**🧠 One-Day Workshop: Unsupervised Learning with Python**

**📅 Agenda (One Full Day – ~6 Hours Effective Learning)**

| **Topic** |
| --- |
| Welcome & Introduction to Unsupervised Learning |
| Clustering Techniques: K-Means, Hierarchical, DBSCAN |
|  |
| Dimensionality Reduction: PCA, t-SNE |
| 🍽️ Lunch Break |
| Association Rule Learning (Apriori, Eclat) |
| Real-world Dataset Hands-On (Clustering + PCA) |
|  |
| Evaluation Metrics + Challenges in Unsupervised Learning |
| Quiz + Wrap-Up + Certificate Distribution (if applicable) |

**📘 Theory + Code for Each Topic**

**1️⃣ What is Unsupervised Learning?**

* No labelled data
* Goal: Discover structure/patterns in data
* Applications:
  + Market segmentation
  + Anomaly detection
  + Recommender systems
  + Image compression

**2️⃣ Clustering Algorithms**

**A. K-Means Clustering**

**🧠 Theory:**

* Partitions data into **K clusters**
* Minimizes **intra-cluster distance**
* Uses **centroids**

**🧪 Python Code:**

from sklearn.datasets import make\_blobs

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

X, \_ = make\_blobs(n\_samples=300, centers=4, random\_state=42)

kmeans = KMeans(n\_clusters=4)

y\_kmeans = kmeans.fit\_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans)

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=200, c='red')

plt.title("K-Means Clustering")

plt.show()

**B. Hierarchical Clustering**

**🧠 Theory:**

* Builds a tree of clusters (dendrogram)
* Agglomerative (bottom-up) or Divisive (top-down)

**🧪 Python Code:**

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.datasets import make\_blobs

X, \_ = make\_blobs(n\_samples=100, centers=3)

linked = linkage(X, 'ward')

plt.figure(figsize=(10, 7))

dendrogram(linked)

plt.title("Hierarchical Clustering Dendrogram")

plt.show()

**C. DBSCAN**

**🧠 Theory:**

* Density-based
* No need to specify number of clusters
* Good for irregular shapes

**🧪 Python Code:**

from sklearn.cluster import DBSCAN

from sklearn.datasets import make\_moons

X, \_ = make\_moons(n\_samples=300, noise=0.05)

db = DBSCAN(eps=0.2, min\_samples=5)

labels = db.fit\_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=labels)

plt.title("DBSCAN Clustering")

plt.show()

**3️⃣ Dimensionality Reduction**

**A. PCA – Principal Component Analysis**

**🧠 Theory:**

* Converts correlated features into fewer uncorrelated components
* Used for visualization and speeding up models

**🧪 Python Code:**

from sklearn.decomposition import PCA

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

data = load\_iris()

X = data.data

pca = PCA(n\_components=2)

X\_pca = pca.fit\_transform(X)

plt.scatter(X\_pca[:, 0], X\_pca[:, 1], c=data.target)

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.title("PCA on Iris Dataset")

plt.show()

**B. t-SNE – t-Distributed Stochastic Neighbor Embedding**

**🧠 Theory:**

* Non-linear dimensionality reduction
* Good for visualization of high-dimensional data

**🧪 Python Code:**

from sklearn.manifold import TSNE

from sklearn.datasets import load\_digits

digits = load\_digits()

tsne = TSNE(n\_components=2, perplexity=30, n\_iter=300)

X\_tsne = tsne.fit\_transform(digits.data)

plt.scatter(X\_tsne[:, 0], X\_tsne[:, 1], c=digits.target, cmap='tab10')

plt.title("t-SNE on Digits Dataset")

plt.show()

**4️⃣ Association Rule Learning**

**A. Apriori Algorithm**

**🧠 Theory:**

* Used for market basket analysis
* Finds frequent itemsets and association rules

**🧪 Python Code using mlxtend:**

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

# Sample transactions

dataset = [

['milk', 'bread', 'butter'],

['bread', 'butter'],

['milk', 'bread'],

['milk', 'bread', 'butter'],

['bread', 'butter']

]

df = pd.DataFrame(dataset)

df = df.apply(lambda x: pd.Series(x.dropna().values), axis=1).stack().reset\_index(level=1, drop=True).to\_frame('item')

df['transaction'] = df.index

basket = pd.get\_dummies(df.pivot\_table(index='transaction', columns='item', aggfunc=len, fill\_value=0).astype(bool))

frequent\_itemsets = apriori(basket, min\_support=0.4, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

**5️⃣ Evaluation Metrics for Clustering**

* **Silhouette Score**
* **Davies–Bouldin Index**
* **Inertia (for K-means)**
* **Visual inspection (for low dimensions)**

**🧪 Sample Code:**

from sklearn.metrics import silhouette\_score

score = silhouette\_score(X, y\_kmeans)

print(f"Silhouette Score: {score}")

**6️⃣ Real-World Example Exercise (Hands-On)**

**Dataset: Mall Customers Dataset**

* Features: Age, Annual Income, Spending Score

import pandas as pd

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

df = pd.read\_csv('Mall\_Customers.csv')

X = df[['Annual Income (k$)', 'Spending Score (1-100)']]

kmeans = KMeans(n\_clusters=5)

df['Cluster'] = kmeans.fit\_predict(X)

plt.scatter(X['Annual Income (k$)'], X['Spending Score (1-100)'], c=df['Cluster'], cmap='rainbow')

plt.xlabel("Annual Income")

plt.ylabel("Spending Score")

plt.title("Customer Segmentation")

plt.show()

**📊 Tools/Platforms for Hands-On**

* Google Colab (no setup needed)
* Jupyter Notebook
* Kaggle Notebooks