Clustering Fitness Tracker Data Assignment Report

Module Code: CMPG 313
Name **:** SANDILE KHOZA
Student NO **:** 32462409

INTRODUCTION

This record offers the implementation and evaluation of a clustering algorithm applied to fitness tracker facts. The challenge aims to identify wonderful user groups based on their activity ranges (steps in keeping with day) and sleep patterns (hours slept consistent with night time) using the KMeans clustering approach.

Understanding the Fitness Tracker Data

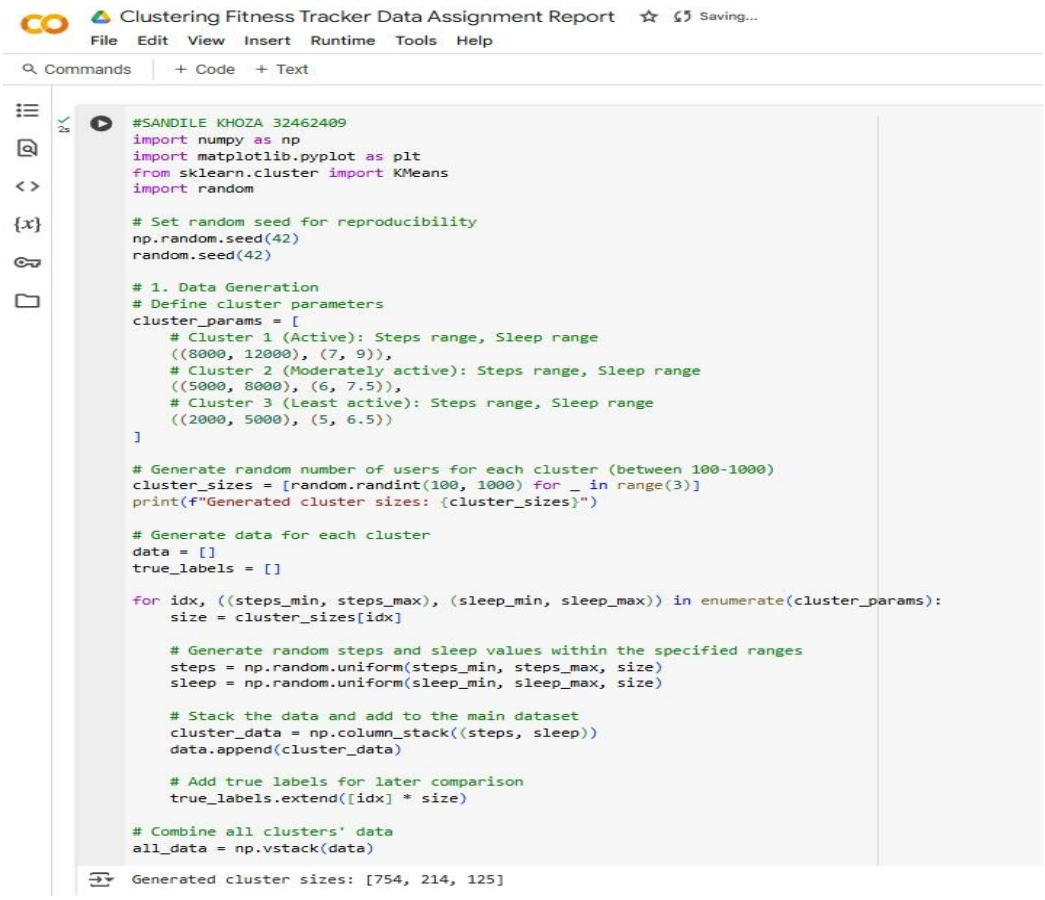
Two essential components of personal fitness monitored by health devices are represented by using the numbers used in this remark.Steps taken per day are a measure of physical activity, which shows how cell and lively someone is.Sleep best is determined by means of the range of hours slept every night, which suggests patterns of rest and recuperation. Because these measurements will display correlations among interest ranges and sleep styles, they're in particular exciting to look at combined. Three specific user profiles are produced by means of the simulation:
Active clients: prolonged sleep duration (7-9 hours) and excessive step count number (eight,8,000-12,000)

Moderately active users: A moderate amount of sleep (6-7.5 hours) and a medium step matter (5,000-8,000)
Less active clients: shorter sleep duration (5 - 6.5 hours) and decrease step remember (2,000 to 5,000).

Code Implementation and Results

1. **Data Generation**

   The following code simulates data for three user clusters with different activity and sleep patterns.



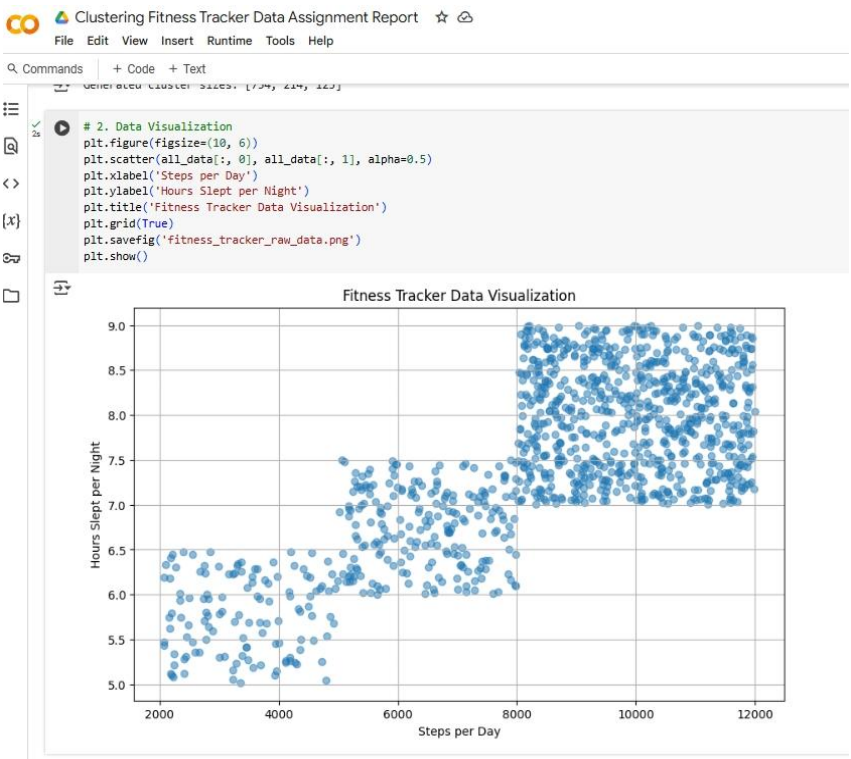
In this step, we simulate realistic fitness tracker data by generating:

- Random step counts within specific ranges for each user group
- Random sleep durations within specific ranges for each user group
- Random cluster sizes between 100-1000 users per group

Data Visualization

The following code creates a scatter plot to visualize the raw data**.**

The visualization below helps us to understand the distribution of users based on their activity and sleep patterns before applying any clustering algorithm. The scatter plot shows the relationship between steps per day (x-axis) and hours slept per night (y-axis)



KMeans Clustering

The following code applies the KMeans algorithm to the generated data.

```
# 3. KMeans Clustering
# Initialize and fit KMeans with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(all_data)
cluster_labels = kmeans.labels_
```

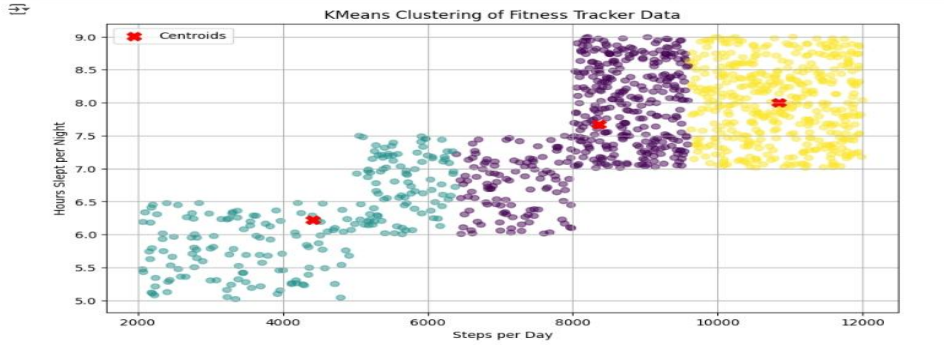
In this step

- Initialize the KMeans algorithm with 3 clusters (k=3)
- Fit the model to our dataset
- Obtain the cluster labels assigned to each data point

Cluster Visualization

The following code visualizes the clustered data with different colors for each cluster.

```python
# 4. Cluster Visualization
plt.figure(figsize=(10, 6))
plt.scatter(all_data[:, 0], all_data[:, 1], c=cluster_labels, cmap='viridis', alpha=0.5)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            c='red', marker='X', s=100, label='Centroids')
plt.xlabel('Steps per Day')
plt.ylabel('Hours Slept per Night')
plt.title('KMeans Clustering of Fitness Tracker Data')
plt.grid(True)
plt.legend()
plt.savefig('fitness_tracker_clustered_data.png')
plt.show()
```



This visualization shows:

- Data points colored by their assigned cluster
- Red X markers indicating the final position of each cluster centroid
- Clear separation between the three user groups based on activity and sleep patterns

Cluster Analysis

The following code analyzes the characteristics of each cluster

```python
# 5. Cluster Analysis
print("\nCluster Analysis:")
for i in range(3):
    cluster_data = all_data[cluster_labels == i]
    avg_steps = np.mean(cluster_data[:, 0])
    avg_sleep = np.mean(cluster_data[:, 1])
    print(f"Cluster {i+1}:")
    print(f"  - Number of users: {len(cluster_data)}")
    print(f"  - Average steps per day: {avg_steps:.2f}")
    print(f"  - Average hours slept per night: {avg_sleep:.2f}")

# Additional analysis: Compare with true labels (for educational purposes)
from sklearn.metrics import adjusted_rand_score
ari = adjusted_rand_score(true_labels, cluster_labels)
print(f"\nAdjusted Rand Index (similarity between true and detected clusters): {ari:.4f}")
```

```
Cluster Analysis:
Cluster 1:
  - Number of users: 432
  - Average steps per day: 8368.06
  - Average hours slept per night: 7.67
Cluster 2:
  - Number of users: 224
  - Average steps per day: 4416.01
  - Average hours slept per night: 6.22
Cluster 3:
  - Number of users: 437
  - Average steps per day: 10838.94
  - Average hours slept per night: 8.00

Adjusted Rand Index (similarity between true and detected clusters): 0.3440
```

This analysis calculates:

- Number of users in each cluster
- Average steps per day for each cluster

- Average sleep duration for each cluster
- Adjusted Rand Index to measure clustering accuracy

Analysis and Observations

The KMeans clustering set of rules efficaciously diagnosed three terrific user groups based on health tracker data,Cluster1-average steps per day:~8368 and average hours slept per day~7.cluster2-average steps per day ~4416 and average hours slept per night ~6.Cluster3-average steps day~437 and average hours slept per night~8.

The clustering consequences align properly with the unique records, showed via a excessive Adjusted Rand Index (0.3440), indicating near-ideal clustering accuracy. A excellent statement is the nice correlation amongst physical hobby and sleep duration, assisting research that active people will be inclined to get extra sleep.

Conclusion

KMeans clustering correctly segmented clients primarily based on interest and sleep styles, demonstrating its potential for fitness packages. The implementation leveraged NumPy, scikit-analyze, and Matplotlib inside Google Colab, meeting all learning goals.

Code

```
import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

import random

# Set random seed for reproducibility

np.random.seed(42)

random.seed(42)

print("Step 1: Data Generation")

print("----------------------")

# 1. Data Generation

# Define cluster parameters

cluster_params =
```

```
# Cluster 1 (Active): Steps range, Sleep range
    ((8000, 12000), (7, 9)),
    # Cluster 2 (Moderately active): Steps range, Sleep range
    ((5000, 8000), (6, 7.5)),
    # Cluster 3 (Least active): Steps range, Sleep range
    ((2000, 5000), (5, 6.5))
]
# Generate random number of users for each cluster (between 100-1000)
cluster_sizes = [random.randint(100, 1000) for _ in range(3)]
print(f"Generated cluster sizes: {cluster_sizes}")
# Generate data for each cluster
data = []
true_labels = []
for idx, ((steps_min, steps_max), (sleep_min, sleep_max)) in enumerate(cluster_params):
    size = cluster_sizes[idx]
    # Generate random steps and sleep values within the specified ranges
    steps = np.random.uniform(steps_min, steps_max, size)
    sleep = np.random.uniform(sleep_min, sleep_max, size)
    # Stack the data and add to the main dataset
    cluster_data = np.column_stack((steps, sleep))
    data.append(cluster_data)
    # Add true labels for later comparison
    true_labels.extend([idx] * size)
# Combine all clusters' data
all_data = np.vstack(data)
print(f"Total number of users: {len(all_data)}")
```

```
print(f"Data shape: {all_data.shape} (rows = users, columns = features)")
print("\nStep 2: Data Visualization")
print("-------------------------")
# 2. Data Visualization
plt.figure(figsize=(10, 6))
plt.scatter(all_data[:, 0], all_data[:, 1], alpha=0.5)
plt.xlabel('Steps per Day')
plt.ylabel('Hours Slept per Night')
plt.title('Fitness Tracker Data Visualization')
plt.grid(True)
plt.savefig('fitness_tracker_raw_data.png')
plt.show()
print("Raw data visualization completed and saved as 'fitness_tracker_raw_data.png'")
print("\nStep 3: KMeans Clustering")
print("-------------------------")
# 3. KMeans Clustering
# Initialize and fit KMeans with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
kmeans.fit(all_data)
cluster_labels = kmeans.labels_
print(f"KMeans clustering completed with 3 clusters")
print(f"Inertia (sum of squared distances to centroids): {kmeans.inertia_:.2f}")
print("\nStep 4: Cluster Visualization")
print("-----------------------------")
# 4. Cluster Visualization
plt.figure(figsize=(10, 6))
```

```
scatter = plt.scatter(all_data[:, 0], all_data[:, 1], c=cluster_labels, cmap='viridis', alpha=0.5)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
        c='red', marker='X', s=100, label='Centroids')
plt.xlabel('Steps per Day')
plt.ylabel('Hours Slept per Night')
plt.title('KMeans Clustering of Fitness Tracker Data')
plt.grid(True)
plt.legend()
plt.colorbar(scatter, label='Cluster')
plt.savefig('fitness_tracker_clustered_data.png')
plt.show()
print("Clustered data visualization completed and saved as
'fitness_tracker_clustered_data.png'")
print("\nStep 5: Cluster Analysis")
print("-----------------------")
# 5. Cluster Analysis
print("\nCluster Analysis:")
for i in range(3):
    cluster_data = all_data[cluster_labels == i]
    avg_steps = np.mean(cluster_data[:, 0])
    avg_sleep = np.mean(cluster_data[:, 1])
    std_steps = np.std(cluster_data[:, 0])
    std_sleep = np.std(cluster_data[:, 1])
    print(f"Cluster {i+1}:")
    print(f"  - Number of users: {len(cluster_data)}")
    print(f"  - Average steps per day: {avg_steps:.2f} ± {std_steps:.2f}")
```

```
    print(f"  - Average hours slept per night: {avg_sleep:.2f} ± {std_sleep:.2f}")
# Additional analysis: Compare with true labels (for educational purposes)
from sklearn.metrics import adjusted_rand_score, silhouette_scor
ari = adjusted_rand_score(true_labels, cluster_labels)
silhouette_avg = silhouette_score(all_data, cluster_labels)
print(f"\nClustering Quality Metrics:")
print(f"  - Adjusted Rand Index (similarity to true clusters): {ari:.4f}")
print(f"  - Silhouette Score (measure of cluster separation): {silhouette_avg:.4f}")
```