

```
In [1]: import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

```
In [2]: data = pd.read_csv("fitness_tracker_data.csv")
data
```

```
Out[2]:
```

	user_id	age	gender	steps_per_day	active_minutes	calories_burned	heart_rate_avg	sleep_hours	stress_level	goal_achieved
0	8270	46	Male	8304	254	2717	109	8.4	Moderate	False
1	1860	56	Male	9345	12	2810	113	7.3	High	False
2	6390	20	Male	17352	72	2270	73	7.9	Moderate	False
3	6191	49	Female	2597	58	2257	109	5.6	Low	True
4	6734	27	Female	4336	85	1635	74	7.6	High	False
...
995	6232	30	Male	5664	217	2840	96	8.6	Low	False
996	6797	21	Female	15050	229	1914	62	9.6	Low	True
997	5926	43	Female	2680	199	3308	68	6.9	Low	False
998	7016	42	Male	6357	293	2547	112	5.9	Moderate	True
999	4335	47	Male	3363	246	3373	77	8.7	Low	False

1000 rows × 10 columns

```
In [3]: # Use only numerical columns for clustering
numerical_cols = ["age", "steps_per_day", "active_minutes", "calories_burned", "heart_rate_avg", "sleep_hours"]
X = data[numerical_cols]
```

```
In [4]: # Scale the numerical features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
In [5]: # Apply K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42) # Set 3 clusters (can adjust based on domain knowledge)
clusters = kmeans.fit_predict(X_scaled)

# Add cluster labels to the original dataset
data["cluster"] = clusters
```

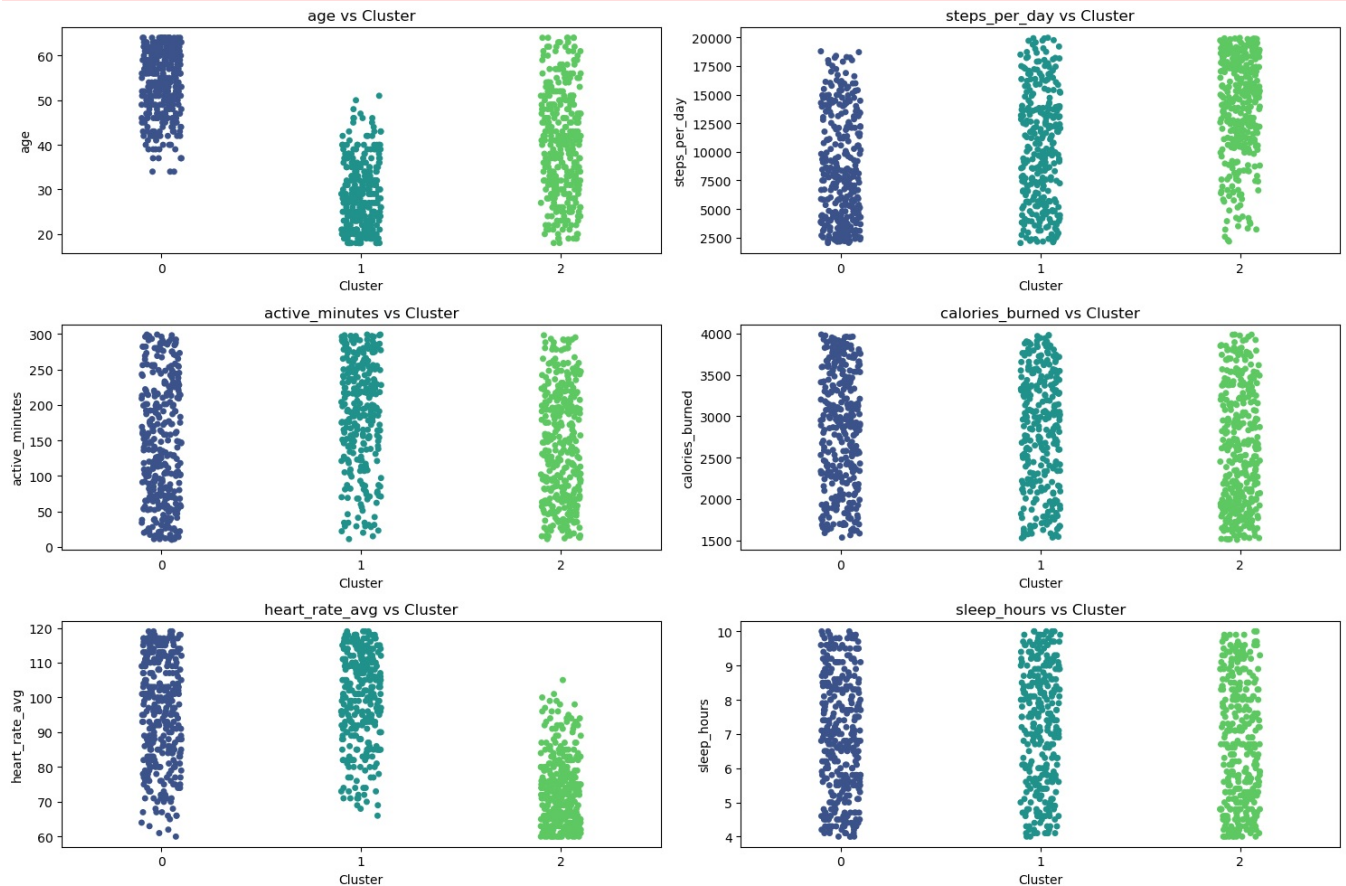
C:\Users\KEERTHANA.R\anaconda3\lib\site-packages\sklearn\cluster_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warning
warnings.warn(
C:\Users\KEERTHANA.R\anaconda3\lib\site-packages\sklearn\cluster_kmeans.py:1382: 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=4.
warnings.warn(

```
In [6]: import matplotlib.pyplot as plt
import seaborn as sns

# Dot plot for each numerical feature with clusters
plt.figure(figsize=(15, 10))
for i, col in enumerate(numerical_cols):
    plt.subplot(3, 2, i + 1)
    sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
    plt.title(f"{col} vs Cluster")
    plt.xlabel("Cluster")
    plt.ylabel(col)

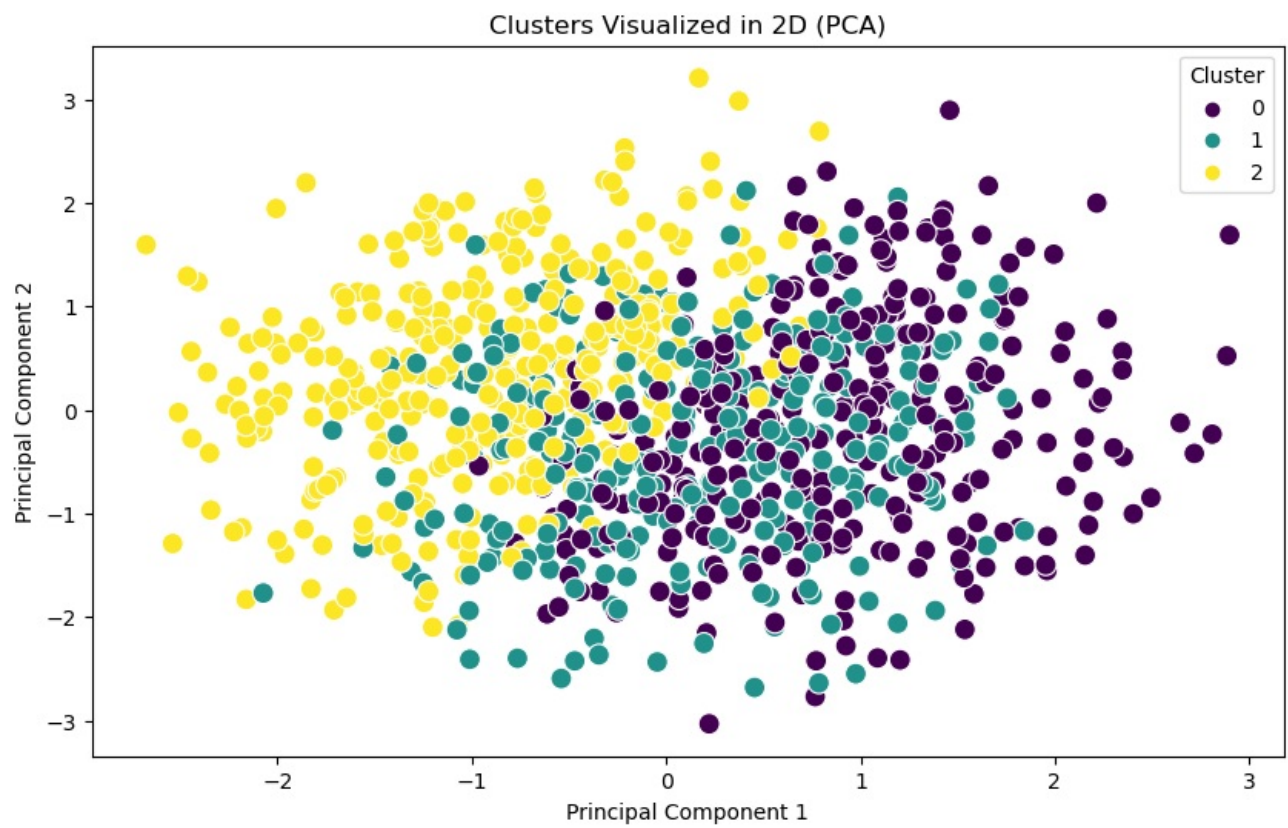
plt.tight_layout()
plt.show()
```

```
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_23996\403759507.py:8: FutureWarning: Passing `palette` without
t assigning `hue` is deprecated.
sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_23996\403759507.py:8: FutureWarning: Passing `palette` without
t assigning `hue` is deprecated.
sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_23996\403759507.py:8: FutureWarning: Passing `palette` without
t assigning `hue` is deprecated.
sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_23996\403759507.py:8: FutureWarning: Passing `palette` without
t assigning `hue` is deprecated.
sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_23996\403759507.py:8: FutureWarning: Passing `palette` without
t assigning `hue` is deprecated.
sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
C:\Users\KEERTHANA.R\AppData\Local\Temp\ipykernel_23996\403759507.py:8: FutureWarning: Passing `palette` without
t assigning `hue` is deprecated.
sns.stripplot(x="cluster", y=col, data=data, jitter=True, palette="viridis")
```



```
In [7]: # Apply PCA to reduce dimensions to 2
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Create a scatter plot of clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=data["cluster"], palette="viridis", s=100)
plt.title("Clusters Visualized in 2D (PCA)")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.legend(title="Cluster")
plt.show()
```



```
In [8]: from sklearn.metrics import silhouette_score

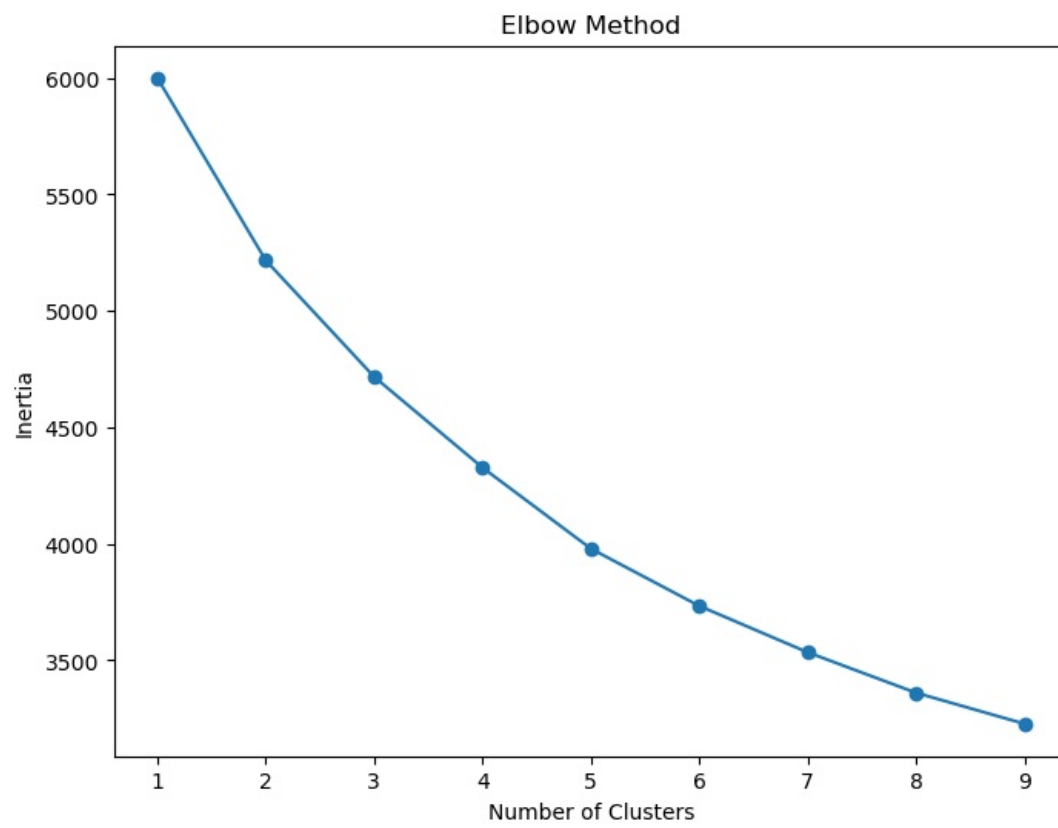
# Calculate silhouette score
sil_score = silhouette_score(X_scaled, clusters)
print(f"Silhouette Score: {sil_score:.2f}")
```

Silhouette Score: 0.12

```
In [9]: distortions = []
for k in range(1, 10):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    distortions.append(kmeans.inertia_)

plt.figure(figsize=(8, 6))
plt.plot(range(1, 10), distortions, marker='o')
plt.title("Elbow Method")
plt.xlabel("Number of Clusters")
plt.ylabel("Inertia")
plt.show()
```

[illegible]



In []:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js