Project Report

on

Personalized Health & Fitness Clustering

Name : Deep Bhanushali
Sap ID : 86092400012
Roll no: A012

Name : Fardeen Khatri
Sap ID: 86092400018
Roll no: A018

Project Overview

This project focuses on segmenting individuals based on their fitness and health patterns using **unsupervised machine learning**, particularly **K-Means clustering**. The insights derived from clustering are used to generate **personalized wellness recommendations** for different groups. The dataset used includes various physical, physiological, and lifestyle parameters, making it suitable for building a robust health profiling system.

Objectives

- Identify natural groupings in fitness data using clustering algorithms.
- Visualize patterns in user profiles using PCA and t-SNE.
- Generate personalized fitness recommendations for each user cluster.
- Lay the groundwork for intelligent health coaching or fitness tracking applications.

Dataset Description

- **File Name**: fitlife_health_fitness.csv
- **Attributes Used**: age, height_cm, weight_kg, bmi, duration_minutes, intensity, calories_burned, daily_steps, avg_heart_rate, resting_heart_rate, blood_pressure_systolic, blood_pressure_diastolic, hours_sleep, stress_level, hydration_level, fitness_level
- **Dropped Columns**:
  participant_id, gender, activity_type, health_condition, smoking_status

```
   participant_id        date  age gender  height_cm  weight_kg  \
0               1  2024-01-01   56      F      165.3       53.7
1               1  2024-01-04   56      F      165.3       53.9
2               1  2024-01-05   56      F      165.3       54.2
3               1  2024-01-07   56      F      165.3       54.4
4               1  2024-01-09   56      F      165.3       54.7

     activity_type  duration_minutes intensity  calories_burned  ...  \
0          Dancing                41       Low              3.3  ...
1         Swimming                28       Low              2.9  ...
2         Swimming                21    Medium              2.6  ...
3  Weight Training                99    Medium             10.7  ...
4         Swimming               100    Medium             12.7  ...

   stress_level  daily_steps  hydration_level   bmi  resting_heart_rate  \
0             3         7128              1.5  19.6                69.5
1             7         7925              1.8  19.6                69.5
2             7         7557              2.7  19.6                69.5
3             8        11120              2.6  19.6                69.5
4             1         5406              1.5  19.6                69.5

   blood_pressure_systolic  blood_pressure_diastolic  health_condition  \
0                    110.7                      72.9               NaN
1                    110.7                      72.9               NaN
2                    110.7                      72.9               NaN
3                    110.7                      72.9               NaN
4                    110.7                      72.9               NaN

   smoking_status fitness_level
0           Never          0.04
1           Never          0.07
2           Never          0.09
3           Never          0.21
4           Never          0.33
```

- **Tools & Libraries Used**
- **Languages & Platforms**: Python, Jupyter Notebook / Google Colab
- **Libraries**:
    - pandas, numpy – Data processing
    - matplotlib, seaborn – Visualization
    - scikit-learn – Clustering (KMeans), PCA, t-SNE
    - yellowbrick – KElbowVisualizer for optimal k value

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans, DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from yellowbrick.cluster import KElbowVisualizer
```

Methodology

Step 1: Data Preprocessing

- Removed non-informative columns and duplicates.

```
drop_cols = ['participant_id', 'gender', 'activity_type', 'health_condition', 'smoking_status']
df.drop(columns=drop_cols, inplace=True, errors='ignore')
```

```
df = df.drop_duplicates()
```

```
df = df.dropna()
```

- Encoded the categorical intensity feature using Label Encoding.

```
# Encode categorical variables (intensity)
if 'intensity' in df.columns:
    label_encoder = LabelEncoder()
    df['intensity'] = label_encoder.fit_transform(df['intensity'])
```

```
# Selecting Features for Clustering
features = ['age', 'height_cm', 'weight_kg', 'bmi', 'duration_minutes', 'intensity', 'calories_burned',
            'daily_steps', 'avg_heart_rate', 'resting_heart_rate', 'blood_pressure_systolic',
            'blood_pressure_diastolic', 'hours_sleep', 'stress_level', 'hydration_level', 'fitness_level']

cluster_data = df[features]
```

- Handled missing values and normalized all features using StandardScaler.

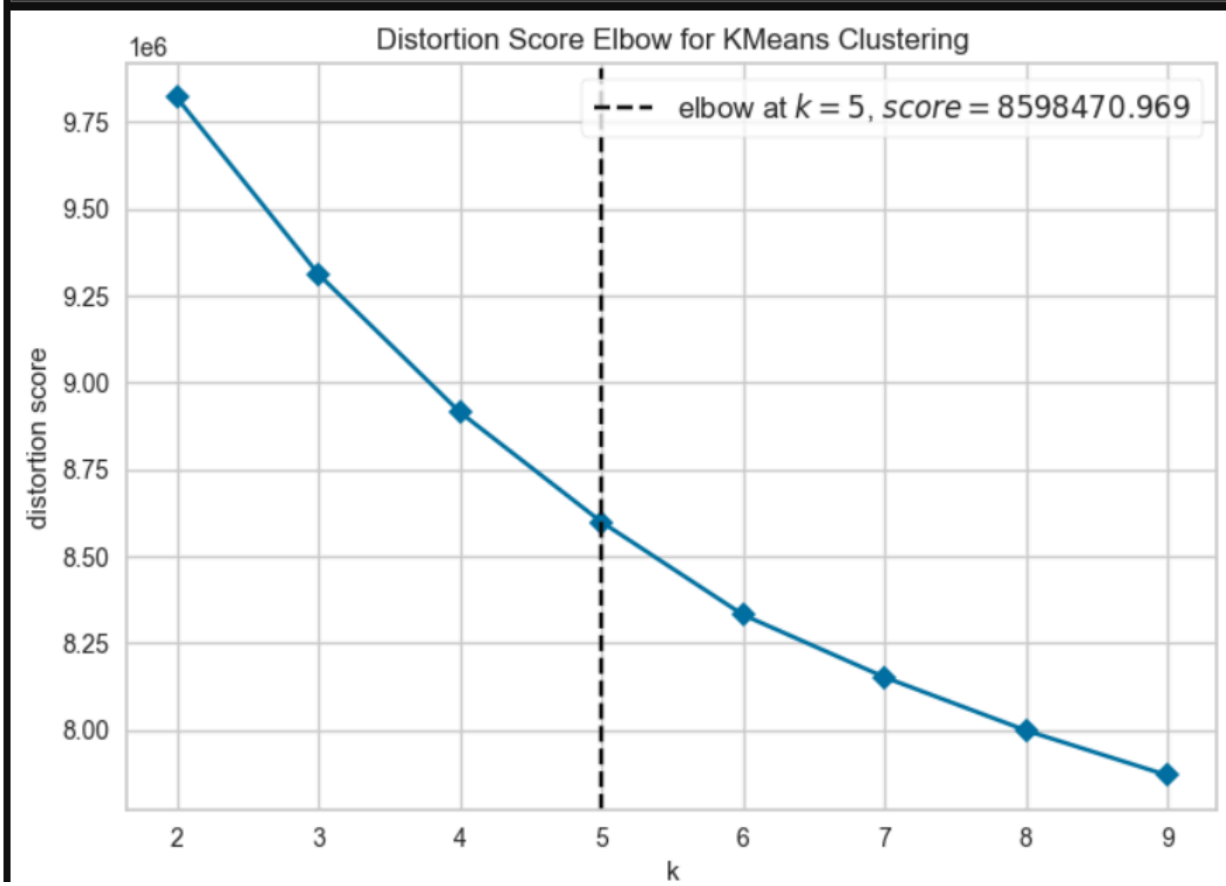
```
# Normalize Data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(cluster_data)

scaled_df = pd.DataFrame(scaled_data, columns=features)

print(scaled_df.head())
```

```
        age  height_cm  weight_kg       bmi  duration_minutes  intensity  \
0  1.055931  -0.359673  -1.835205 -0.878027         -0.995349  -0.383885
1  1.055931  -0.359673  -1.826301 -0.878027         -1.441361  -0.383885
2  1.055931  -0.359673  -1.812945 -0.878027         -1.681521   0.897187
3  1.055931  -0.359673  -1.804041 -0.878027          0.994548   0.897187
4  1.055931  -0.359673  -1.790685 -0.878027          1.028857   0.897187

   calories_burned  daily_steps  avg_heart_rate  resting_heart_rate  \
0        -1.209879    -0.730195       -1.597227           -0.100609
1        -1.249937    -0.342314       -1.653360           -0.100609
2        -1.279980    -0.521411       -0.306160           -0.100609
3        -0.468808     1.212616        0.535839           -0.100609
4        -0.268518    -1.568251       -1.092027           -0.100609

   blood_pressure_systolic  blood_pressure_diastolic  hours_sleep  \
0                -0.929298                 -0.884539    -0.461695
1                -0.929298                 -0.884539     1.081408
2                -0.929298                 -0.884539    -0.873189
3                -0.929298                 -0.884539     0.155546
4                -0.929298                 -0.884539     0.052672

   stress_level  hydration_level  fitness_level
0     -0.813129        -1.725978      -1.723750
1      0.630900        -1.207888      -1.718298
2      0.630900         0.346383      -1.714663
3      0.991908         0.173686      -1.692855
4     -1.535144        -1.725978      -1.671046
```

Step 2: Finding Optimal Clusters

- Used the **Elbow Method** via Yellowbrick to determine the optimal number of clusters (k).
- Final choice: k = 5

```
# STEP 4: Finding Optimal Clusters for K-Means #
###########################################
kmeans = KMeans(random_state=42, n_init=10)
visualizer = KElbowVisualizer(kmeans, k=(2,10), metric='distortion', timings=False)
visualizer.fit(scaled_df)
visualizer.show()
```



Step 3: K-Means Clustering

- Applied K-Means clustering and calculated **Silhouette Score** to assess the clustering quality.

```
# STEP 5: Apply K-Means Clustering #
#######################################
optimal_k = 5  # Adjust based on Elbow Method result
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init=10)
scaled_df['Cluster_KMeans'] = kmeans.fit_predict(scaled_df)

# Silhouette Score for K-Means
silhouette_kmeans = silhouette_score(scaled_df.iloc[:, :-1], scaled_df['Cluster_KMeans'])
print(f"K-Means Silhouette Score: {silhouette_kmeans}")
```

Step 4: Dimensionality Reduction

- Applied **PCA (2D)** for visualization of clustering boundaries.

```
# STEP 7: Visualizing Clusters with PCA #
################################################
pca = PCA(n_components=2)
df_pca = pca.fit_transform(scaled_df.iloc[:, :-2])
scaled_df['PCA1'] = df_pca[:, 0]
scaled_df['PCA2'] = df_pca[:, 1]

# Plot K-Means Clusters
plt.figure(figsize=(10,5))
sns.scatterplot(data=scaled_df, x="PCA1", y="PCA2", hue="Cluster_KMeans", palette="viridis")
plt.title("K-Means Clustering Results (PCA Reduced)")
plt.show()
```

- Used **t-SNE** for deeper 2D and 3D exploration of data clusters.

```
# t-SNE 2D
tsne_2d = TSNE(n_components=2, random_state=42, perplexity=30)
tsne_result_2d = tsne_2d.fit_transform(scaled_df.iloc[:, :-3])
scaled_df['TSNE1'] = tsne_result_2d[:, 0]
scaled_df['TSNE2'] = tsne_result_2d[:, 1]

plt.figure(figsize=(10,5))
sns.scatterplot(data=scaled_df, x="TSNE1", y="TSNE2", hue="Cluster_KMeans", palette="deep")
plt.title("K-Means Clustering Results (t-SNE 2D)")
plt.show()
```

```
# t-SNE 3D
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')
tsne_3d = TSNE(n_components=3, random_state=42, perplexity=30)
tsne_result_3d = tsne_3d.fit_transform(scaled_df.iloc[:, :-5])
ax.scatter(tsne_result_3d[:, 0], tsne_result_3d[:, 1], tsne_result_3d[:, 2],
           c=scaled_df['Cluster_KMeans'], cmap='rainbow', s=60)
ax.set_title("K-Means Clustering Results (t-SNE 3D)")
plt.show()
```

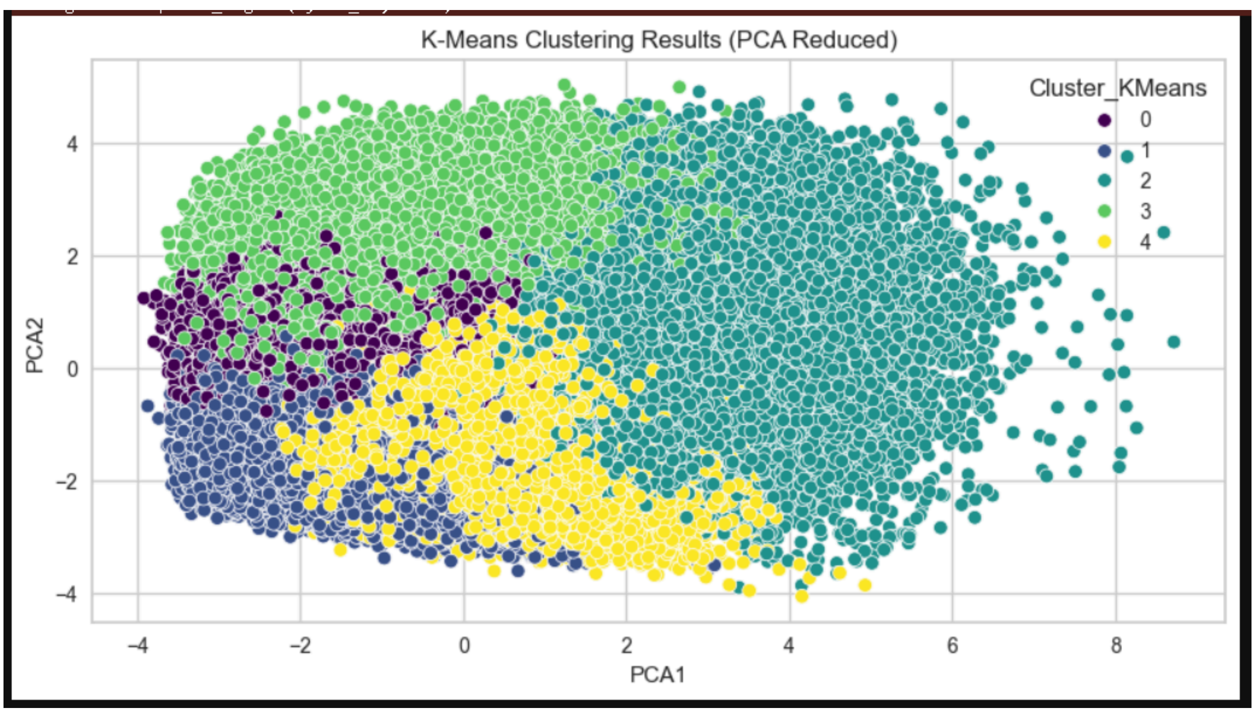
Step 5: Personalized Recommendations

- Each cluster was analyzed to understand dominant trends.
- Custom health & fitness advice was generated per cluster using a defined mapping function.

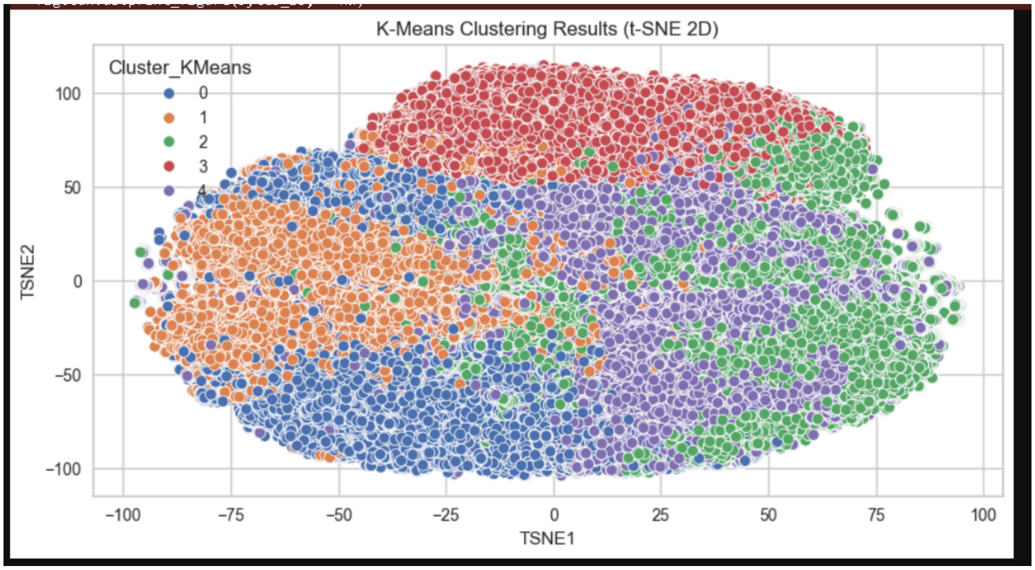
```
def recommend_plan(cluster_label):
    recommendations = {
        0: "Increase daily steps and hydration, reduce stress levels.",
        1: "Maintain balanced workouts and optimize sleep routine.",
        2: "Focus on improving endurance with high-intensity workouts.",
        3: "Incorporate strength training and monitor blood pressure closely.",
        4: "Work on weight management and cardiovascular health strategies."
    }
    return recommendations.get(cluster_label, "No recommendation available.")

scaled_df['Recommendation'] = scaled_df['Cluster_KMeans'].apply(recommend_plan)

# Show Recommendations
print(scaled_df[['Cluster_KMeans', 'Recommendation']].drop_duplicates())
```

Visualizations

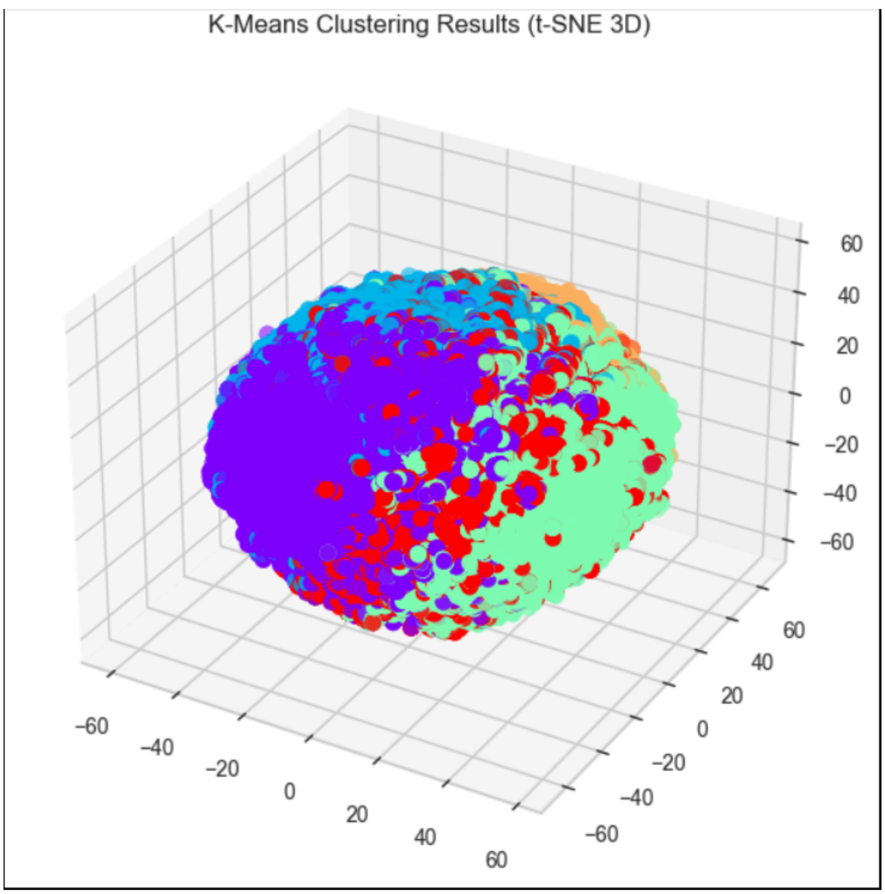
- PCA Cluster Plot



- t-SNE 2D Cluster Plot



- t-SNE 3D Cluster Plot



Personalized Recommendation Output

```
     Cluster_KMeans                                     Recommendation
0                 1  Maintain balanced workouts and optimize sleep ...
6                 3  Incorporate strength training and monitor bloo...
107               2  Focus on improving endurance with high-intensi...
121               4  Work on weight management and cardiovascular h...
234               0  Increase daily steps and hydration, reduce str...
```

```
# Silhouette Score for K-Means
silhouette_kmeans = silhouette_score(scaled_df.iloc[:, :-1], scaled_df['Cluster_KMeans'])
print(f"K-Means Silhouette Score: {silhouette_kmeans}")
```

```
K-Means Silhouette Score: 0.069090670656O2925
```

Conclusion

This project illustrates how **unsupervised learning techniques** like K-Means and dimensionality reduction tools like PCA/t-SNE can uncover **hidden user profiles** in health data. The outcome facilitates **personalized recommendations**, which are crucial for real-world applications in fitness tracking and preventive healthcare analytics.