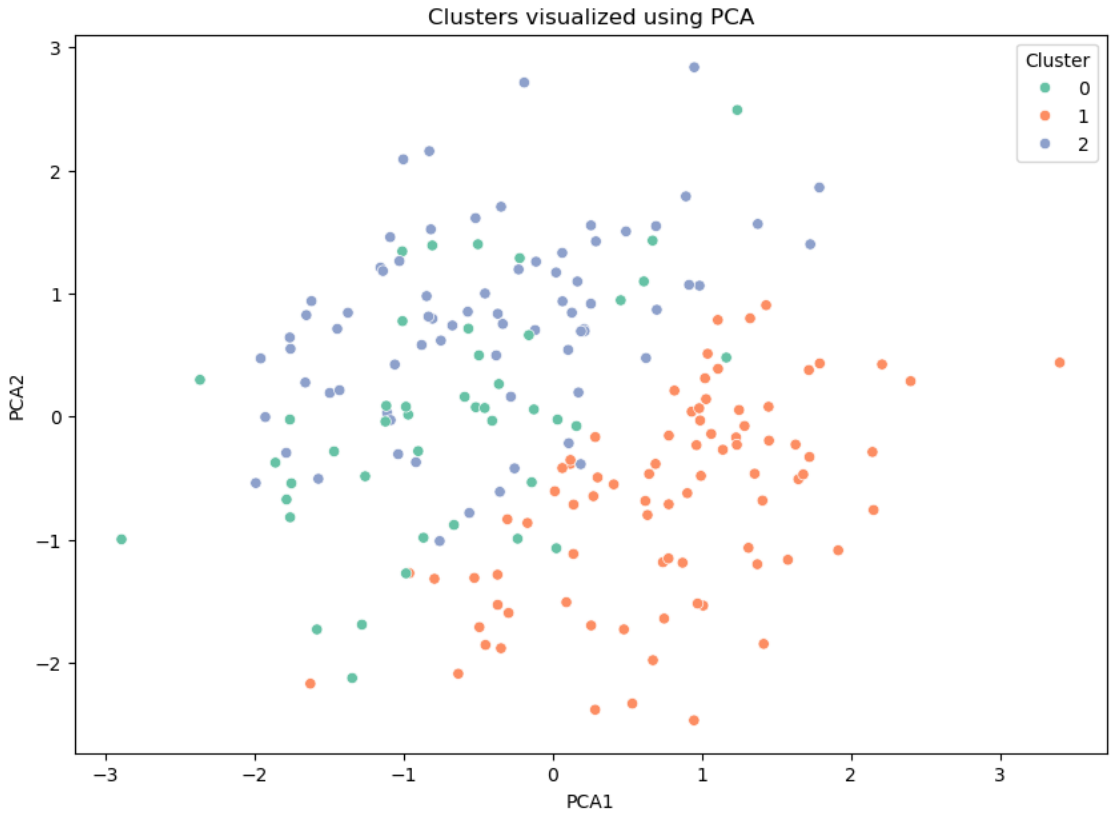
untitled3

September 17, 2024

[1]:
```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler

# Load the dataset
file_path = 'C:/Users/Neha Thakur/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Normalize the data for clustering and PCA (excluding non-numerical columns if␣
 ↪any)
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Perform KMeans clustering
kmeans = KMeans(n_clusters=3, random_state=42)  # Assume 3 clusters for␣
 ↪demonstration
data['Cluster'] = kmeans.fit_predict(data_scaled)

# Perform PCA for dimensionality reduction
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data_scaled)

# Add PCA components to the data
data['PCA1'] = data_pca[:, 0]
data['PCA2'] = data_pca[:, 1]

# Visualize clusters using PCA components
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=data, palette='Set2')
plt.title('Clusters visualized using PCA')
plt.show()
```

```
C:\Users\Neha Thakur\anaconda3\Lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
```

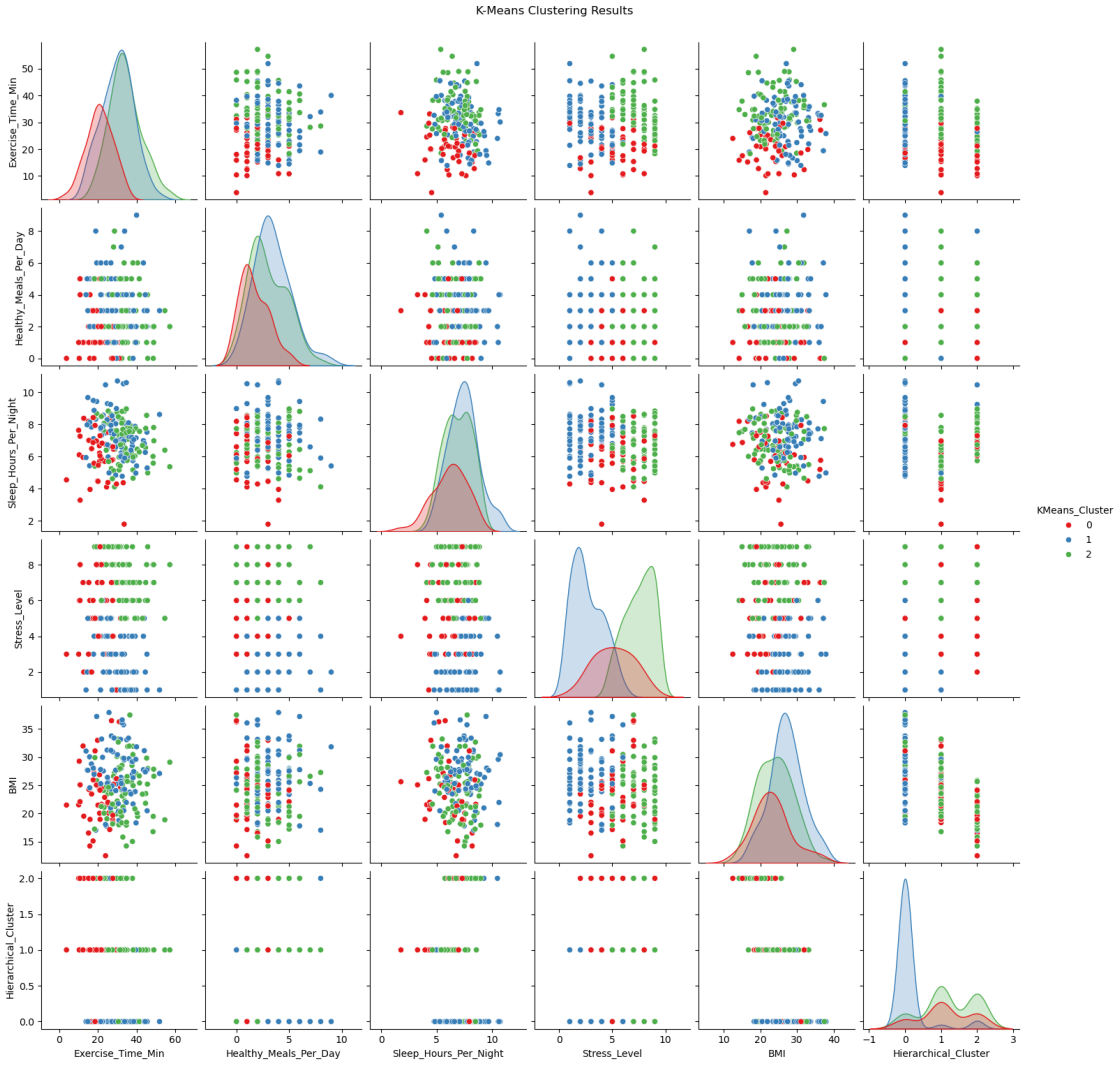
```
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

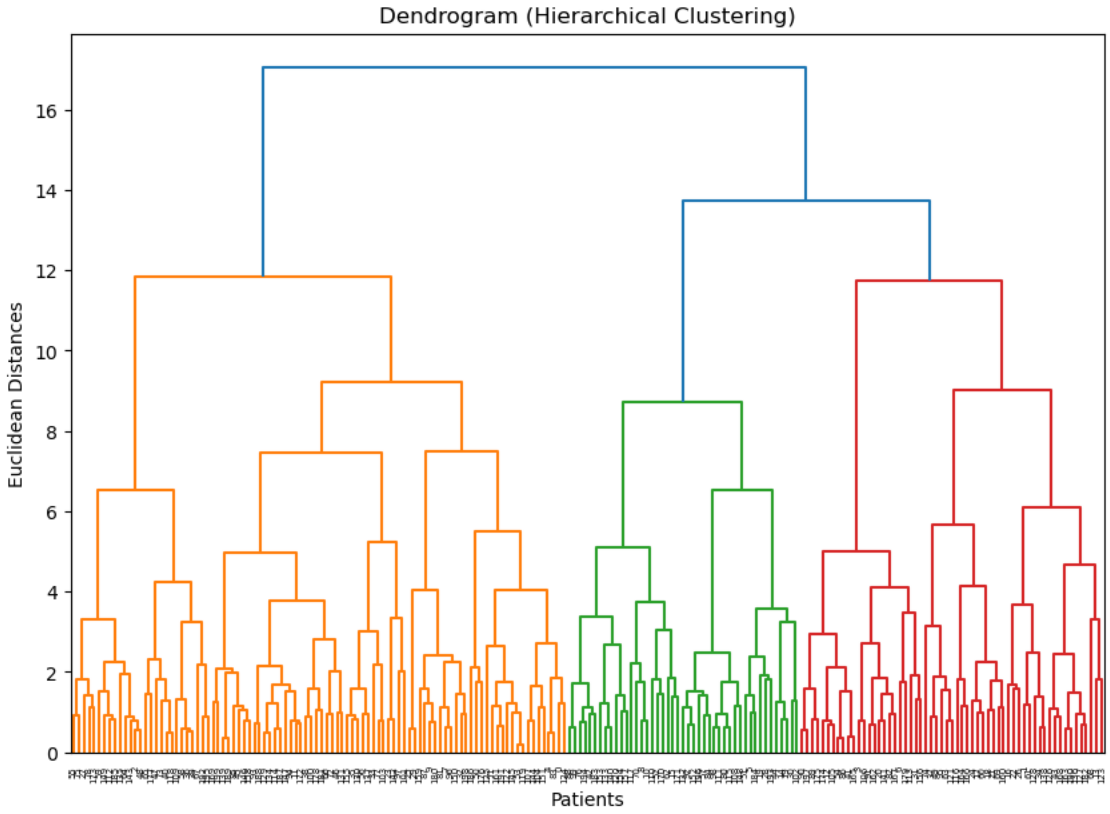


[3]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.cluster.hierarchy as sch

# Load the dataset
file_path = 'C:/Users/Neha Thakur/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)  # Assuming 3 clusters
data['KMeans_Cluster'] = kmeans.fit_predict(data_scaled)
```

```
# Hierarchical Clustering (Agglomerative)
hierarchical = AgglomerativeClustering(n_clusters=3)
data['Hierarchical_Cluster'] = hierarchical.fit_predict(data_scaled)

# Visualize KMeans Clusters using pairplot
sns.pairplot(data, hue='KMeans_Cluster', palette='Set1')
plt.suptitle('K-Means Clustering Results', y=1.02)
plt.show()

# Visualize Hierarchical Clusters using Dendrogram
plt.figure(figsize=(10, 7))
dendrogram = sch.dendrogram(sch.linkage(data_scaled, method='ward'))
plt.title('Dendrogram (Hierarchical Clustering)')
plt.xlabel('Patients')
plt.ylabel('Euclidean Distances')
plt.show()

# Analyze cluster characteristics for KMeans
cluster_analysis = data.groupby('KMeans_Cluster').mean()
print("KMeans Cluster Analysis:\n", cluster_analysis)

# Analyze cluster characteristics for Hierarchical
hierarchical_analysis = data.groupby('Hierarchical_Cluster').mean()
print("Hierarchical Cluster Analysis:\n", hierarchical_analysis)
```

```
C:\Users\Neha Thakur\anaconda3\Lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```


K-Means Clustering Results
Exercise_Time_Min
Healthy_Meals_Per_Day
Sleep_Hours_Per_Night
Stress_Level
BMI
Hierarchical_Cluster
KMeans_Cluster
0
1
2



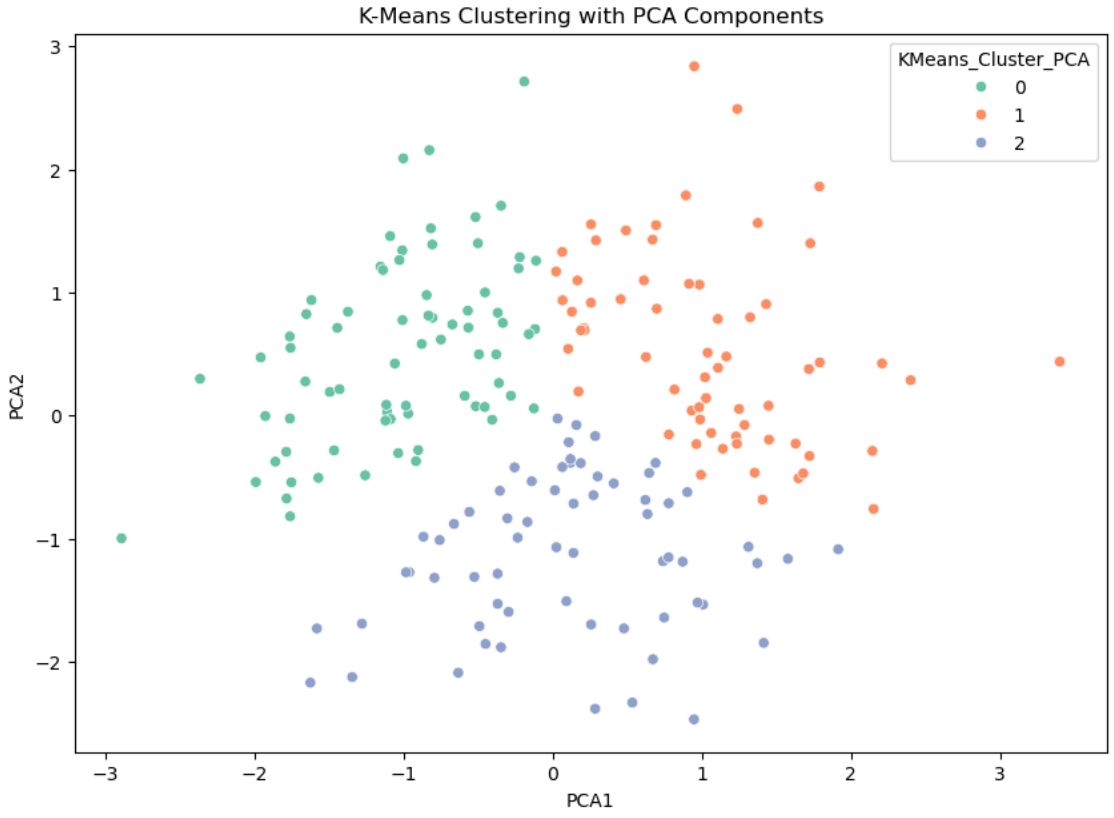

```
KMeans Cluster Analysis:
                 Exercise_Time_Min  Healthy_Meals_Per_Day  \
KMeans_Cluster
0                        20.882706               1.704545
1                        30.153795               3.414634
2                        34.148754               2.972973

                Sleep_Hours_Per_Night  Stress_Level        BMI  \
KMeans_Cluster
0                            6.169744      5.136364  23.233885
1                            7.385093      2.670732  27.124795
2                            6.887434      7.486486  24.101048

                Hierarchical_Cluster
KMeans_Cluster
0                           1.113636
1                           0.121951
2                           1.216216
Hierarchical Cluster Analysis:
                      Exercise_Time_Min  Healthy_Meals_Per_Day  \
Hierarchical_Cluster
```

```
0                                30.450182              3.187500
1                                31.139440              2.288136
2                                25.733636              2.977778

                        Sleep_Hours_Per_Night  Stress_Level        BMI  \
Hierarchical_Cluster
0                                    7.302486      3.322917  27.761629
1                                    5.707485      6.338983  25.025856
2                                    7.754138      6.800000  19.741329

                        KMeans_Cluster
Hierarchical_Cluster
0                             1.041667
1                             1.186441
2                             1.333333
```

[5]:
```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = 'C:/Users/Neha Thakur/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# Perform PCA
pca = PCA(n_components=2)  # Reducing to 2 components for visualization
data_pca = pca.fit_transform(data_scaled)

# Convert PCA result into a DataFrame
pca_df = pd.DataFrame(data_pca, columns=['PCA1', 'PCA2'])

# Add back to original data for analysis
data['PCA1'] = pca_df['PCA1']
data['PCA2'] = pca_df['PCA2']

# K-Means Clustering on PCA-reduced data
kmeans_pca = KMeans(n_clusters=3, random_state=42)
data['KMeans_Cluster_PCA'] = kmeans_pca.fit_predict(pca_df)

# Visualize the clusters after PCA dimensionality reduction
```

```
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PCA1', y='PCA2', hue='KMeans_Cluster_PCA', data=data,␣
 ↪palette='Set2')
plt.title('K-Means Clustering with PCA Components')
plt.show()

# Evaluate the explained variance ratio by each PCA component
explained_variance = pca.explained_variance_ratio_
print(f"Explained variance by each PCA component: {explained_variance}")

# Analysis: KMeans Clustering on PCA-reduced data
cluster_analysis_pca = data.groupby('KMeans_Cluster_PCA').mean()
print("Cluster Analysis with PCA:\n", cluster_analysis_pca)
```

```
C:\Users\Neha Thakur\anaconda3\Lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available
threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```




```
Explained variance by each PCA component: [0.23691549 0.22082517]
```

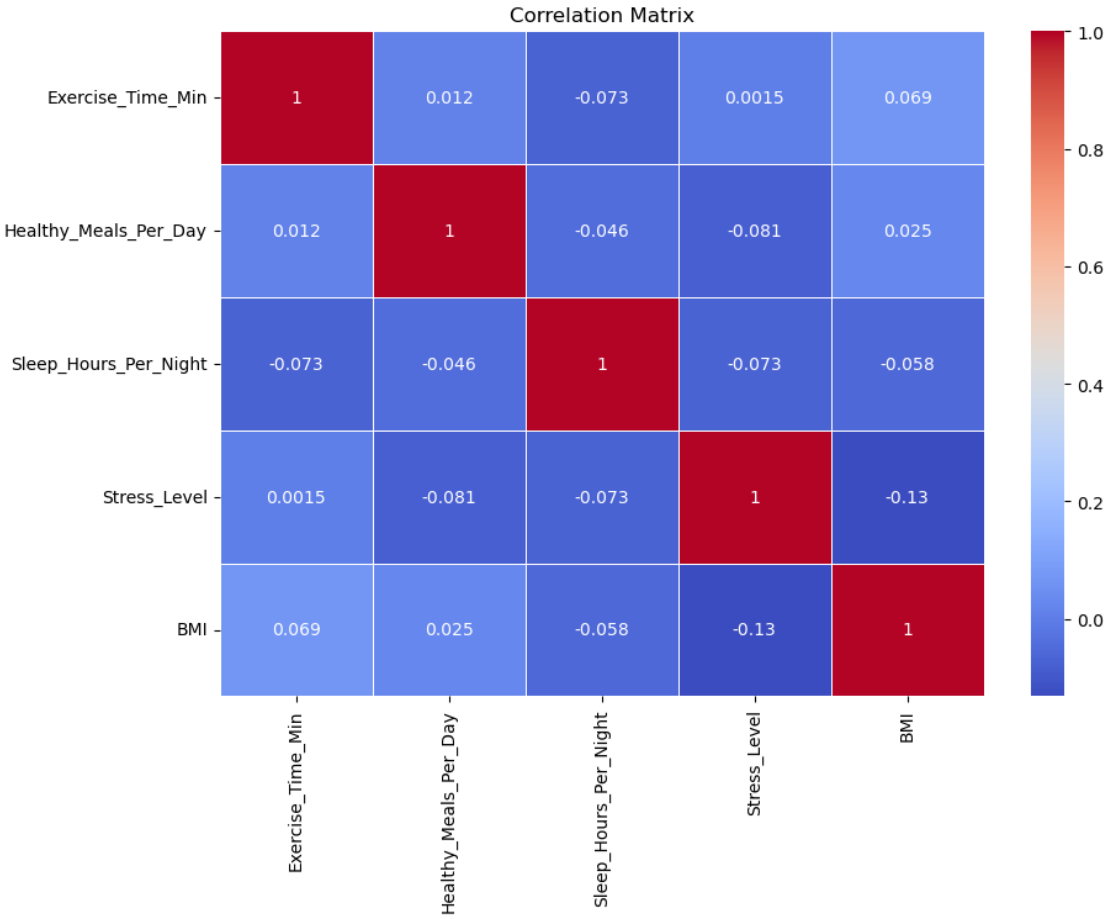
```
Cluster Analysis with PCA:
                     Exercise_Time_Min  Healthy_Meals_Per_Day  \
KMeans_Cluster_PCA
0                            27.701378               2.027778
1                            36.180755               3.515625
2                            25.131102               3.187500

                    Sleep_Hours_Per_Night  Stress_Level        BMI      PCA1  \
KMeans_Cluster_PCA
0                                6.645553      7.125000  22.117568 -1.022315
1                                6.151486      4.296875  28.404155  1.052000
2                                8.039711      3.296875  25.307357  0.098104

                        PCA2
KMeans_Cluster_PCA
0                   0.518794
1                   0.547539
2                  -1.131182
```

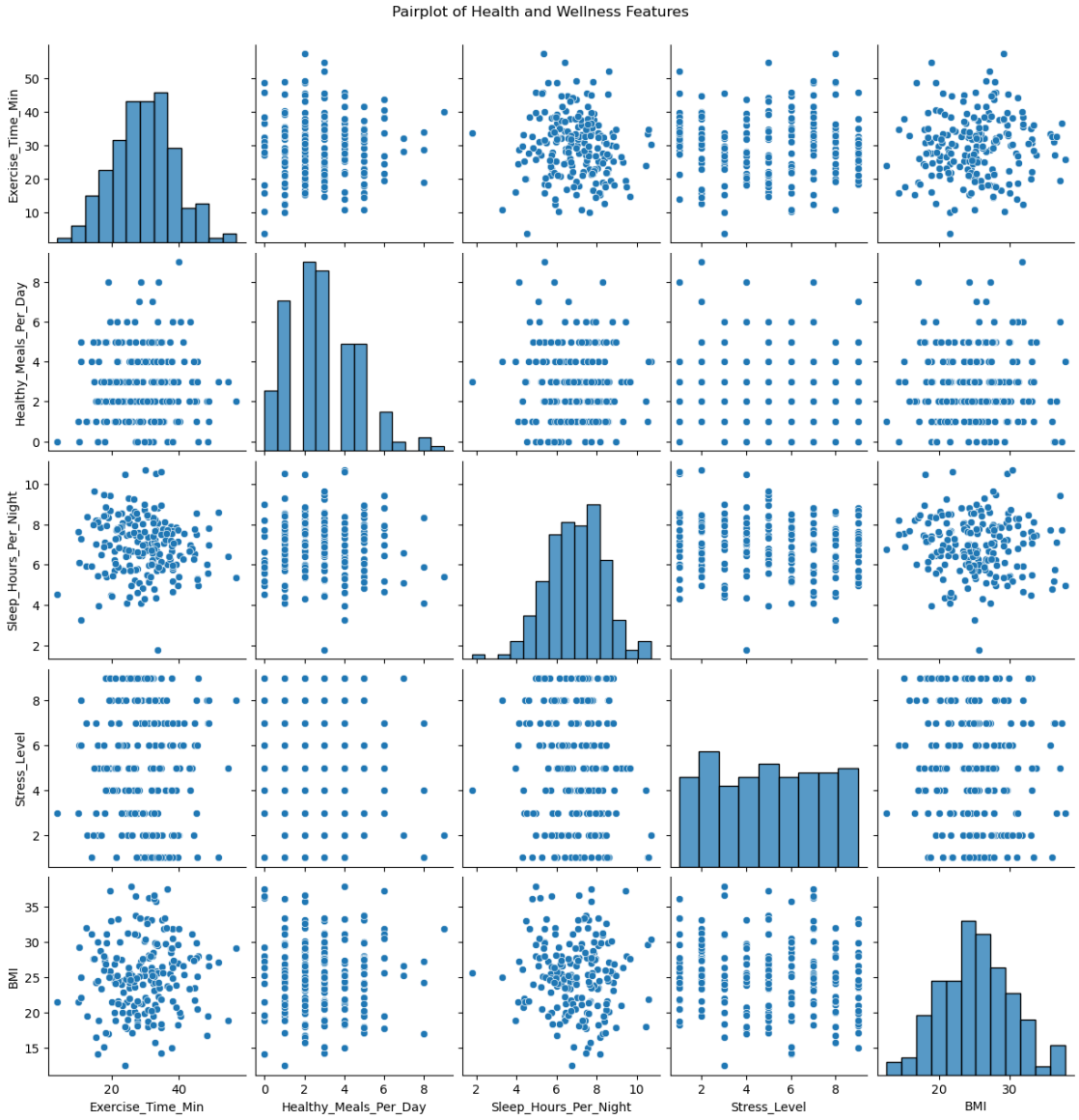
[9]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px

# Load the dataset
file_path = 'C:/Users/Neha Thakur/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Display basic information about the dataset
print("Dataset Information:")
print(data.info())

# Descriptive statistics
print("\nDescriptive Statistics:")
print(data.describe())

# Correlation matrix
plt.figure(figsize=(10, 7))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()

# Pairplot to visualize relationships between variables
sns.pairplot(data)
plt.suptitle('Pairplot of Health and Wellness Features', y=1.02)
plt.show()
```

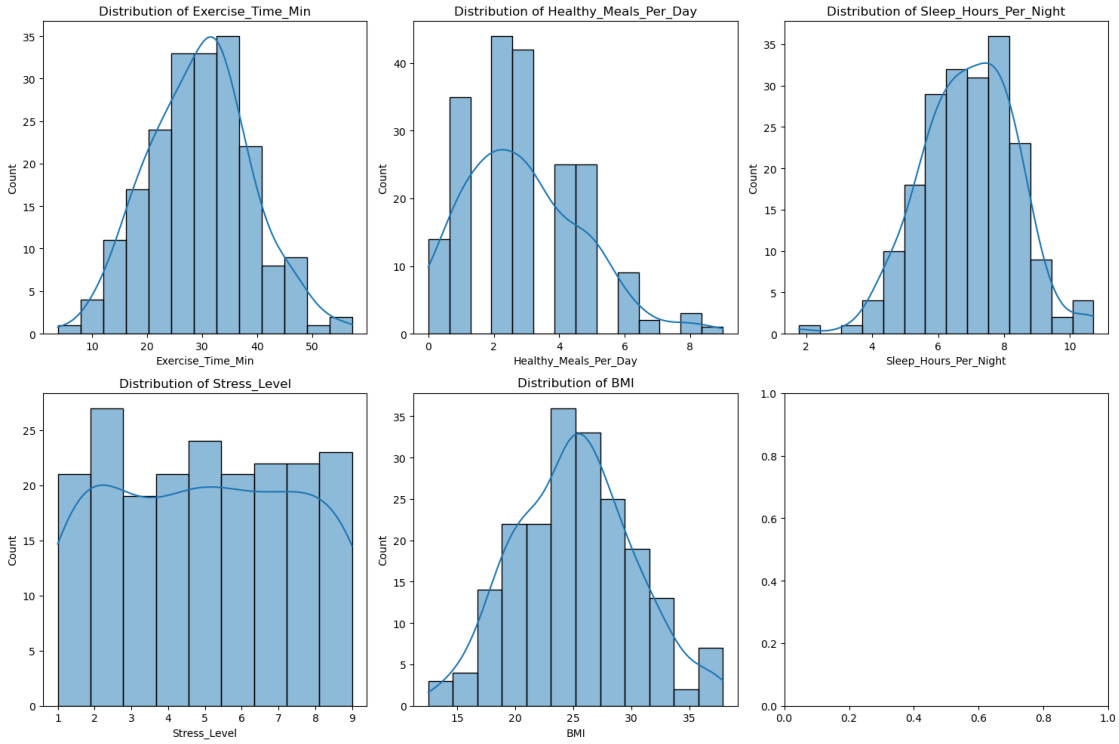
```
# Distribution of each feature
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
axes = axes.flatten()
for i, col in enumerate(data.columns):
    sns.histplot(data[col], kde=True, ax=axes[i])
    axes[i].set_title(f'Distribution of {col}')
plt.tight_layout()
plt.show()

# Boxplots for identifying outliers
plt.figure(figsize=(10, 7))
sns.boxplot(data=data)
plt.title('Boxplot for Outlier Detection')
plt.show()

# Scatter plot with Plotly to explore specific relationships
fig = px.scatter(data, x='Exercise_Time_Min', y='BMI',
                 size='Healthy_Meals_Per_Day', color='Stress_Level',
                 title="Exercise Time vs BMI with Healthy Meals and Stress␣
 ↪Level")
fig.show()

# Scatter Matrix (interactive) using Plotly
fig = px.scatter_matrix(data, dimensions=data.columns,
                        color='Stress_Level', title="Scatter Matrix of Wellness␣
 ↪Data")
fig.show()

# Analyzing the distribution of Stress_Level
plt.figure(figsize=(10, 5))
sns.countplot(x='Stress_Level', data=data, palette='Set2')
plt.title('Distribution of Stress Level Scores')
plt.show()

# Analyzing the relationship between Sleep Hours and Stress Level
plt.figure(figsize=(10, 7))
sns.boxplot(x='Stress_Level', y='Sleep_Hours_Per_Night', data=data,␣
 ↪palette='Set3')
plt.title('Stress Level vs Sleep Hours Per Night')
plt.show()
```

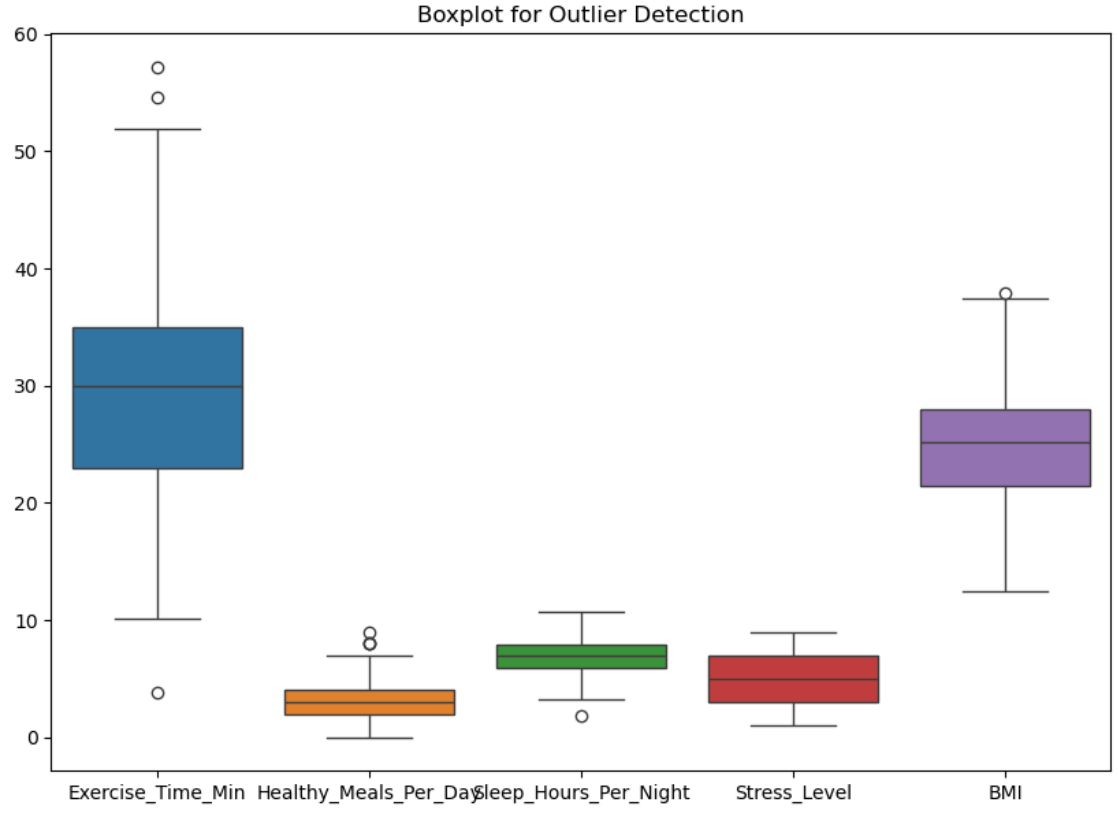
```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
```

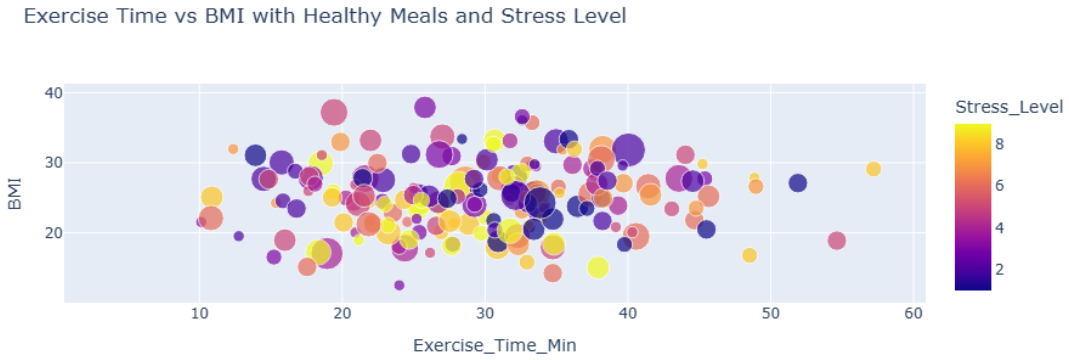
```
 0   Exercise_Time_Min      200 non-null    float64
 1   Healthy_Meals_Per_Day  200 non-null    int64
 2   Sleep_Hours_Per_Night  200 non-null    float64
 3   Stress_Level           200 non-null    int64
 4   BMI                    200 non-null    float64
dtypes: float64(3), int64(2)
memory usage: 7.9 KB
None

Descriptive Statistics:
       Exercise_Time_Min  Healthy_Meals_Per_Day  Sleep_Hours_Per_Night  \
count         200.000000             200.000000             200.000000
mean           29.592290               2.875000               6.933582
std             9.310039               1.815449               1.422471
min             3.802549               0.000000               1.778787
25%            22.948723               2.000000               5.967243
50%            29.958081               3.000000               6.972331
75%            35.008525               4.000000               7.886509
max            57.201692               9.000000              10.708419

       Stress_Level         BMI
count    200.000000  200.000000
mean       4.995000   25.150008
std        2.605556    5.070778
min        1.000000   12.502971
25%        3.000000   21.458196
50%        5.000000   25.155662
75%        7.000000   28.011155
max        9.000000   37.898547
```

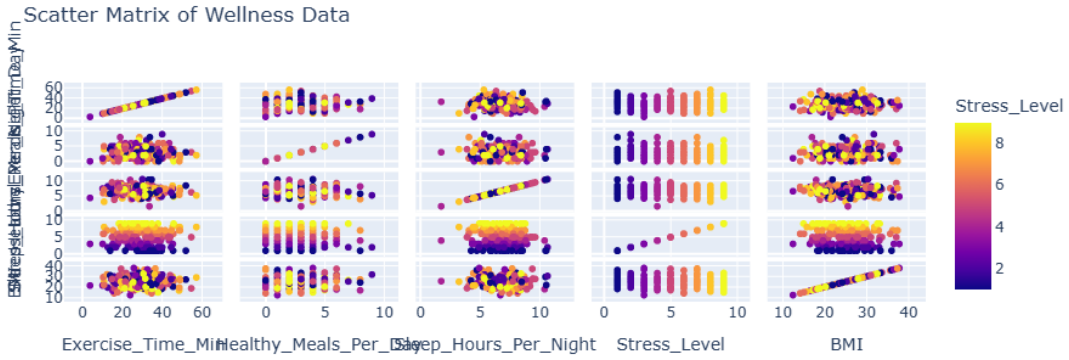

Correlation Matrix
Exercise_Time_Min
Healthy_Meals_Per_Day
Sleep_Hours_Per_Night
Stress_Level
BMI
1
0.012
-0.073
0.0015
0.069
0.012
1
-0.046
-0.081
0.025
-0.073
-0.046
1
-0.073
-0.058
0.0015
-0.081
-0.073
1
-0.13
0.069
0.025
-0.058
-0.13
1
1.0
0.8
0.6
0.4
0.2
0.0


Pairplot of Health and Wellness Features
Exercise_Time_Min
Healthy_Meals_Per_Day
Sleep_Hours_Per_Night
Stress_Level
BMI


Distribution of Exercise_Time_Min
Exercise_Time_Min
Distribution of Healthy_Meals_Per_Day
Healthy_Meals_Per_Day
Distribution of Sleep_Hours_Per_Night
Sleep_Hours_Per_Night
Distribution of Stress_Level
Stress_Level
Distribution of BMI
BMI
Count

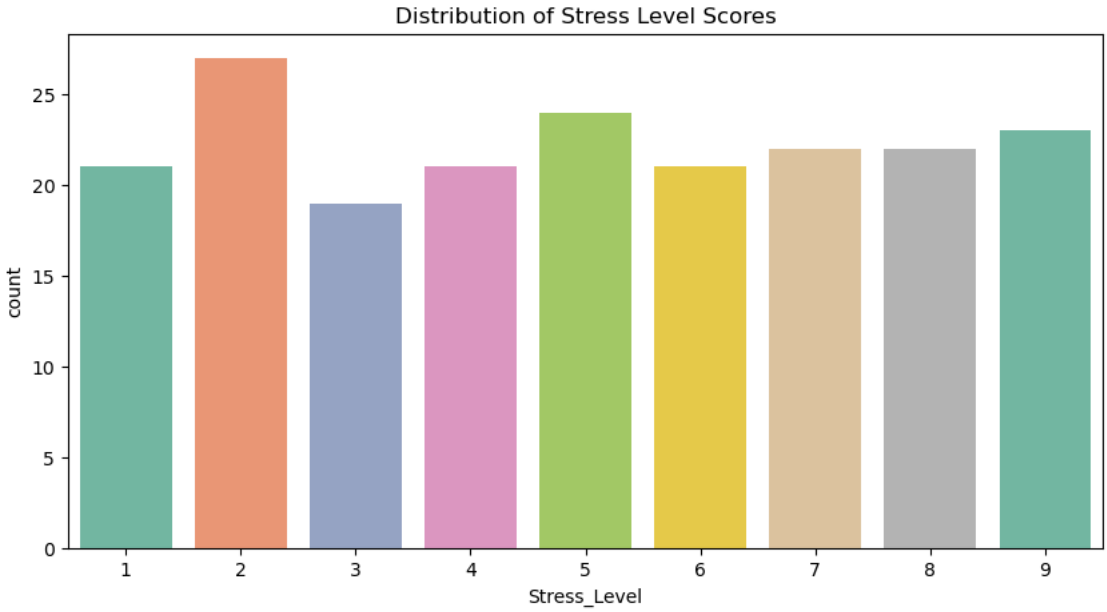

Boxplot for Outlier Detection
Exercise_Time_Min
Healthy_Meals_Per_Day
Sleep_Hours_Per_Night
Stress_Level
BMI



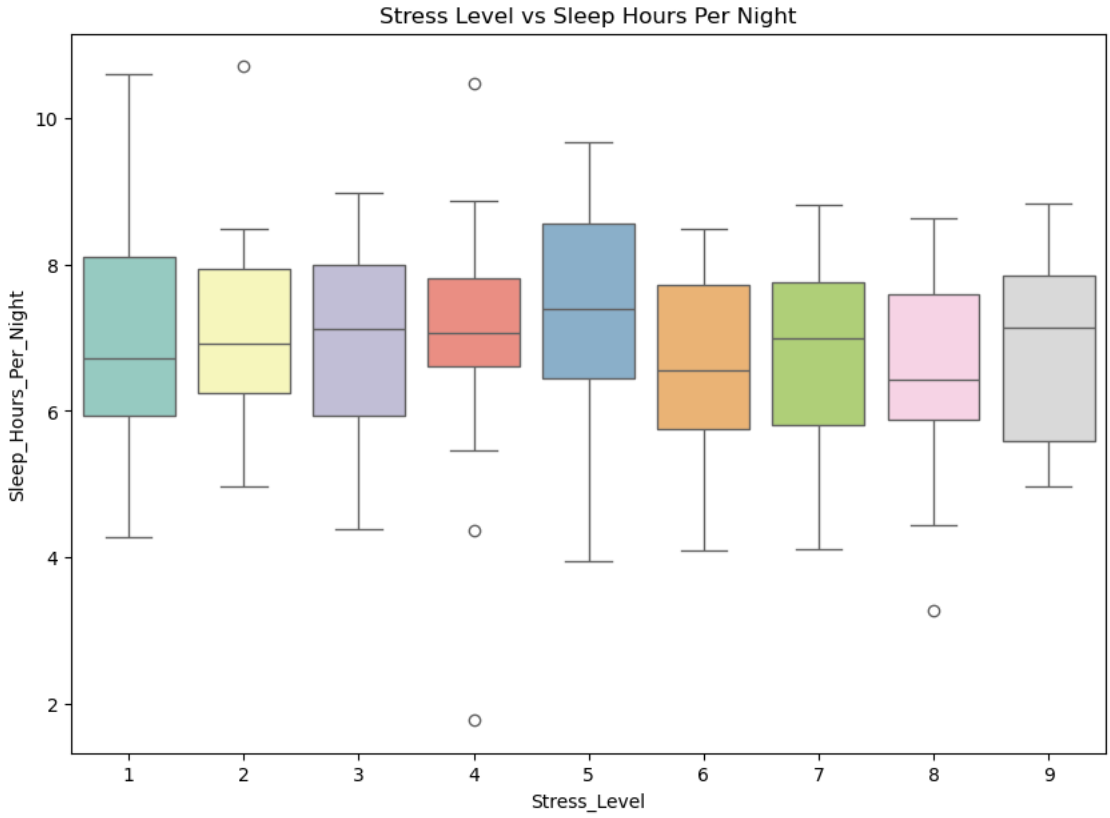


C:\Users\Neha Thakur\AppData\Local\Temp\ipykernel_12392\1639016864.py:57:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.



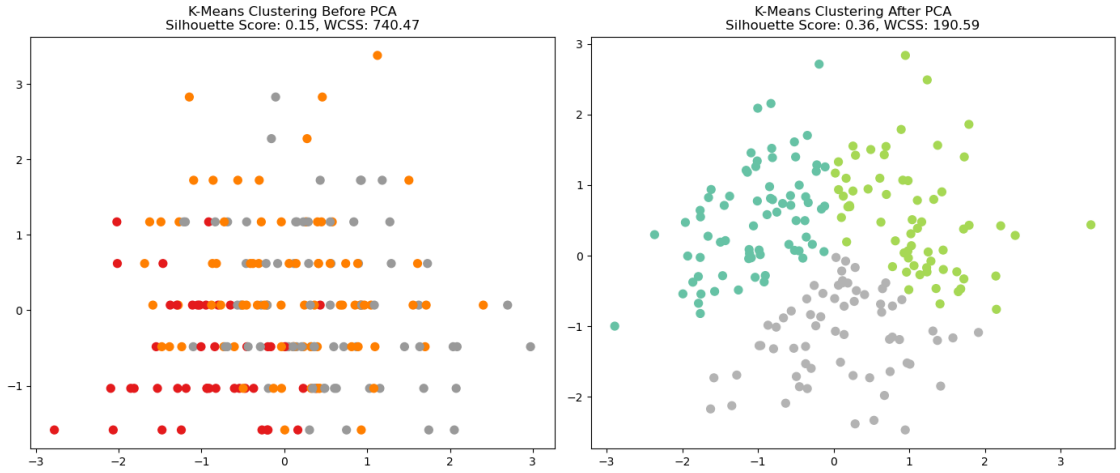

```
C:\Users\Neha Thakur\AppData\Local\Temp\ipykernel_12392\1639016864.py:63:
FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```



[11]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

# Load the dataset
file_path = 'C:/Users/Neha Thakur/Downloads/simulated_health_wellness_data.csv'
data = pd.read_csv(file_path)

# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)

# K-Means Clustering before PCA
kmeans_before_pca = KMeans(n_clusters=3, random_state=42)
data['KMeans_Cluster_Before_PCA'] = kmeans_before_pca.fit_predict(data_scaled)

# Evaluate KMeans before PCA
```

```
silhouette_before_pca = silhouette_score(data_scaled,␣
 ↪data['KMeans_Cluster_Before_PCA'])
wcss_before_pca = kmeans_before_pca.inertia_

print(f"Silhouette Score (Before PCA): {silhouette_before_pca}")
print(f"WCSS (Before PCA): {wcss_before_pca}")

# Apply PCA
pca = PCA(n_components=2)  # Reducing to 2 components
data_pca = pca.fit_transform(data_scaled)

# K-Means Clustering after PCA
kmeans_after_pca = KMeans(n_clusters=3, random_state=42)
data['KMeans_Cluster_After_PCA'] = kmeans_after_pca.fit_predict(data_pca)

# Evaluate KMeans after PCA
silhouette_after_pca = silhouette_score(data_pca,␣
 ↪data['KMeans_Cluster_After_PCA'])
wcss_after_pca = kmeans_after_pca.inertia_

print(f"Silhouette Score (After PCA): {silhouette_after_pca}")
print(f"WCSS (After PCA): {wcss_after_pca}")

# Plot the clusters before and after PCA
plt.figure(figsize=(14, 6))

# Before PCA
plt.subplot(1, 2, 1)
plt.scatter(data_scaled[:, 0], data_scaled[:, 1],␣
 ↪c=data['KMeans_Cluster_Before_PCA'], cmap='Set1', s=50)
plt.title(f"K-Means Clustering Before PCA\nSilhouette Score:␣
 ↪{silhouette_before_pca:.2f}, WCSS: {wcss_before_pca:.2f}")

# After PCA
plt.subplot(1, 2, 2)
plt.scatter(data_pca[:, 0], data_pca[:, 1], c=data['KMeans_Cluster_After_PCA'],␣
 ↪cmap='Set2', s=50)
plt.title(f"K-Means Clustering After PCA\nSilhouette Score:␣
 ↪{silhouette_after_pca:.2f}, WCSS: {wcss_after_pca:.2f}")

plt.tight_layout()
plt.show()

# Explained variance ratio of PCA components
explained_variance = pca.explained_variance_ratio_
print(f"Explained Variance by PCA components: {explained_variance}")
```

```
C:\Users\Neha Thakur\anaconda3\Lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.

C:\Users\Neha Thakur\anaconda3\Lib\site-
packages\sklearn\cluster\_kmeans.py:1446: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less
chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.


Silhouette Score (Before PCA): 0.1516159911787657
WCSS (Before PCA): 740.46626764846
Silhouette Score (After PCA): 0.3625606718282869
WCSS (After PCA): 190.5881092579563
```




```
Explained Variance by PCA components: [0.23691549 0.22082517]
```

[ ]: