Common Unsupervised Algorithms

Unsupervised learning is primarily categorised into two types:

- Clustering Algorithms
- Dimensionality Reduction

1. Clustering Algorithms

Clustering algorithms group data points into clusters based on their similarity. The goal is to partition data into subsets where points in the same group are more similar to each other than to those in other groups.

- **Purpose:** To group similar data points into clusters.
- **Output:** Groups of similar data points (clusters).

Common Clustering Algorithms:

- **K-Means:** Divides data into K clusters by minimising the variance within each cluster.
- **Hierarchical Clustering:** Builds a hierarchy of clusters by iteratively merging or splitting clusters.
- **DBSCAN:** Groups data based on density, useful for detecting noise or outliers.
- **Gaussian Mixture Models (GMM):** Assumes that the data is generated from a mixture of several Gaussian distributions.

2. Dimensionality Reduction

Dimensionality reduction algorithms simplify the dataset by reducing the number of features while retaining important information. This helps in visualizing or processing large, complex data more efficiently.

- **Purpose:** To reduce the number of features while preserving the most relevant information.
- **Output:** Fewer features that still represent the data effectively.

Common Dimensionality Reduction Algorithms:

- **Principal Component Analysis (PCA):** Reduces the dimensionality by transforming the data into a set of linearly uncorrelated components.
- **t-SNE:** A nonlinear technique for visualising high-dimensional data in 2D or 3D.
- **Autoencoders:** Neural networks that compress data into a lower dimension and then reconstruct it, often used in deep learning.

Unsupervised learning focuses on discovering the structure of data through clustering or reducing complexity without needing predefined labels.

Example 1:

Python code using K-Means Clustering to group customers based on similar buying habits:

```
# Import necessary libraries
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Sample dataset representing customer buying habits
(e.g., frequency, total amount spent, items purchased)
# In a real-world scenario, this data would be loaded from
a file or database
data = {'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Total_Spent': [200, 450, 300, 500, 700, 1000,
400, 650, 1200, 900],
        'Frequency': [5, 8, 6, 9, 12, 15, 7, 11, 18, 14],
        'Items_Purchased': [12, 20, 14, 22, 30, 50, 18,
26, 55, 40]}

# Convert data to a DataFrame
df = pd.DataFrame(data)

# Features (without the CustomerID)
X = df[['Total_Spent', 'Frequency', 'Items_Purchased']]

# Standardize the data to bring features to the same scale
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform K-Means clustering (let's assume 3 clusters)
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Visualize the clusters using a pairplot
sns.pairplot(df, hue='Cluster', vars=['Total_Spent',
'Frequency', 'Items_Purchased'])
plt.show()

# Print the cluster centers and the cluster assignment for
each customer
print("Cluster Centers:\n", kmeans.cluster_centers_)
print("\nCluster Assignment:\n", df[['CustomerID',
'Cluster']])
```

Output:

```
Cluster Centers:
 [[-0.84983659 -0.86824314 -0.81072406]
 [ 1.31833624  1.28169226  1.38410571]
 [ 0.1470871   0.24806947 -0.04934842]]

Cluster Assignment:
    CustomerID  Cluster
0           1        0
1           2        0
2           3        0
3           4        0
4           5        2
5           6        1
6           7        0
7           8        2
8           9        1
9          10        1
```

Explanation

1. **Import Necessary Libraries:**
   - **pandas:** For data manipulation and analysis.
   - **KMeans from sklearn.cluster:** For performing K-Means clustering.
   - **StandardScaler from sklearn.preprocessing:** For standardising features.
   - **matplotlib.pyplot and seaborn:** For visualising data.

2. **Sample Dataset:**
   - **Data Creation:** Represents customer buying habits. Each customer has attributes such as Total_Spent, Frequency, and Items_Purchased.

3. **Convert Data to DataFrame:**
   - **DataFrame Creation:** Converts the dictionary data into a DataFrame for easier manipulation and analysis.

4. **Feature Selection:**
   - **Feature Extraction:** Selects the relevant columns for clustering analysis, excluding CustomerID since it's an identifier, not a feature.

5. **Standardise the Data:**
   - **Standardisation:** Scales the features to have a mean of 0 and a standard deviation of 1. This step ensures that each feature contributes equally to the clustering algorithm.

6. **Perform K-Means Clustering:**
   - **K-Means Initialisation:** Initialises the K-Means algorithm to create 3 clusters.
   - **Fit and Predict:** Applies K-Means clustering to the standardized data and assigns each customer to one of the 3 clusters. The cluster assignment is stored in a new column Cluster in the DataFrame.

7. **Visualise the Clusters:**
   - **Pairplot Visualisation:** Creates a pairplot to visualise the clusters. Each point is colored according to its cluster assignment, and pairwise relationships between features (Total_Spent, Frequency, Items_Purchased) are plotted.

8. **Print Cluster Centers and Assignments:**
   - **Cluster Centers:** Prints the coordinates of the cluster centers, which represent the centroid of each cluster in the feature space.
   - **Cluster Assignments:** Displays the CustomerID along with the cluster assignment, showing which cluster each customer belongs to.

Example 2:

Python code to perform Principal Component Analysis (PCA) on a dataset representing customer attributes:

```
# Import necessary libraries
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Sample dataset representing customer buying habits (e.g.,
frequency, total amount spent, items purchased)
# In a real-world scenario, this data would be loaded from
a file or database
data = {'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Total_Spent': [200, 450, 300, 500, 700, 1000, 400,
650, 1200, 900],
        'Frequency': [5, 8, 6, 9, 12, 15, 7, 11, 18, 14],
        'Items_Purchased': [12, 20, 14, 22, 30, 50, 18, 26,
55, 40]}

# Convert data to a DataFrame
df = pd.DataFrame(data)

# Features (without the CustomerID)
X = df[['Total_Spent', 'Frequency', 'Items_Purchased']]

# Standardize the data to bring features to the same scale
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform K-Means clustering (let's assume 3 clusters)
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Visualize the clusters using a pairplot
sns.pairplot(df, hue='Cluster', vars=['Total_Spent',
'Frequency', 'Items_Purchased'])
plt.show()

# Print the cluster centers and the cluster assignment for
each customer
print("Cluster Centers:\n", kmeans.cluster_centers_)
print("\nCluster Assignment:\n", df[['CustomerID',
'Cluster']])
```

Output:

```
Cluster Centers:
 [[-0.84983659 -0.86824314 -0.81072406]
 [ 1.31833624  1.28169226  1.38410571]
 [ 0.1470871   0.24806947 -0.04934842]]

Cluster Assignment:
    CustomerID  Cluster
0           1        0
1           2        0
2           3        0
3           4        0
4           5        2
5           6        1
6           7        0
7           8        2
8           9        1
9          10        1
```

Explanation

1. **Import Necessary Libraries:**
   - **pandas:** For data manipulation and analysis.
   - **KMeans from sklearn.cluster:** For performing K-Means clustering.
   - **StandardScaler from sklearn.preprocessing:** For standardising features.
   - **matplotlib.pyplot and seaborn**: For visualising data.
2. **Sample Dataset:**
   - **Data Creation:** Represents customer buying habits. Each customer has attributes such as Total_Spent, Frequency, and Items_Purchased.
3. **Convert Data to DataFrame:**
   - **DataFrame Creation:** Converts the dictionary data into a DataFrame for easier manipulation and analysis.
4. **Feature Selection:**
   - **Feature Extraction:** Selects the relevant columns for clustering analysis, excluding CustomerID since it's an identifier, not a feature.

5. Standardise the Data:

- **Standardisation:** Scales the features to have a mean of 0 and a standard deviation of 1. This step ensures that each feature contributes equally to the clustering algorithm.

6. Perform K-Means Clustering:

- **K-Means Initialisation:** Initialises the K-Means algorithm to create 3 clusters.
- **Fit and Predict:** Applies K-Means clustering to the standardized data and assigns each customer to one of the 3 clusters. The cluster assignment is stored in a new column Cluster in the DataFrame.

7. Visualise the Clusters:

- **Pairplot Visualisation:** Creates a pairplot to visualise the clusters. Each point is colored according to its cluster assignment, and pairwise relationships between features (Total_Spent, Frequency, Items_Purchased) are plotted.

8. Print Cluster Centers and Assignments:

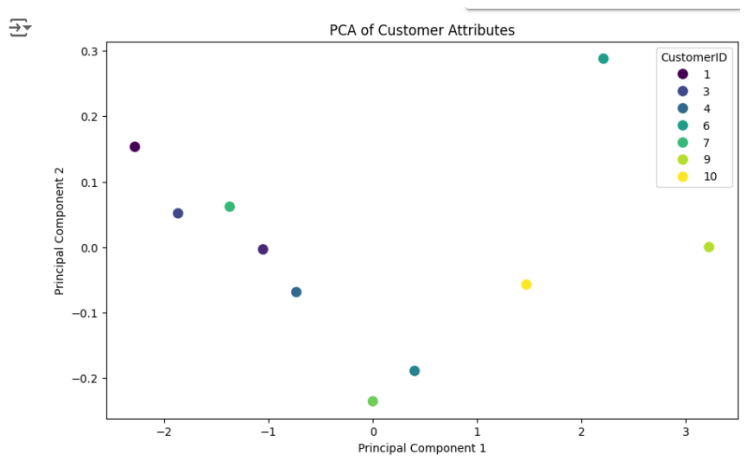
- **Cluster Centers:** Prints the coordinates of the cluster centers, which represent the centroid of each cluster in the feature space.
- **Cluster Assignments:** Displays the CustomerID along with the cluster assignment, showing which cluster each customer belongs to.

Example 2:

Python code to perform Principal Component Analysis (PCA) on a dataset representing customer attributes:

```
# Import necessary libraries
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns

# Sample dataset representing customer attributes
# In a real-world scenario, this data would be loaded from
a file or database
data = {'CustomerID': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
 'Total_Spent': [200, 450, 300, 500, 700, 1000, 400, 650,
1200, 900],
```

```
'Frequency': [5, 8, 6, 9, 12, 15, 7, 11, 18, 14],
'Items_Purchased': [12, 20, 14, 22, 30, 50, 18, 26, 55,
40]}

# Convert data to a DataFrame
df = pd.DataFrame(data)

# Features (without the CustomerID)
X = df[['Total_Spent', 'Frequency', 'Items_Purchased']]

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Perform PCA
pca = PCA(n_components=2)  # Reduce to 2 dimensions for
visualization
X_pca = pca.fit_transform(X_scaled)

# Create a DataFrame with the principal components
df_pca = pd.DataFrame(data=X_pca, columns=['PC1', 'PC2'])
df_pca['CustomerID'] = df['CustomerID']

# Plot the principal components
plt.figure(figsize=(10, 6))
sns.scatterplot(x='PC1', y='PC2', data=df_pca,
hue='CustomerID', palette='viridis', s=100)
plt.title('PCA of Customer Attributes')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.legend(title='CustomerID')
plt.show()

# Print the explained variance ratios for the principal
components
print("Explained Variance Ratio of Principal
Components:\n", pca.explained_variance_ratio_)

# Print the components (directions of maximum variance)
print("\nPrincipal Components:\n", pca.components_)
```




```
Explained Variance Ratio of Principal Components:
 [0.99242003 0.00708463]

Principal Components:
 [[ 0.5789811   0.5774157   0.5756492 ]
 [-0.22238872 -0.56741524  0.79283239]]
```

Explanation

1. **Importing Libraries:**
   - **pandas:** Used for data manipulation and analysis. Here, it's used to create and handle the dataset.
   - **PCA from sklearn.decomposition:** Performs Principal Component Analysis.
   - **StandardScaler from sklearn.preprocessing:** Standardises features to have a mean of 0 and a standard deviation of 1.
   - **matplotlib.pyplot** and **seaborn:** Libraries for data visualization.

2. **Creating a Sample Dataset:**
   - **Dataset Creation: We create a sample DataFrame df with customer attributes:** Total_Spent, Frequency, and Items_Purchased. CustomerID is included for later reference but not used in PCA.

3. **Feature Selection:**
   - **Feature Extraction:** We select the columns that will be used for PCA analysis. These features are Total_Spent, Frequency, and Items_Purchased.

4. **Standardising the Data:**
   - **Standardisation:** PCA is sensitive to the scale of the data. We use StandardScaler to standardise the features so they have a mean of 0 and a standard deviation of 1. This ensures that each feature contributes equally to the analysis.

5. Applying PCA:

- **PCA Initialisation:** We initialise PCA to reduce the dimensionality to 2 components for easier visualisation.
- **Fit and Transform:** We fit the PCA model to the standardized data and transform it to obtain the principal components.

6. Creating a DataFrame for PCA Results:

- **Results DataFrame:** We create a DataFrame df_pca containing the principal components PC1 and PC2, along with the CustomerID for reference.

7. Visualisation:

- **Plotting:** We use seaborn to create a scatter plot of the first two principal components. Each point represents a customer, and the color (hue) is used to distinguish between different CustomerIDs.

8. Explained Variance Ratios and Principal Components:

- **Explained Variance Ratio:** Shows how much variance is captured by each principal component. This helps understand the importance of each component.
- **Principal Components:** Displays the directions of maximum variance in the data. Each component is a vector showing the weight of each feature.