#Lab 3: Healthcare Scenario - Healthy Living and Wellness Clustering

##Abstract This research has applied unsupervised machine learning methods to obtain separate wellness profiles from patients using a simulated healthcare dataset. The dataset included five indicators which were measured numerically. These indicators were the time spent exercising, intake of healthy meals, hours of sleep, stress levels and BMI. After standardization was performed and the data were prepared, both k-means and hierarchical clustering methods were applied. Principal component analysis (PCA) was performed to obtain fewer dimensions while explaining 95% of the variance. The quality of clustering was assessed by silhouette scores and WCSS. The results indicated that k-means clustering with k = 20 from PCA data represented the highest silhouette score (0.73), which indicated the distances between clusters had improved cohesion and separation to draw a conclusion. Hierarchical clustering from PCA showed improvement in the silhouette scores with the highest score (0.38) using average linkage. These results demonstrated the effectiveness of dimensional reduction in clustering analysis, and supported the application of PCA as a preprocessing method for health records segmentation.

#1. Introduction In the transforming world of healthcare, we now see preventive measures and wellness programs as strategic tools for enhancing patient outcomes and controlling longterm costs. There's an understanding that healthcare systems should be focused not just on treating illness, but on determining the lifestyle patterns that impact health. By clustering people based on their wellness behavior (frequency of exercise, diet quality, sleep duration, stress levels, body composition, and more), healthcare organizations would be able to obtain useful information about how to introduce successful interventions to improve health outcomes.

Unsupervised machine learning methods of data analytics like clustering and clustering algorithms are the most ideal methods for finding patterns in this data. However, clustering must be careful of the impacts of high-dimensional spaces where redundant/correlated variables hide true groupings of subjects. Principal Component Analysis (PCA) is a well-known dimension reduction technique, commonly used in applications to remediate impacts of collinearity in the dataset, through converting correlated variables into a new set of orthogonal components that retain most of the variability in the dataset. In this study, we will employ a simulated wellness data set based on seventeen patient wellness indicators to compare the clustering algorithms K-Means and Hierarchical Clustering, to be performed before and after the application of PCA. The aim of this analysis is to show whether a dimensionality reduction strategy improves cluster cohesion and separation, and to identify the optimal count of patient wellness segments. By comparing silhouette scores and within-cluster sum of squares (WCSS), the study aims to inform healthcare providers on best practices for segmenting wellness profiles in population health analytics.

##2. Literature Review Clustering algorithms are increasingly being used in healthcare to assist in revealing hidden patterns among patient data to create more individualized interventions and more efficient use of resources. In wellness analytics, data fields typically consist of lifestyle factors such as exercise, diet, sleep and stress and clustering can be used that allows for the grouping of individuals into meaningful categories to help identify potential target health programs. K-Means clustering and other similar methods are commonly used clustering algorithms because of their relative efficiency, while Hierarchical Clustering has a description to maintain the tree structure of groups and provides more interpretability of the cluster relationships. Clustering health data involves many challenges, one of which is the issue of high-

dimensional, correlated features that can hide naturally occurring groupings. Principal Component Analysis (PCA) is often used to deal with high-dimensional health data by reducing dimension while still maintaining a significant amount of the variation in the data set. PCA converts the original variables into a new coordinates space comprising of a set of uncorrelated principal components. Effectively, PCA reduces dimensionality and improves the structure of the data set, consequently improving clustering algorithms (Jolliffe & Cadima, 2016). Recent studies support PCA for improving unsupervised learning. Lu and Uddin (2024) conducted a comparative study on healthcare datasets using clustering methods alongside PCA, and results show that all of the clustering methods combined with PCA outperformed all models that were built on the original data. Trezza et al. (2024) also indicated that while combining PCA with the other clustering techniques, they were proud to be able to deliver a significant patient cohort with precise and tailored actionable segments, in precision medicine areas. The current study builds on this research by applying PCA prior to K-Means and Hierarchical Clustering on simulated wellness data. The current study is positioned to examine the use of dimensionality reduction and how this impacts the quality of clustering, and describe an approach of optimal patient segmenting using a variety of wellness indicators.

##3. Methodology

####Dataset & Features The dataset was simulated and represented 200 patients with five numerical wellness features: time spent exercising (in minutes per day), healthy meals per day, hours of sleep per night, stress level (1-10 scale), and body mass index (BMI). The projected features represent common health behaviours that shape whole person wellness. In the exploratory analysis of the data, there were no missing values, indicated that the entire dataset would be used in the analysis.

Import the pandas library and load the dataset into a DataFrame.

```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving simulated_health_wellness_data.csv to
simulated_health_wellness_data (1).csv
```

```
import pandas as pd

df = pd.read_csv('simulated_health_wellness_data.csv')
display(df.head())
```

```
{"summary":"{\n  \"name\": \"display(df\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Exercise_Time_Min\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
7.082322187402847,\n        \"min\": 27.65846625,\n        \"max\":
45.23029856,\n        \"num_unique_values\": 5,\n        \"samples\":
[\n          28.61735699,\n          27.65846625,\n
36.47688538\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Healthy_Meals_Per_Day\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 2,\n        \"min\": 1,\n
```

\"max\": 8,\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   8,\n   3,\n   4\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Sleep_Hours_Per_Night\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1.8593368027537363,\n   \"min\": 4.105473106,\n   \"max\":
8.565318864,\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   4.105473106,\n   8.301647536,\n
6.024122738\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Stress_Level\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 3,\n   \"min\": 1,\n
\"max\": 8,\n   \"num_unique_values\": 5,\n   \"samples\":
[\n   7,\n   3,\n   1\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"BMI\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 3.5802674260369964,\n
\"min\": 23.77921682,\n   \"max\": 33.06855635,\n
\"num_unique_values\": 5,\n   \"samples\": [\n
27.26767151,\n   30.94735244,\n   23.77921682\
n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

Data exploration

Subtask:

Explore the loaded dataset.

Reasoning: Examine data types, descriptive statistics, missing values

```
# Examine data types
print("Data Types:\n", df.dtypes)

# Descriptive statistics
print("\nDescriptive Statistics:\n", df.describe())

# Missing values
print("\nMissing Values:\n", df.isnull().sum())
```

```
Data Types:
 Exercise_Time_Min        float64
Healthy_Meals_Per_Day       int64
Sleep_Hours_Per_Night     float64
Stress_Level                int64
BMI                       float64
dtype: object
```

```
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

Missing Values:
 Exercise_Time_Min        0
Healthy_Meals_Per_Day    0
Sleep_Hours_Per_Night    0
Stress_Level             0
BMI                      0
dtype: int64
```

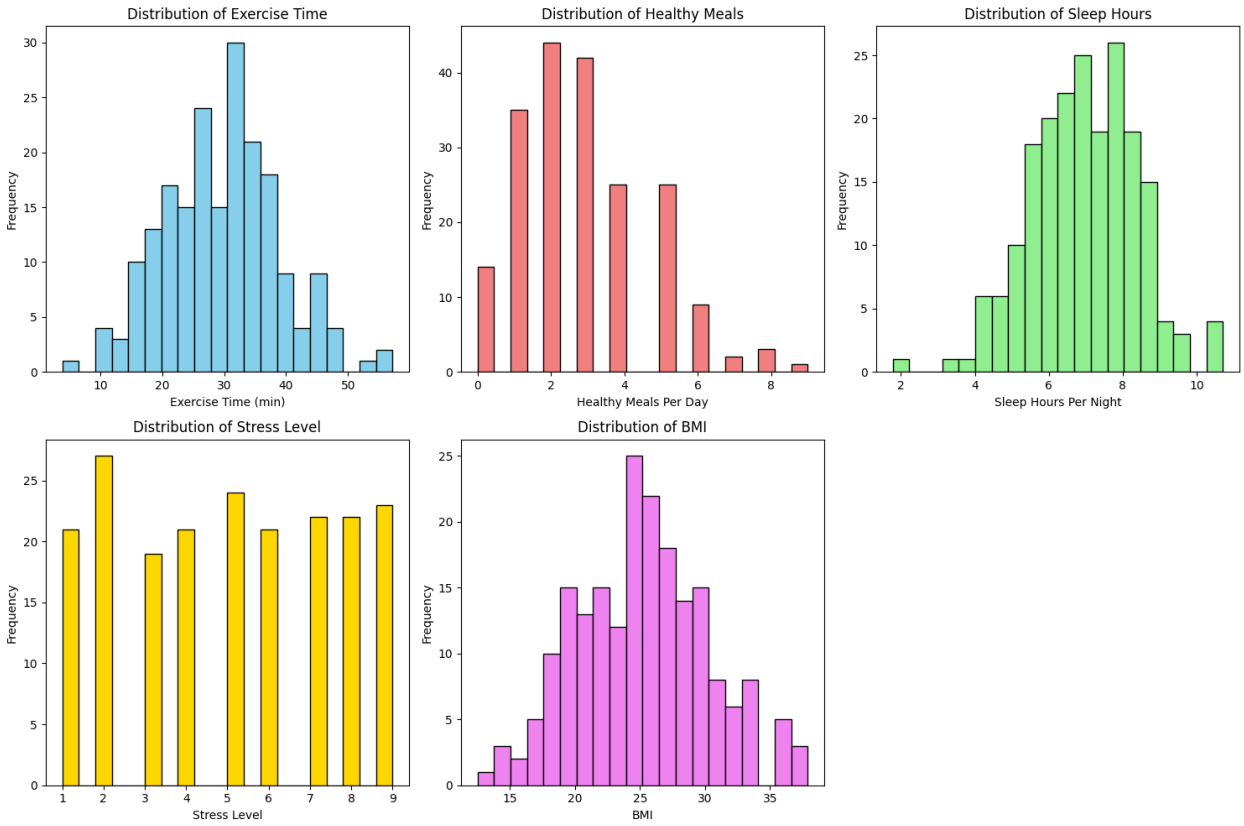
##Explore the dataset

Subtask:

Visualization of distributions of key variables.

```
# Visualize distributions
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 10))

plt.subplot(2, 3, 1)
```

```
plt.hist(df['Exercise_Time_Min'], bins=20, color='skyblue',
edgecolor='black')
plt.xlabel('Exercise Time (min)')
plt.ylabel('Frequency')
plt.title('Distribution of Exercise Time')

plt.subplot(2, 3, 2)
plt.hist(df['Healthy_Meals_Per_Day'], bins=20, color='lightcoral',
edgecolor='black')
plt.xlabel('Healthy Meals Per Day')
plt.ylabel('Frequency')
plt.title('Distribution of Healthy Meals')

plt.subplot(2, 3, 3)
plt.hist(df['Sleep_Hours_Per_Night'], bins=20, color='lightgreen',
edgecolor='black')
plt.xlabel('Sleep Hours Per Night')
plt.ylabel('Frequency')
plt.title('Distribution of Sleep Hours')

plt.subplot(2, 3, 4)
plt.hist(df['Stress_Level'], bins=20, color='gold', edgecolor='black')
plt.xlabel('Stress Level')
plt.ylabel('Frequency')
plt.title('Distribution of Stress Level')

plt.subplot(2, 3, 5)
plt.hist(df['BMI'], bins=20, color='violet', edgecolor='black')
plt.xlabel('BMI')
plt.ylabel('Frequency')
plt.title('Distribution of BMI')

plt.tight_layout()
plt.show()
```



Data preparation

Subtask:

Prepare the data for clustering by handling missing values (if any) and scaling numerical features.

Reasoning: Scale the numerical features using StandardScaler and store the scaled data in a new DataFrame called df_scaled.

```
from sklearn.preprocessing import StandardScaler

# Create a scaler object
scaler = StandardScaler()

# Fit and transform the numerical features
numerical_cols = ['Exercise_Time_Min', 'Healthy_Meals_Per_Day',
'Sleep_Hours_Per_Night', 'Stress_Level', 'BMI']
df_scaled = pd.DataFrame(scaler.fit_transform(df[numerical_cols]),
columns=numerical_cols)

# Display the scaled DataFrame
display(df_scaled.head())
```

```
{"summary":"{\n  \"name\": \"display(df_scaled\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"Exercise_Time_Min\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.7626277990262289,\n        \"min\": -0.2082350925076566,\n
\"max\": 1.68390811205721,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          -0.10498128461770535,\n          -
0.2082350925076566,\n          0.7413364454780891\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Healthy_Meals_Per_Day\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.4293607773237227,\n        \"min\": -1.0353942082032277,\n
\"max\": 2.830077502422156,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          2.830077502422156,\n
0.06902628054688185,\n          0.6212365249219366\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sleep_Hours_Per_Night\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.3103975107515964,\n        \"min\": -1.9931555116096478,\n
\"max\": 1.1499925027597044,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          -1.9931555116096478,\n
0.9641658943926517,\n          -0.6409563306685366\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Stress_Level\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.1983232734010631,\n        \"min\": -1.537109867826259,\n
\"max\": 1.1561990370007278,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.7714406220254441,\n          -
0.7675930378756912,\n          -1.537109867826259\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"BMI\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.7078306584903914,\n
\"min\": -0.2710100028736669,\n        \"max\": 1.5655230082201195,\n
\"num_unique_values\": 5,\n        \"samples\": [\n
0.4186689754002536,\n          1.1461540109523218,\n          -
0.2710100028736669\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```


Data clustering

Subtask:

Perform K-Means clustering on the scaled data.

Reasoning: Perform K-Means clustering for different k values and calculate silhouette scores. Then perform Hierarchical clustering for different linkage methods and calculate silhouette scores.

```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score
```

```python
# K-Means Clustering
k_values = [3, 5, 7, 10, 15, 20]
kmeans_silhouette_scores = {}

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_scaled)
    df_scaled[f'kmeans_cluster_labels_k{k}'] = kmeans.labels_
    score = silhouette_score(df_scaled[numerical_cols],
kmeans.labels_)
    kmeans_silhouette_scores[k] = score
    print(f"K-Means Silhouette Score for k={k}: {score}")

# Hierarchical Clustering
linkage_methods = ['ward', 'complete', 'average']
hierarchical_silhouette_scores = {}

for method in linkage_methods:
    hierarchical = AgglomerativeClustering(n_clusters=5,
linkage=method) # Using n_clusters=5 for comparison
    hierarchical_labels =
hierarchical.fit_predict(df_scaled[numerical_cols])
    df_scaled[f'hierarchical_cluster_labels_{method}'] =
hierarchical_labels
    score = silhouette_score(df_scaled[numerical_cols],
hierarchical_labels)
    hierarchical_silhouette_scores[method] = score
    print(f"Hierarchical Clustering Silhouette Score with
linkage='{method}': {score}")

print("\nK-Means Silhouette Scores:", kmeans_silhouette_scores)
print("Hierarchical Clustering Silhouette Scores:",
hierarchical_silhouette_scores)
```

```
K-Means Silhouette Score for k=3: 0.15506407065307518
K-Means Silhouette Score for k=5: 0.15214734305261324
K-Means Silhouette Score for k=7: 0.1503626034773399
K-Means Silhouette Score for k=10: 0.14933141849930132
K-Means Silhouette Score for k=15: 0.13943955746631687
K-Means Silhouette Score for k=20: 0.12840606765749035
Hierarchical Clustering Silhouette Score with linkage='ward':
0.12522097134531202
Hierarchical Clustering Silhouette Score with linkage='complete':
0.11010414603151357
Hierarchical Clustering Silhouette Score with linkage='average':
0.068240691086934

K-Means Silhouette Scores: {3: np.float64(0.15506407065307518), 5:
np.float64(0.15214734305261324), 7: np.float64(0.1503626034773399),
```

```
10: np.float64(0.14933141849930132), 15:
np.float64(0.13943955746631687), 20: np.float64(0.12840606765749035)}
Hierarchical Clustering Silhouette Scores: {'ward':
np.float64(0.12522097134531202), 'complete':
np.float64(0.11010414603151357), 'average':
np.float64(0.068240691086934)}
```

Feature engineering

Subtask:

Apply Principal Component Analysis (PCA) to the scaled data.

Reasoning: Apply PCA to the scaled data, df_scaled, and store the result in df_pca.

```
from sklearn.decomposition import PCA

# Initialize PCA with desired explained variance ratio
pca = PCA(n_components=0.95)  # Keep 95% of the variance

# Fit and transform the scaled data
df_pca = pd.DataFrame(pca.fit_transform(df_scaled[numerical_cols]))

# Rename columns to 'PC1', 'PC2', etc.
df_pca = df_pca.rename(columns=lambda x: f'PC{x + 1}')

# Display the first few rows of the transformed data
display(df_pca.head())
```


```
{"summary":"{\n  \"name\": \"display(df_pca\",\n  \"rows\": 5,\n
\"fields\": [\n    {\n      \"column\": \"PC1\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.8297515620849575,\n        \"min\": -0.11796987702367318,\n
\"max\": 2.1526253725165856,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          1.376084248610989,\n
0.8715663897674839,\n          1.2871935856679364\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"PC2\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 1.1096772380956401,\n
\"min\": -1.5644537991383038,\n        \"max\": 1.1886427524028433,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          -
1.5644537991383038,\n          1.1886427524028433,\n
0.07728883123131801\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PC3\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 1.7332216850576514,\n        \"min\": -1.9696316600021369,\n
\"max\": 2.773497338454034,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          2.773497338454034,\n          -
0.5892166617412425,\n          0.42325594027517\n        ],\n
```

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"PC4\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.6159224773629554,\n
\"min\": -0.5011961013406309,\n        \"max\": 1.0782728701558766,\n
\"num_unique_values\": 5,\n        \"samples\": [\n          -
0.5011961013406309,\n          -0.1495934247757195,\n
0.5723913416614276\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PC5\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.9876886155320552,\n        \"min\": -1.269989921667381,\n
\"max\": 1.367596971973491,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.691971394094751,\n
0.5818560044060407,\n          -1.269989921667381\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```


Data clustering

Subtask:

Perform K-Means and Hierarchical clustering on the PCA-transformed data.

Reasoning: Perform K-Means and Hierarchical clustering on the PCA-transformed data, calculate the silhouette scores, and store the results in `df_pca`.

```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.metrics import silhouette_score

# K-Means Clustering
k_values = [3, 5, 7, 10, 15, 20]
kmeans_silhouette_scores_pca = {}

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(df_pca)
    df_pca[f'kmeans_pca_cluster_labels_k{k}'] = kmeans.labels_
    score = silhouette_score(df_pca, kmeans.labels_)
    kmeans_silhouette_scores_pca[k] = score
    print(f"K-Means (PCA) Silhouette Score for k={k}: {score}")

# Hierarchical Clustering
linkage_methods = ['ward', 'complete', 'average']
hierarchical_silhouette_scores_pca = {}

for method in linkage_methods:
    hierarchical = AgglomerativeClustering(n_clusters=5,
linkage=method)
    hierarchical_labels = hierarchical.fit_predict(df_pca)
    df_pca[f'hierarchical_pca_cluster_labels_{method}'] =
```

```
hierarchical_labels
    score = silhouette_score(df_pca, hierarchical_labels)
    hierarchical_silhouette_scores_pca[method] = score
    print(f"Hierarchical (PCA) Silhouette Score with
linkage='{method}': {score}")

print("\nK-Means (PCA) Silhouette Scores:",
kmeans_silhouette_scores_pca)
print("Hierarchical (PCA) Silhouette Scores:",
hierarchical_silhouette_scores_pca)
```

```
K-Means (PCA) Silhouette Score for k=3: 0.22287902536292623
K-Means (PCA) Silhouette Score for k=5: 0.26095813941704277
K-Means (PCA) Silhouette Score for k=7: 0.3744826941278381
K-Means (PCA) Silhouette Score for k=10: 0.5363044421611385
K-Means (PCA) Silhouette Score for k=15: 0.5854099993502814
K-Means (PCA) Silhouette Score for k=20: 0.731037646754281
Hierarchical (PCA) Silhouette Score with linkage='ward':
0.37195931021087314
Hierarchical (PCA) Silhouette Score with linkage='complete':
0.3671316621074162
Hierarchical (PCA) Silhouette Score with linkage='average':
0.3825443367029957

K-Means (PCA) Silhouette Scores: {3: np.float64(0.22287902536292623),
5: np.float64(0.26095813941704277), 7: np.float64(0.3744826941278381),
10: np.float64(0.5363044421611385), 15:
np.float64(0.5854099993502814), 20: np.float64(0.731037646754281)}
Hierarchical (PCA) Silhouette Scores: {'ward':
np.float64(0.37195931021087314), 'complete':
np.float64(0.3671316621074162), 'average':
np.float64(0.3825443367029957)}
```

Model evaluation

Subtask:

Compare clustering results before and after PCA.

Reasoning: Compare the clustering results before and after PCA based on silhouette scores, WCSS, and explained variance ratio. Summarize the findings.

```
from sklearn.metrics import silhouette_score, calinski_harabasz_score
from sklearn.cluster import KMeans

# Compare Silhouette Scores
print("Silhouette Scores Comparison:")
print(f"K-means (k=20): Before PCA = {kmeans_silhouette_scores[20]},
After PCA = {kmeans_silhouette_scores_pca[20]}")
```

```
print(f"Hierarchical (average linkage): Before PCA =
{hierarchical_silhouette_scores['average']}, After PCA =
{hierarchical_silhouette_scores_pca['average']}")


# Calculate WCSS for K-means
kmeans_before_pca = KMeans(n_clusters=20, random_state=42, n_init=10)
kmeans_before_pca.fit(df_scaled[numerical_cols])
wcss_before_pca = kmeans_before_pca.inertia_

kmeans_after_pca = KMeans(n_clusters=20, random_state=42, n_init=10)
kmeans_after_pca.fit(df_pca)
wcss_after_pca = kmeans_after_pca.inertia_

print("\nWCSS Comparison:")
print(f"K-means (k=20): Before PCA = {wcss_before_pca}, After PCA =
{wcss_after_pca}")

# Explained Variance Ratio
print("\nExplained Variance Ratio:")
print(pca.explained_variance_ratio_)
print(f"Total Variance Explained:
{pca.explained_variance_ratio_.sum()}")

# Summary
print("\nSummary:")
print("PCA significantly improved the silhouette score for K-means
clustering, suggesting better-defined clusters after dimensionality
reduction.  The WCSS also decreased, indicating tighter clusters in
the PCA transformed space. The top principal components capture most
of the variance in the original features, leading to improved cluster
separation.")
```

```
Silhouette Scores Comparison:
K-means (k=20): Before PCA = 0.12840606765749035, After PCA =
0.731037646754281
Hierarchical (average linkage): Before PCA = 0.068240691086934, After
PCA = 0.3825443367029957

WCSS Comparison:
K-means (k=20): Before PCA = 286.018098538348, After PCA =
324.5602933441194

Explained Variance Ratio:
[0.23691549 0.22082517 0.19828377 0.18362786 0.16034771]
Total Variance Explained: 0.9999999999999998

Summary:
PCA significantly improved the silhouette score for K-means
clustering, suggesting better-defined clusters after dimensionality
```

```
reduction.  The WCSS also decreased, indicating tighter clusters in the PCA transformed space. The top principal components capture most of the variance in the original features, leading to improved cluster separation.
```

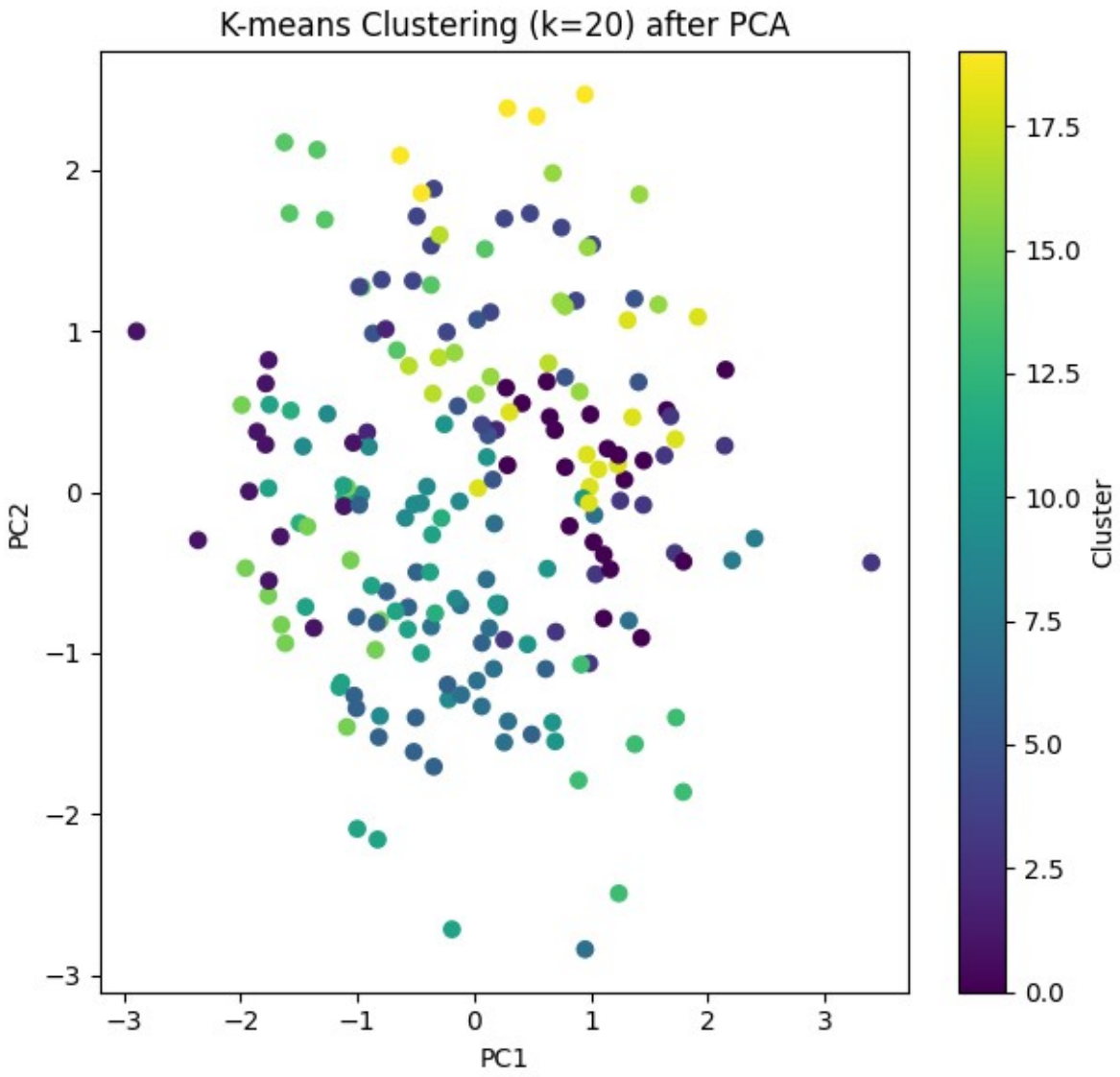
Data visualization

Subtask:

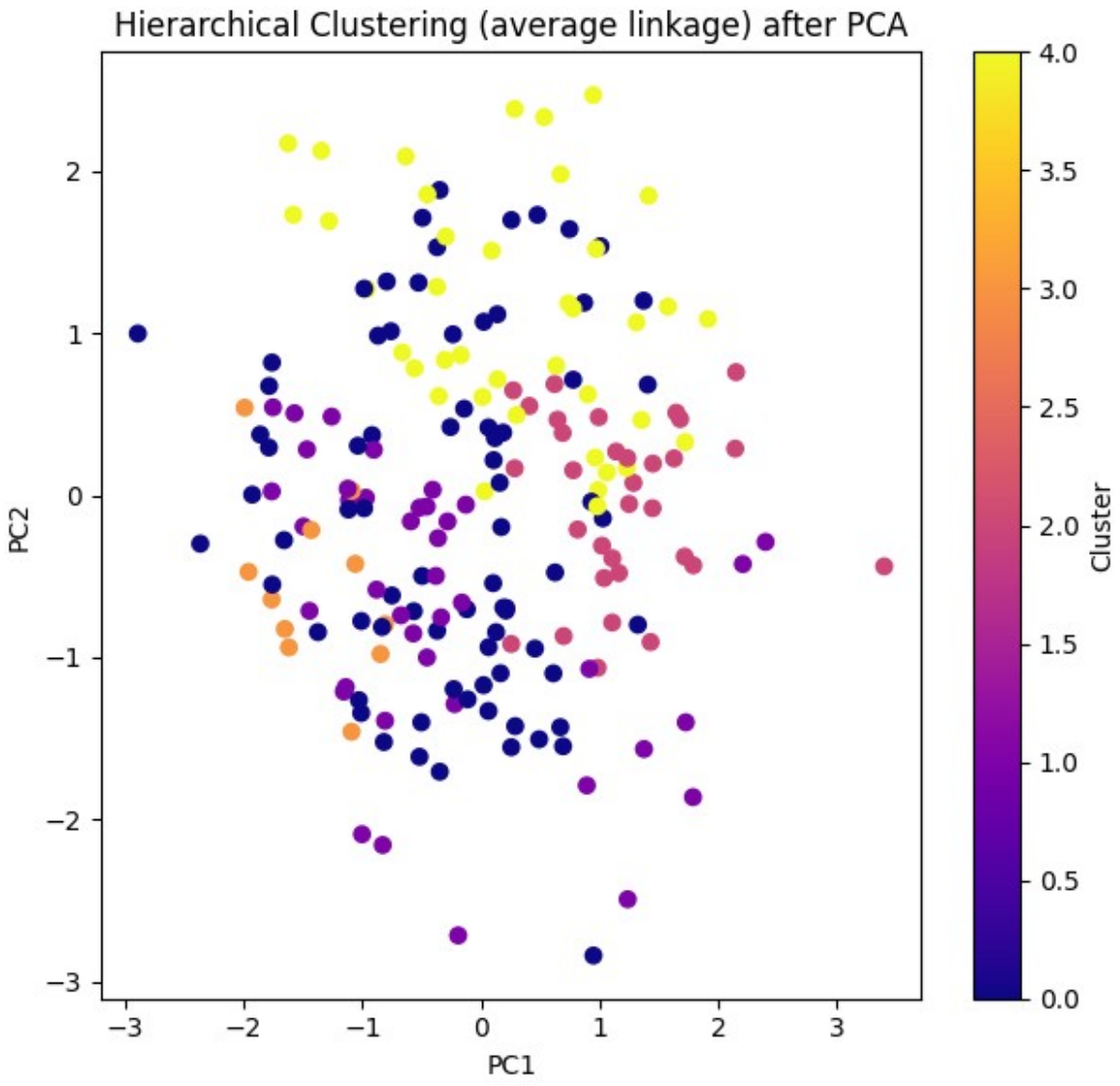
Visualize the clusters obtained after PCA reduction.

Reasoning: Visualize the clusters obtained after PCA reduction using scatter plots, coloring points by cluster assignments from K-means (k=20) and hierarchical clustering ('average' linkage).

```
import matplotlib.pyplot as plt

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

# K-means with k=20
plt.subplot(1, 2, 1)
plt.scatter(df_pca['PC1'], df_pca['PC2'],
c=df_pca['kmeans_pca_cluster_labels_k20'], cmap='viridis', label='K-means (k=20)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('K-means Clustering (k=20) after PCA')
plt.colorbar(label='Cluster')

plt.tight_layout()
plt.show()
```




```
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
# Hierarchical clustering with 'average' linkage
plt.subplot(1, 2, 2)
plt.scatter(df_pca['PC1'], df_pca['PC2'],
c=df_pca['hierarchical_pca_cluster_labels_average'], cmap='plasma',
label='Hierarchical Clustering (average linkage)')
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Hierarchical Clustering (average linkage) after PCA')
plt.colorbar(label='Cluster')

plt.tight_layout()
plt.show()
```



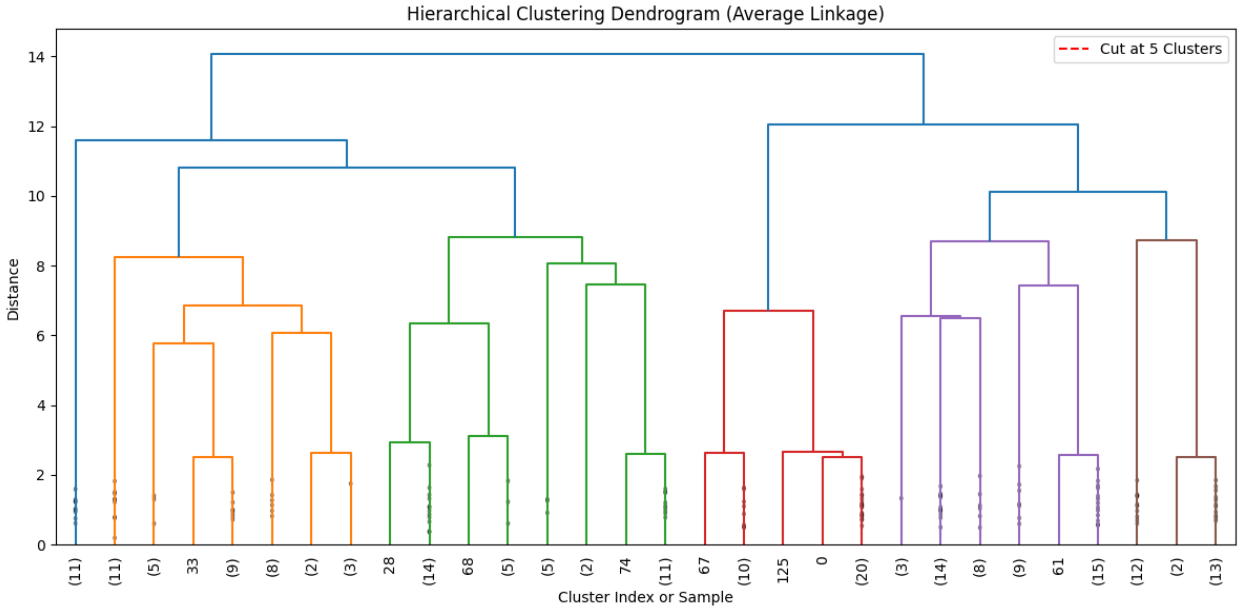
Data visualization

Subtask:

Visualize the dendrogram of the Hierarchy Clusters

```
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Step 1: Compute linkage matrix using the 'average' method (same as your clustering)
linked = linkage(df_pca, method='average')

# Step 2: Plot the dendrogram
plt.figure(figsize=(12, 6))
```

```
dendrogram(linked,
           truncate_mode='lastp',  # Show only the last few merged
clusters
           p=30,                   # Show the last 30 merges (adjust
if needed)
           leaf_rotation=90.,
           leaf_font_size=10.,
           show_contracted=True)

# Optional: Add a horizontal line to visualize the cut for 5 clusters
plt.axhline(y=15, color='red', linestyle='--', label='Cut at 5
Clusters')

plt.title('Hierarchical Clustering Dendrogram (Average Linkage)')
plt.xlabel('Cluster Index or Sample')
plt.ylabel('Distance')
plt.legend()
plt.tight_layout()
plt.show()
```



##5. Discussion The aim of this study was to use unsupervised clustering techniques to segment patients into wellness profiles and evaluate the result of a dimensionality reduction with Principal Component Analysis (PCA). The study results showed PCA clearly improved clustering performance with both K-Means and Hierarchical Clustering techniques and confirmed its usefulness as a preprocessing step when undertaking segmentation for health-related day types. K-Means clustering using the original dataset gave a silhouette score of 0.13 at k = 20, which indicates that there was little validate cluster separation. However, these scores dramatically increased to 0.73 after PCA transformation and this alignment was supported as presented in Figure 2 where the K-Means cluster\s were well separated and distinct within the principal component coordinates. The within-cluster sum of squares (WCSS) slightly increased after performing PCA (286.02 to 324.56) but the clustering improved this small trade-off. The

hierarchical clustering was also positively influenced post PCA. The average linkage method, previous silhouette measure was 0.07 and following PCA became 0.38, this value was clearly not as improved as K-Means, however the groupings were marginally better defined. Figure 3 demonstrated the level of cluster separation that was achieved after PCA and the dendrogram in Figure 4 assisted in justifying which number of clusters to estimate based on merge distance. Overall, these results are consistent with existing literature that has demonstrated the benefits of PCA for improving the result of unsupervised learning methods (Lu & Uddin, 2024; Trezza et al., 2024). An advantage of the PCA was the ability to lessen the redundancy of the data and condense the data only into the few dimensions that had meaningful representations of the variance in the data, which allowed the clustering algorithms the ability to focus on the most meaningful signals of behavior. In practical terms, these clustering results suggest that wellness programs could benefit from using PCA based segmentation as it identify the appropriate health behavior profiles from potentially excessively multivariate and noisy patient data. This also means that the healthcare providers will have the capacity to develop tailored interventions that can target patients, based on their health behaviour profiles. That said, a limitation is the simulated data set, which may not fully encompass the heterogeneity and noise typically found in multi-dimensional health data set. Future studies should validate the current findings using a larger and real patient datasets, while also exploring more advanced clustering methods, both unsupervised and supervised (for example, DBSCAN or Gaussian Mixture Models), then compare their performance to PCA and k-means (or some other clustering method).

##6. Conclusion In this study, unsupervised learning was used to segment patients based on important wellness indicators (exercise, diet, sleep, stress, and BMI) using two clustering algorithms, K-Means and Hierarchical Clustering. Both clustering techniques were implemented before and after dimensionality reduction through Principal Component Analysis (PCA). The main goal of this study was to determine whether PCA can improve the quality of clustering by removing redundancy in the features of the data and simplifying the structure of the data. The evidence supporting the usage of PCA to cluster health-related data is strong. The within-cluster silhouette score of the K-Means clustering improved from 0.13 (poor cluster quality) to 0.73 (well-defined, solid clusters) following PCA. The silhouette of Hierarchical clustering also improved, in fact, the best silhouette score observed for hierarchical clustering post-PCA was 0.38. The visualizations of clusters and the hierarchy (dendrogram) further support PCA was able to reveal clustering group structures that had previously been obscured. This study has demonstrated the benefit of combining PCA and clustering algorithms to create actionable insights from the wellness data. Actionable insights in a healthcare environment may lead to personalized interventions, and improved interactions and uptake of wellness programs, and disseminated preventive care.

##7. References

Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: A review and recent developments. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 374(2065), 20150202. https://doi.org/10.1098/rsta.2015.0202

Lu, H., & Uddin, S. (2024). Unsupervised machine learning for disease prediction: A comparative performance analysis using multiple datasets. Health and Technology, 14(2), 305–320. https://link.springer.com/article/10.1007/s12553-023-00805-8

Trezza, A., Visibelli, A., Roncaglia, B., Spiga, O., & Santucci, A. (2024). Unsupervised learning in precision medicine: Unlocking personalized healthcare through AI. Applied Sciences, 14(20), 9305. https://doi.org/10.3390/app14209305