Unsupervised Model Evaluation

Unsupervised learning differs from supervised learning in that it operates on unlabelled data. The algorithm's goal is to identify patterns, structures, or clusters within the dataset without any predefined labels. Common tasks in unsupervised learning include clustering and dimensionality reduction.

- **Training Process:**
  Training a supervised model involves several key steps:
  1. **Prepare the Data:** Collect and clean the dataset, ensuring that there are no missing values or outliers. Since unsupervised learning works without labels, we focus on the features (X) only.
  2. **Choose the Model:** Select an appropriate unsupervised learning algorithm, such as K-means clustering, DBSCAN, or Principal Component Analysis (PCA).
  3. **Train the Model:** The model learns from the data to uncover hidden structures. In clustering, for example, the algorithm groups data points into clusters based on their similarities.
  4. **Evaluate the Model:** Since unsupervised models do not use labels, evaluation is more subjective and often involves assessing metrics like silhouette score for clustering or variance explained for dimensionality reduction.

  Example:

  Scenario: Clustering Customer Data

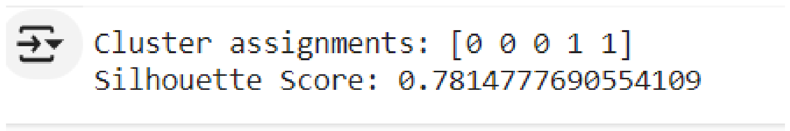
  Objective: Cluster customers into groups based on their purchasing behaviour.

Here is the code for the above objective:

```python
# Import necessary libraries
from sklearn.cluster import KMeans
import numpy as np

# Data: Features (purchase amounts in categories)
X = np.array([[1, 2], [2, 3], [3, 4], [8, 7], [9, 8]])
```

```
# Initialise the KMeans model
kmeans = KMeans(n_clusters=2, random_state=0)

# Train the model
kmeans.fit(X)

# Predict clusters for the data points
clusters = kmeans.predict(X)
# Calculate the silhouette score
silhouette_avg = silhouette_score(X, clusters)



# Print the cluster assignments
print("Cluster assignments:", clusters)
print("Silhouette Score:", silhouette_avg)
```

Output:



```
Cluster assignments: [0 0 0 1 1]
Silhouette Score: 0.7814777690554109
```

Explanation:

Here is a breakdown of the code step by step:

1. **Import Necessary Libraries**
   - **KMeans from sklearn.cluster:** This imports the K-Means clustering algorithm, which is used to group data points (customers, in this case) into clusters based on their feature similarity.
   - **numpy:** A Python library used to handle arrays and perform numerical operations. In this case, it is used to create the data for clustering.

2. **Create the Data**
   - **X:** This is a numpy array that holds the features of the data points (or customers). Each row in X represents a customer, and the columns represent different features (for example, purchase amounts in different categories).

Customer	Feature 1	Feature 2
1	1	2
2	2	3
3	3	4
4	8	7
5	9	8

Here, each customer is described by two features (e.g., spending habits in two different categories).

3. Initialise the KMeans Model

- **KMeans(n_clusters=2):** This initialises the K-Means clustering algorithm. You are asking K-Means to divide the data into 2 clusters (n_clusters=2).
- K-Means works by grouping data points into clusters based on their proximity to the cluster's centroid (center point).
- **random_state=0:** This is to ensure reproducibility. It makes sure that if you run the code multiple times, you get the same results by controlling the random initialisation of centroids.

4. Train the Model (Fit the Data)

- **fit(X):** This trains the K-Means model on the data X. The K-Means algorithm:
  - Starts by randomly initialising 2 centroids (because n_clusters=2).
  - Assigns each data point (customer) to the nearest centroid.
  - Recomputes the centroids by averaging the points in each cluster.
  - Repeats the process until the centroids stabilise (don’t change much).

5. Predict Cluster Assignments for Each Data Point

- **predict(X):** After training the K-Means model, this method assigns each customer to the nearest cluster.
  - The result is stored in clusters, which contains the cluster labels (either 0 or 1).
  - These labels represent the group each customer belongs to.

6. **Evaluate the Silhouette Score**
   - **silhouette_score(X, clusters):**
     - **X:** This is the input data, a collection of feature vectors (e.g., purchase amounts in categories).
     - **clusters:** These are the predicted cluster labels (assigned by a clustering algorithm like KMeans).

7. **Print the Cluster Assignments**
   This prints the cluster labels for each data point (customer). For example, the output might look like this:

   Cluster assignments: [0 0 0 1 1]

   This means:

   - Customers 1, 2, and 3 belong to Cluster 0.
   - Customers 4 and 5 belong to Cluster 1.

   This code performs K-Means clustering on customer data with two