distance between clusters during the merging or splitting process. Common linkage criteria include:

   - **Single Linkage**: The distance between the closest points of two clusters.
   - **Complete Linkage**: The distance between the farthest points of two clusters.
   - **Average Linkage**: The average distance between all pairs of points in the two clusters.
   - **Ward's Method**: Aims to minimize the total within-cluster variance.

4. **Distance Metrics**: The choice of distance metric (e.g., Euclidean distance, Manhattan distance) can significantly affect the resulting clusters. The metric defines how the distance between points is calculated.

5. **Applications**: Hierarchical clustering is widely used in various fields, including:

   - **Biology**: For clustering gene expression data or species classification.
   - **Market Research**: For customer segmentation based on purchasing behavior.
   - **Image Analysis**: For grouping similar images or features.

Advantages and Disadvantages

- **Advantages**:
  - No need to pre-specify the number of clusters.
  - The dendrogram provides a clear visual representation of data structure.
  - Can handle different shapes and sizes of clusters.
- **Disadvantages**:
  - Computationally expensive for large datasets (time complexity can be O(n^3)).
  - Sensitive to noise and outliers.
  - Once a decision is made to merge clusters, it cannot be undone.

Exercise: Hierarchical Clustering with Dendrogram Visualization

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# your code here

from sklearn.datasets import load_iris
```

In [2]:
```
# Load the Iris dataset
iris = load_iris()
```

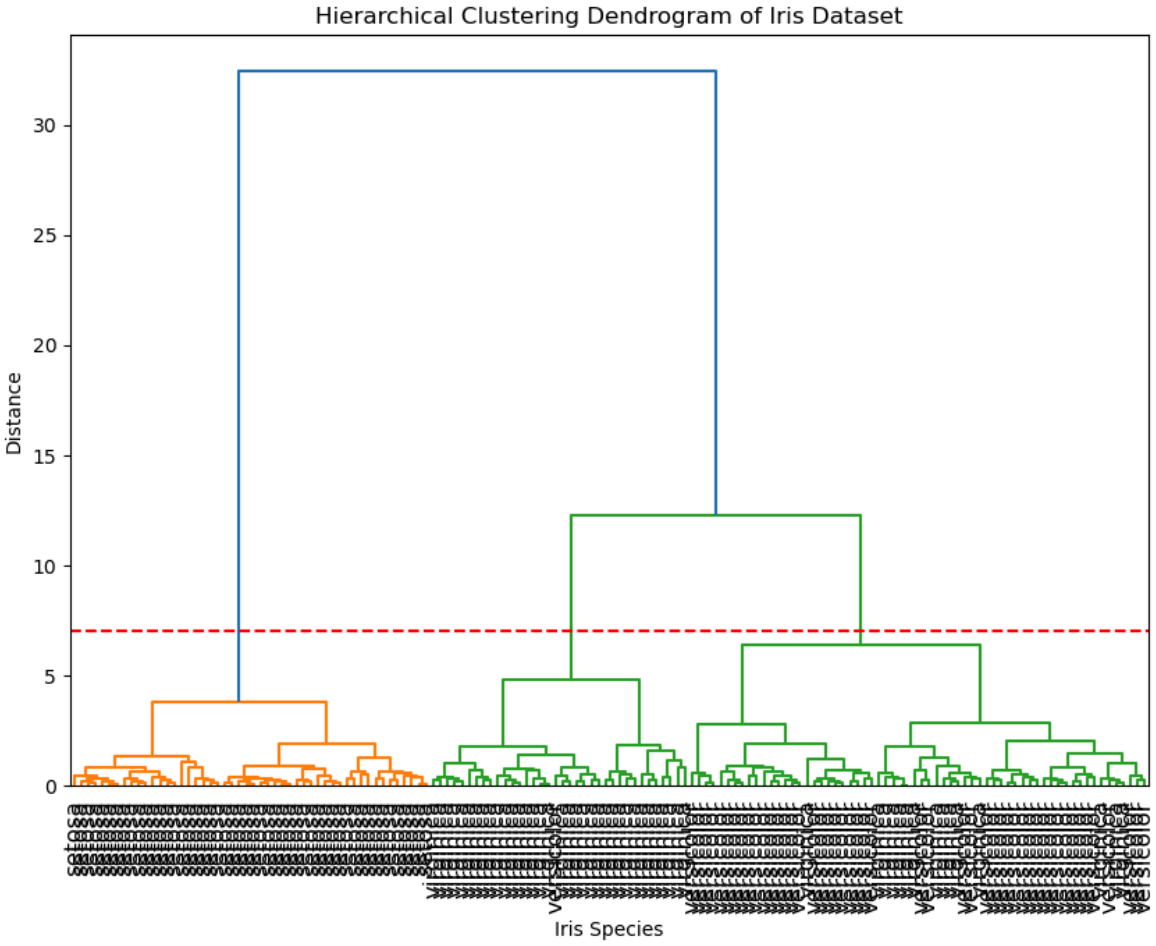
```
X = iris.data  # features
y = iris.target  # target labels
```

In [3]:
```
# Perform hierarchical clustering using Ward's method

# your code here
```

In [4]:
```
# Create a dendrogram

plt.figure(figsize=(10, 7))
dendrogram(Z, labels=iris.target_names[y], leaf_rotation=90, leaf_font_size=12)
plt.title('Hierarchical Clustering Dendrogram of Iris Dataset')
plt.xlabel('Iris Species')
plt.ylabel('Distance')
plt.axhline(y=7, color='r', linestyle='--')  # Optional line to indicate cut-off
plt.show()
```



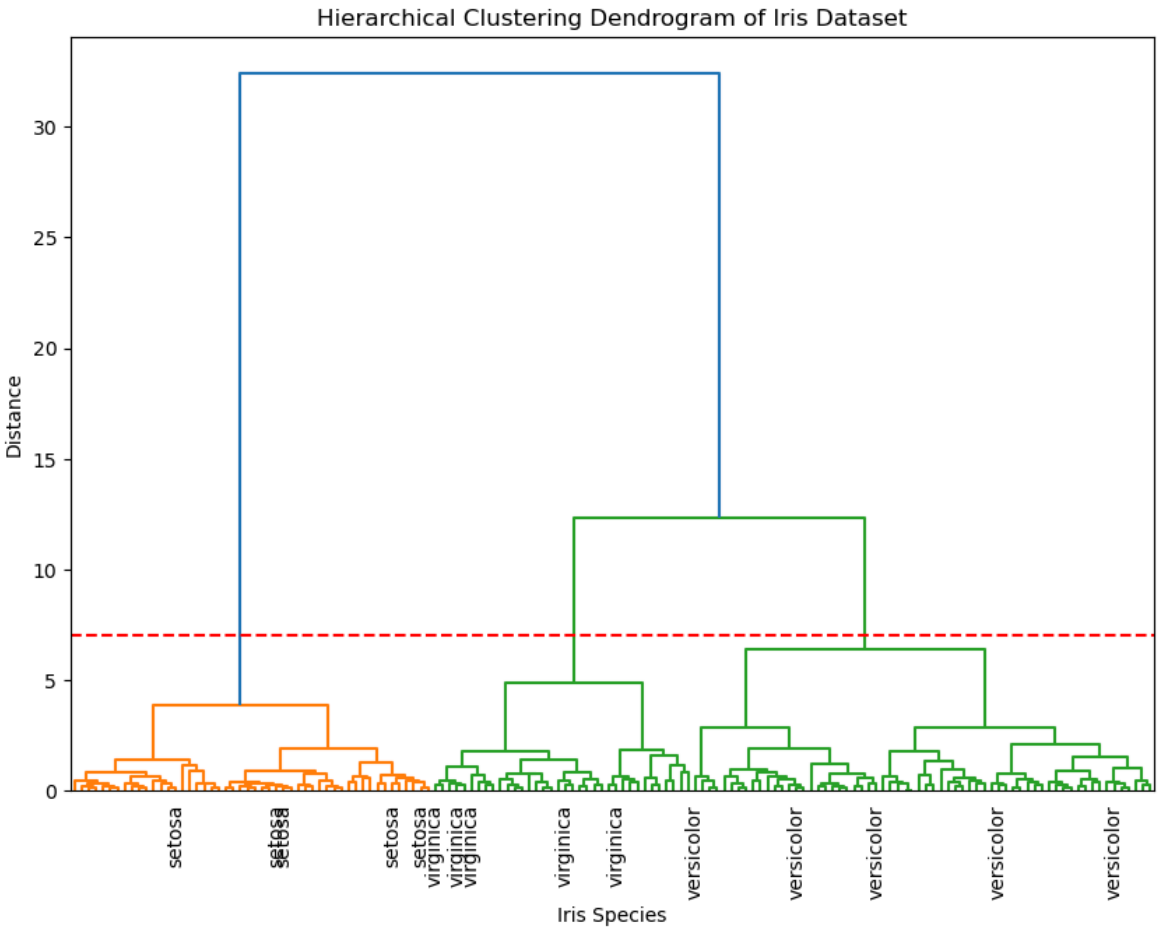
In [5]:
```
# Generate labels for the dendrogram
# Show label for every 20th entry, otherwise leave it empty

# your code here
```

In [6]:
```
# Create a dendrogram
plt.figure(figsize=(10, 7))


# Create the dendrogram with the reduced labels


#your code here
```



- https://scikit-learn.org/stable/auto_examples/cluster/plot_agglomerative_dendrogram.html#sphx-glr-auto-examples-cluster-plot-agglomerative-dendrogram-py

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) (Advanced/Optional)

How DBSCAN Works Learn More...

- **Core Points**: Points with at least `min_samples` neighbors within a specified radius `eps`.
- **Border Points**: Points within `eps` of a core point but with fewer than `min_samples` neighbors.
- **Noise Points**: Points that don't fit as core or border points and are thus considered outliers.

Parameters of DBSCAN

- **eps (Radius)**: Defines the neighborhood distance.
- **min_samples**: The minimum number of points needed to form a core point.

Strengths and Limitations

- **Strengths**:
  - Suitable for clusters of varying shapes and sizes.
  - Handles noise well, automatically identifying outliers.
- **Limitations**:
  - Struggles with clusters of varying density.
  - Sensitive to `eps` and `min_samples` values, which require tuning.

Evaluating Clustering Results (Advanced/Optional)

Metrics for Clustering Quality

1. **Silhouette Score**:
   - Measures the cohesion within clusters and separation between clusters.
   - Values range from -1 to 1; higher values indicate better-defined clusters.
2. **Davies-Bouldin Index**:
   - Measures the average similarity ratio between clusters, considering intra-cluster and inter-cluster distances.
   - Lower values indicate better clustering.
3. **Adjusted Rand Index (ARI)**:
   - Compares clustering performance with a known "ground truth," accounting for the possibility of random assignments.
   - Commonly used for evaluating clustering on synthetic or labeled datasets.

Practical Demo on Metrics (10-15 mins)

1. **Goal**: Apply these metrics to evaluate the clustering quality for both K-Means and DBSCAN.
2. **Implementation**:
   - Calculate and compare Silhouette Score, Davies-Bouldin Index, and (if a ground truth exists) Adjusted Rand Index.
   - Discuss how each metric reflects clustering quality and provides insights into algorithm performance.

Example Code:

```
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score, davies_bouldin_score,
adjusted_rand_score
from sklearn.datasets import make_blobs

# Generate synthetic dataset
data, labels_true = make_blobs(n_samples=300, centers=4,
random_state=42)

# Apply K-Means
```

```
kmeans = KMeans(n_clusters=4)
kmeans_labels = kmeans.fit_predict(data)

# Apply DBSCAN
dbscan = DBSCAN(eps=0.5, min_samples=5)
dbscan_labels = dbscan.fit_predict(data)

# Evaluate K-Means
print("K-Means Silhouette Score:", silhouette_score(data,
kmeans_labels))
print("K-Means Davies-Bouldin Index:", davies_bouldin_score(data,
kmeans_labels))
print("K-Means Adjusted Rand Index (if ground truth is available):",
```

3. Dimensionality Reduction and Practical Application

Dimensionality reduction is a process in machine learning and data analysis that involves reducing the number of input variables or features in a dataset while preserving as much information as possible.

It is particularly useful when dealing with high-dimensional data, which can pose challenges such as increased computational cost, overfitting, and difficulties in data visualization and interpretation.

Challenges of High-Dimensional Data

- **Curse of Dimensionality**: As the number of features increases, the volume of the space increases, making data sparse and challenging to analyze.
- **Computational Cost**: High-dimensional data increases the time and resources needed for processing and modeling.
- **Overfitting**: Models can become overly complex and fit noise rather than the underlying data distribution.
- **Noise**: More features can introduce irrelevant data, making it harder to detect meaningful patterns.

Benefits of Dimensionality Reduction

- **Simplifies Models**: Reducing the number of features can lead to simpler and more interpretable models.
- **Improves Interpretability**: Fewer dimensions allow for clearer insights and easier understanding of the data.
- **Visualizes High-Dimensional Data**: Enables the visualization of complex datasets in 2D or 3D spaces.

Principal Component Analysis (PCA)

- **Definition**: A technique that transforms data into a lower-dimensional space while retaining as much variance as possible.
- **Purpose**: To simplify datasets by projecting them onto the directions (principal components) where the variance is maximized.

How PCA Works

1. **Standardization**: Scale the data to have a mean of zero and a variance of one.
2. **Covariance Matrix**: Calculate the covariance matrix to understand how variables relate to each other.
3. **Eigenvalues and Eigenvectors**: Compute the eigenvalues and eigenvectors of the covariance matrix, which indicate the direction and magnitude of variance.
4. **Selection of Top Principal Components**: Choose the top components based on eigenvalues to form a new feature space.

Limitations of PCA

- **Captures Only Linear Relationships**: PCA may fail to capture complex, non-linear relationships in the data.
- **Loss of Interpretability**: The principal components may not always be easily interpretable in terms of original features.

---

t-SNE and UMAP (Advanced/Optional)

t-SNE (t-Distributed Stochastic Neighbor Embedding)

- **Purpose**: Primarily used for visualizing high-dimensional data in 2D or 3D space.
- **Steps**:
  - Convert the high-dimensional data into probability distributions.
  - Minimize the divergence between the original and the embedded distribution.
- **Tuning Parameters**:
  - **Perplexity**: Balances attention between local and global aspects of the data.
- **Pros and Cons**:
  - **Pros**: Effective for non-linear data visualizations.
  - **Cons**: Computationally expensive and can be sensitive to parameter settings.

UMAP (Uniform Manifold Approximation and Projection)

- **Definition**: A non-linear dimensionality reduction technique that is often faster and preserves more of the global structure of data than t-SNE.
- **Tuning Parameters**:
  - **n_neighbors**: Affects local versus global structure.
  - **min_dist**: Controls the minimum distance between points in the low-dimensional space.

- **Pros**: Often provides more interpretable results and is scalable to larger datasets.

Resources:

- https://vas3k.com/blog/machine_learning
-

Live Exercise

Now it's your turn!

Task 1: description of task

```
- instructions
```

END

Live Exercise Solutions

In [9]:
```
#solutions
```

Programming Interveiw Questions

What is the primary difference between supervised and unsupervised learning, particularly in terms of data labeling?

Describe the K-Means clustering algorithm. What are the main steps involved in the K-Means algorithm?

What is the purpose of the elbow method in K-Means clustering, and how does it help in selecting the optimal number of clusters?

Explain the concept of hierarchical clustering. What are the two main approaches to hierarchical clustering, and how do they differ?

What is Principal Component Analysis (PCA), and how does it facilitate dimensionality reduction? What are the main steps involved in applying PCA to a dataset?

In [ ]:

Mohammad Idrees Bhat

Tech Skills Trainer | AI/ML Consultant