To perform clustering analysis on the provided dataset (EastWestAirlines.xlsx), we will follow the steps outlined in the assignment. Below is a detailed guide on how to approach this task using Python and libraries like `pandas`, `scikit-learn`, `matplotlib`, and `seaborn`.

### Step 1: Data Preprocessing

1. \*\*Load the Dataset:\*\*

- Use `pandas` to load the Excel file and inspect the data.

```python

import pandas as pd

# Load the dataset

df = pd.read\_excel('EastWestAirlines.xlsx', sheet\_name='data')

# Display the first few rows of the dataset

print(df.head())

```

2. \*\*Handle Missing Values:\*\*

- Check for missing values and handle them appropriately (e.g., imputation or removal).

```python

# Check for missing values

print(df.isnull().sum())

# Handle missing values (if any)

df = df.dropna() # or use df.fillna() for imputation

```

3. \*\*Remove Outliers:\*\*

- Use statistical methods or visualization techniques to identify and remove outliers.

```python

import seaborn as sns

import matplotlib.pyplot as plt

# Visualize outliers using boxplots

plt.figure(figsize=(12, 6))

sns.boxplot(data=df)

plt.show()

# Remove outliers (example using IQR)

Q1 = df.quantile(0.25)

Q3 = df.quantile(0.75)

IQR = Q3 - Q1

df = df[~((df < (Q1 - 1.5 \* IQR)) | (df > (Q3 + 1.5 \* IQR)).any(axis=1)]

```

4. \*\*Scale the Features:\*\*

- Normalize or standardize the features to ensure that all features contribute equally to the clustering process.

```python

from sklearn.preprocessing import StandardScaler

# Scale the features

scaler = StandardScaler()

df\_scaled = scaler.fit\_transform(df)

```

### Step 2: Exploratory Data Analysis (EDA)

1. \*\*Distribution of Data:\*\*

- Use histograms, pair plots, or other visualization techniques to understand the distribution of the data.

```python

# Pairplot to visualize relationships between features

sns.pairplot(df)

plt.show()

```

2. \*\*Correlation Matrix:\*\*

- Check for correlations between features.

```python

# Correlation matrix

corr\_matrix = df.corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm')

plt.show()

```

### Step 3: Implementing Clustering Algorithms

1. \*\*K-Means Clustering:\*\*

- Use the Elbow method to determine the optimal number of clusters (K).

```python

from sklearn.cluster import KMeans

import numpy as np

# Elbow method to find optimal K

inertia = []

K\_range = range(1, 11)

for k in K\_range:

kmeans = KMeans(n\_clusters=k, random\_state=42)

kmeans.fit(df\_scaled)

inertia.append(kmeans.inertia\_)

# Plot the Elbow curve

plt.figure(figsize=(8, 5))

plt.plot(K\_range, inertia, marker='o')

plt.xlabel('Number of clusters (K)')

plt.ylabel('Inertia')

plt.title('Elbow Method')

plt.show()

# Apply K-Means with optimal K

kmeans = KMeans(n\_clusters=3, random\_state=42)

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

```

2. \*\*Hierarchical Clustering:\*\*

- Use different linkage criteria (e.g., single, complete, average) and visualize the dendrogram.

```python

from scipy.cluster.hierarchy import dendrogram, linkage

# Perform hierarchical clustering

linked = linkage(df\_scaled, method='ward')

# Plot the dendrogram

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

dendrogram(linked, orientation='top', distance\_sort='descending', show\_leaf\_counts=True)

plt.show()

# Apply Agglomerative Clustering

from sklearn.cluster import AgglomerativeClustering

agglo = AgglomerativeClustering(n\_clusters=3, affinity='euclidean', linkage='ward')

df['Cluster'] = agglo.fit\_predict(df\_scaled)

```

3. \*\*DBSCAN Clustering:\*\*

- Experiment with different values of `eps` and `min\_samples`.

```python

from sklearn.cluster import DBSCAN

# Apply DBSCAN

dbscan = DBSCAN(eps=0.5, min\_samples=5)

df['Cluster'] = dbscan.fit\_predict(df\_scaled)

# Check the number of clusters

print(f"Number of clusters: {len(set(df['Cluster'])) - (1 if -1 in df['Cluster'] else 0)}")

```

### Step 4: Cluster Analysis and Interpretation

1. \*\*Analyze Clusters:\*\*

- For each clustering algorithm, analyze the characteristics of the clusters by examining the mean values of the features within each cluster.

```python

# Analyze clusters for K-Means

print(df.groupby('Cluster').mean())

```

2. \*\*Interpretation:\*\*

- Write insights based on the cluster analysis. For example, if one cluster has high values for `Balance` and `Bonus\_miles`, it might represent frequent flyers.

### Step 5: Visualization

1. \*\*Scatter Plots:\*\*

- Visualize the clusters using scatter plots.

```python

# Scatter plot for K-Means

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

sns.scatterplot(x=df['Balance'], y=df['Bonus\_miles'], hue=df['Cluster'], palette='viridis')

plt.title('K-Means Clustering')

plt.show()

```

2. \*\*Pair Plots with Clusters:\*\*

- Use pair plots to visualize the clusters in multiple dimensions.

```python

# Pair plot with clusters

sns.pairplot(df, hue='Cluster', palette='viridis')

plt.show()

```

### Step 6: Evaluation and Performance Metrics

1. \*\*Silhouette Score:\*\*

- Evaluate the quality of clustering using the silhouette score.

```python

from sklearn.metrics import silhouette\_score

# Silhouette score for K-Means

silhouette\_avg = silhouette\_score(df\_scaled, df['Cluster'])

print(f"Silhouette Score: {silhouette\_avg}")

```

2. \*\*Compare Algorithms:\*\*

- Compare the performance of K-Means, hierarchical clustering, and DBSCAN using the silhouette score.

```python

# Silhouette score for DBSCAN

silhouette\_avg\_dbscan = silhouette\_score(df\_scaled, df['Cluster'])

print(f"Silhouette Score for DBSCAN: {silhouette\_avg\_dbscan}")

```

### Summary

- \*\*K-Means:\*\* Easy to implement and works well with spherical clusters. The Elbow method helps in determining the optimal number of clusters.

- \*\*Hierarchical Clustering:\*\* Provides a dendrogram that helps in understanding the hierarchy of clusters. Different linkage criteria can be experimented with.

- \*\*DBSCAN:\*\* Effective for identifying noise and clusters of varying densities. Requires careful tuning of `eps` and `min\_samples`.

### Insights

- \*\*Cluster Characteristics:\*\* Based on the mean values of features within each cluster, you can identify groups such as frequent flyers, occasional travelers, and non-frequent users.

- \*\*Visualization:\*\* Scatter plots and pair plots help in visualizing the separation of clusters and understanding the relationships between features.

By following these steps, you can effectively perform clustering analysis on the EastWestAirlines dataset and gain valuable insights into customer segments.