Sri Sivasubramaniya Nadar College of Engineering, Chennai
(An autonomous Institution affiliated to Anna University)

Degree & Branch	B.E. Computer Science & Engineering	Semester	V
Subject Code & Name	ICS1512 - Machine Learning Algorithms Laboratory		
Academic year	2025-2026 (Odd)	Batch: 2023-2028	**Due date:**

Experiment 7: Clustering Human Activity Recognition Data using K-Means, DBSCAN, and Hierarchical Clustering

1 Aim:

To implement and analyze the performance of clustering algorithms on the Human Activity Recognition dataset:

a) Model A: K-Means Clustering.

b) Model B: DBSCAN (Density-Based Spatial Clustering of Applications with Noise).

c) Model C: Hierarchical Agglomerative Clustering (HAC)

Students must visualize and compare the clusters formed by each algorithm, report results, and analyze performance.

2 Libraries used:

- Pandas
- Numpy
- Matplotlib
- Scikit-learn
- Seaborn

3 Objective:

To implement K-Means, DBSCAN, and Hierarchical Agglomerative Clustering on the Human Activity Recognition dataset, visualize and compare the resulting clusters, and evaluate their performance using internal and external metrics.

[46]:
```
# LOADING THE DATASET
X_train = pd.read_csv('/content/drive/MyDrive/ml-lab/ass7/train/X_train.txt',␣
 ↪sep="\\s+", header=None)
y_train = pd.read_csv('/content/drive/MyDrive/ml-lab/ass7/train/y_train.txt',␣
 ↪sep="\\s+", header=None)
subjects = pd.read_csv('/content/drive/MyDrive/ml-lab/ass7/train/subject_train.
 ↪txt', sep="\\s+", header=None)

print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("subjects shape:", subjects.shape)
```

```
X_train shape: (7352, 561)
y_train shape: (7352, 1)
subjects shape: (7352, 1)
```

[47]:
```
X_train.head()
```

[47]:

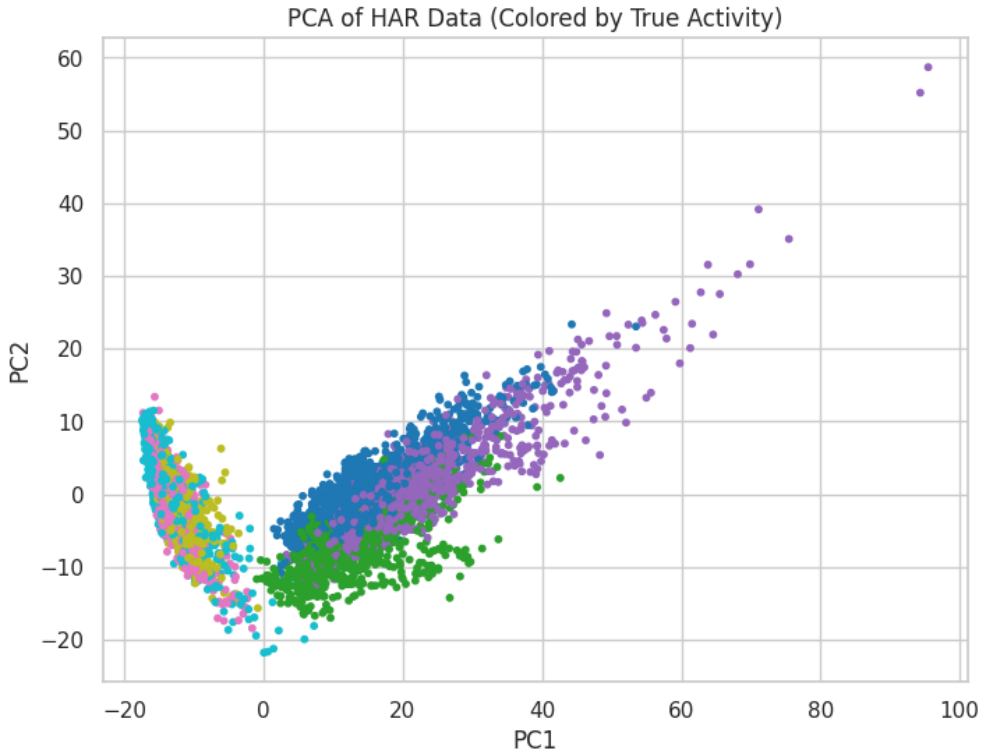
	0	1	2	3	4	5	6	...	551	552	553	554	555	556	557	558	559	560
0	0.288585	-0.020294	-0.132905	-0.995279	-0.983111	-0.913526	-0.995112	...	-0.074323	-0.298676	-0.710304	-0.112754	0.030400	-0.464761	-0.018446	-0.841247	0.179941	-0.058627
1	0.278419	-0.016411	-0.123520	-0.998245	-0.975300	-0.960322	-0.998807	...	0.158075	-0.595051	-0.861499	0.053477	-0.007435	-0.732626	0.703511	-0.844788	0.180289	-0.054317
2	0.279653	-0.019467	-0.113462	-0.995380	-0.967187	-0.978944	-0.996520	...	0.414503	-0.390748	-0.760104	-0.118559	0.177899	0.100699	0.808529	-0.848933	0.180637	-0.049118
3	0.279174	-0.026201	-0.123283	-0.996091	-0.983403	-0.990675	-0.997099	...	0.404573	-0.117290	-0.482845	-0.036788	-0.012892	0.640011	-0.485366	-0.848649	0.181935	-0.047663
4	0.276629	-0.016570	-0.115362	-0.998139	-0.980817	-0.990482	-0.998321	...	0.087753	-0.351471	-0.699205	0.123320	0.122542	0.693578	-0.615971	-0.847865	0.185151	-0.043892

5 rows x 561 columns

[48]:
```
# STANDARDIZATION
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)
```

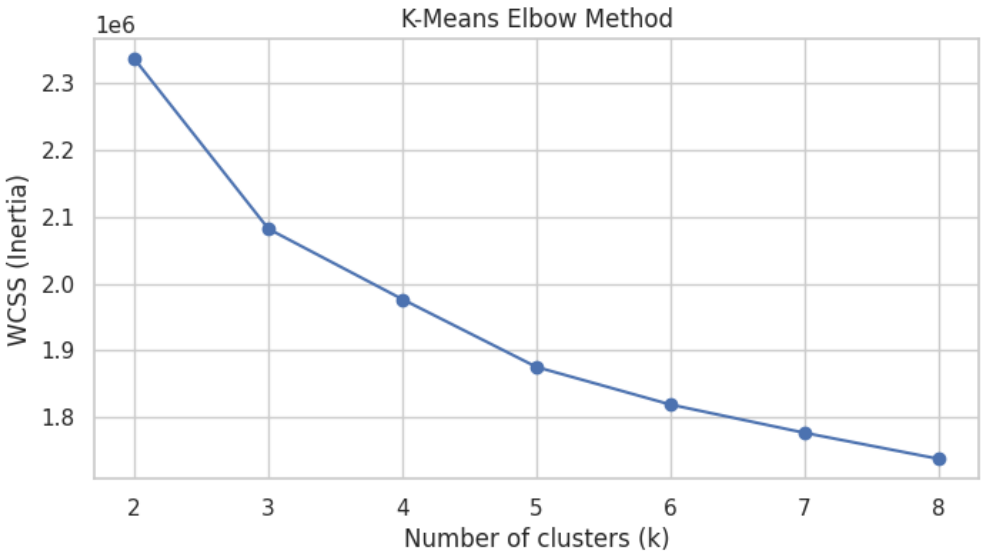
[49]:
```
# DIMENSIONALITY REDUCTION (for visualization)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)
```

```
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=y_train[0], cmap='tab10', s=10)
plt.title('PCA of HAR Data (Colored by True Activity)')
plt.xlabel('PC1'); plt.ylabel('PC2')
plt.show()
```

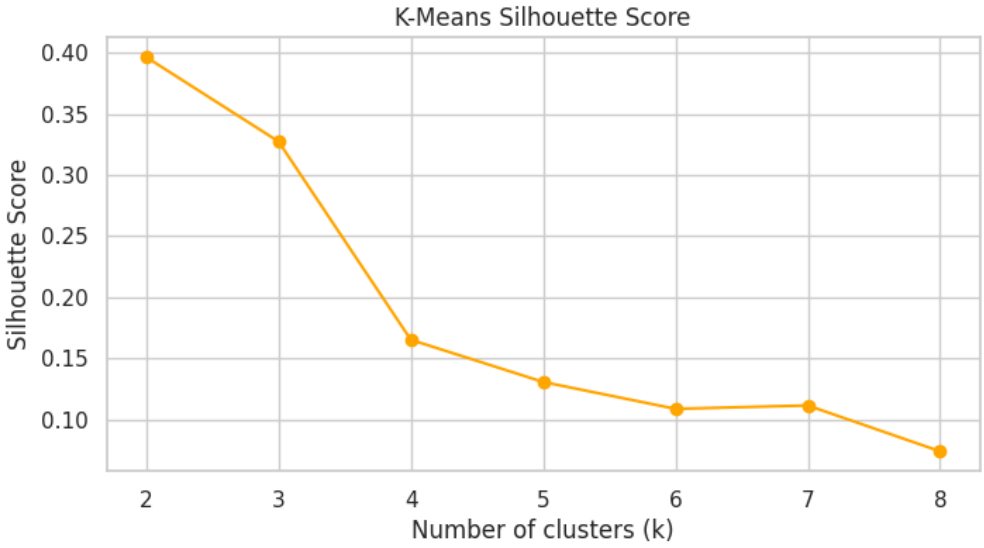


K-Means Clustering + Elbow/Sillhouette

[64]:
```
wcss = []
sil_scores = []
K_range = range(2,9)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    labels = kmeans.fit_predict(X_scaled)
    wcss.append(kmeans.inertia_)
    sil_scores.append(silhouette_score(X_scaled, labels))

plt.figure(figsize=(8,4))
plt.plot(K_range, wcss, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('WCSS (Inertia)')
plt.title('K-Means Elbow Method')
plt.show()
```



[65]:
```
# Silhouette curve
plt.figure(figsize=(8,4))
plt.plot(K_range, sil_scores, marker='o', color='orange')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('K-Means Silhouette Score')
plt.show()
```

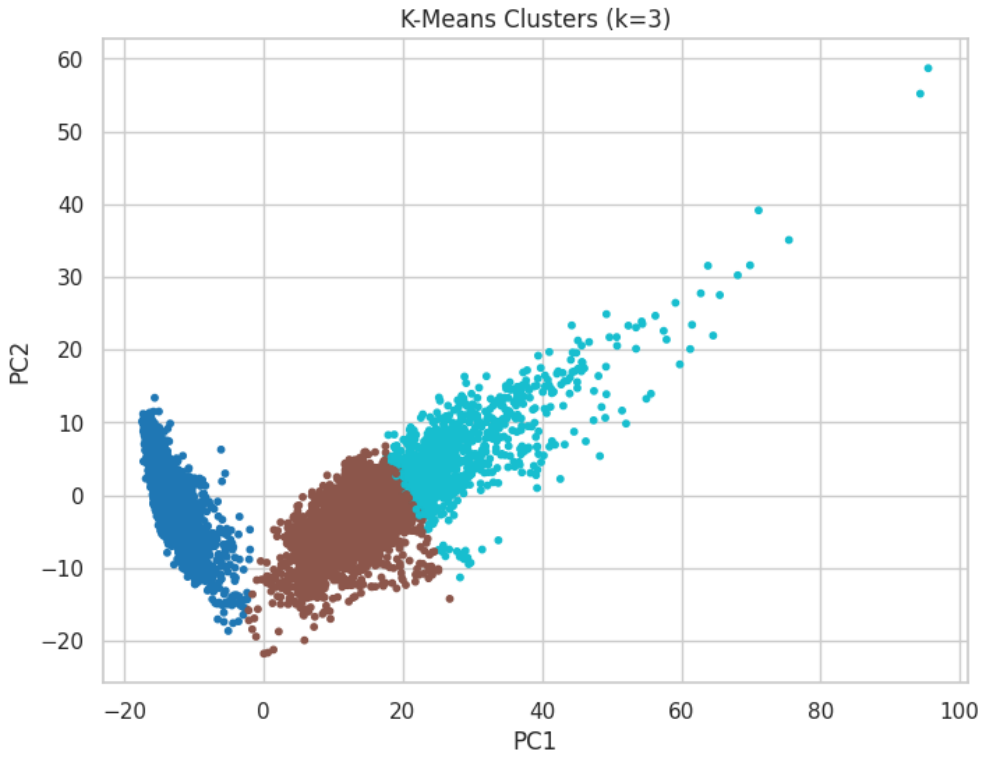


[68]:
```
table = pd.DataFrame({
        "k": list(K_range),
        "WCSS": wcss,
        "Silhouette": sil_scores
    })
print(table.to_string(index=False))
```

```
k          WCSS  Silhouette
2 2.336224e+06    0.396505
3 2.081850e+06    0.327408
4 1.976453e+06    0.165163
5 1.875066e+06    0.130639
6 1.818790e+06    0.108641
7 1.776519e+06    0.111424
8 1.737555e+06    0.073861
```

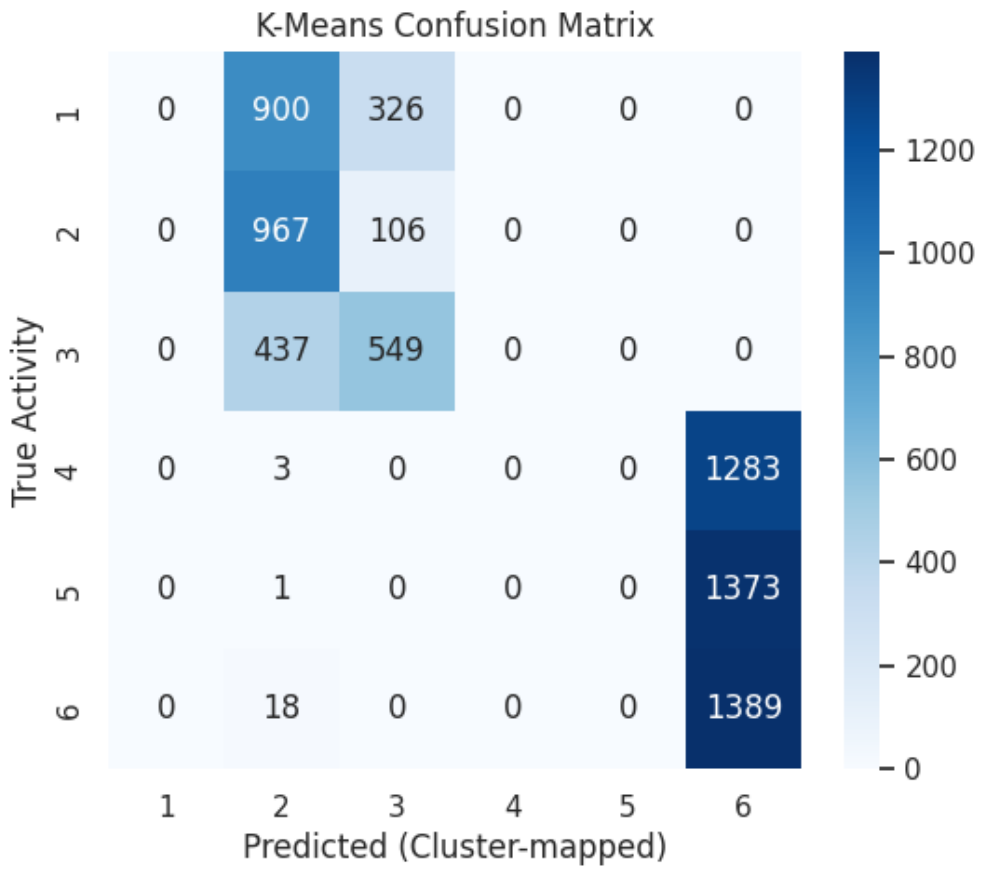
[80]:
```python
# choosing best k
k_best = 3
kmeans = KMeans(n_clusters=k_best, random_state=42, n_init=10)
k_labels = kmeans.fit_predict(X_scaled)
```

[81]:
```python
# PCA visualization of clusters
plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=k_labels, cmap='tab10', s=10)
plt.title(f'K-Means Clusters (k={k_best})')
plt.xlabel('PC1'); plt.ylabel('PC2')
plt.show()
```



[105]:
```python
def cluster_confusion_matrix(true_labels, cluster_labels, algo_name="K-Means"):
    # Map clusters to most common true label
    labels = np.unique(cluster_labels)
    mapping = {}
```

```
    for lbl in labels:
        mask = cluster_labels == lbl
        true_for_cluster = true_labels[mask]
        if len(true_for_cluster) == 0: continue
        mapping[lbl] = np.bincount(true_for_cluster).argmax()

    mapped_labels = np.array([mapping[lbl] for lbl in cluster_labels])

    # Compute confusion matrix
    cm = confusion_matrix(true_labels, mapped_labels)
    cm_df = pd.DataFrame(cm, index=np.unique(true_labels), columns=np.
↪unique(true_labels))

    plt.figure(figsize=(6,5))
    sns.heatmap(cm_df, annot=True, fmt="d", cmap="Blues")
    plt.title(f"{algo_name} Confusion Matrix")
    plt.xlabel("Predicted (Cluster-mapped)")
    plt.ylabel("True Activity")
    plt.show()
    return cm_df

cm_kmeans = cluster_confusion_matrix(y_train[0].values, k_labels, "K-Means")
```

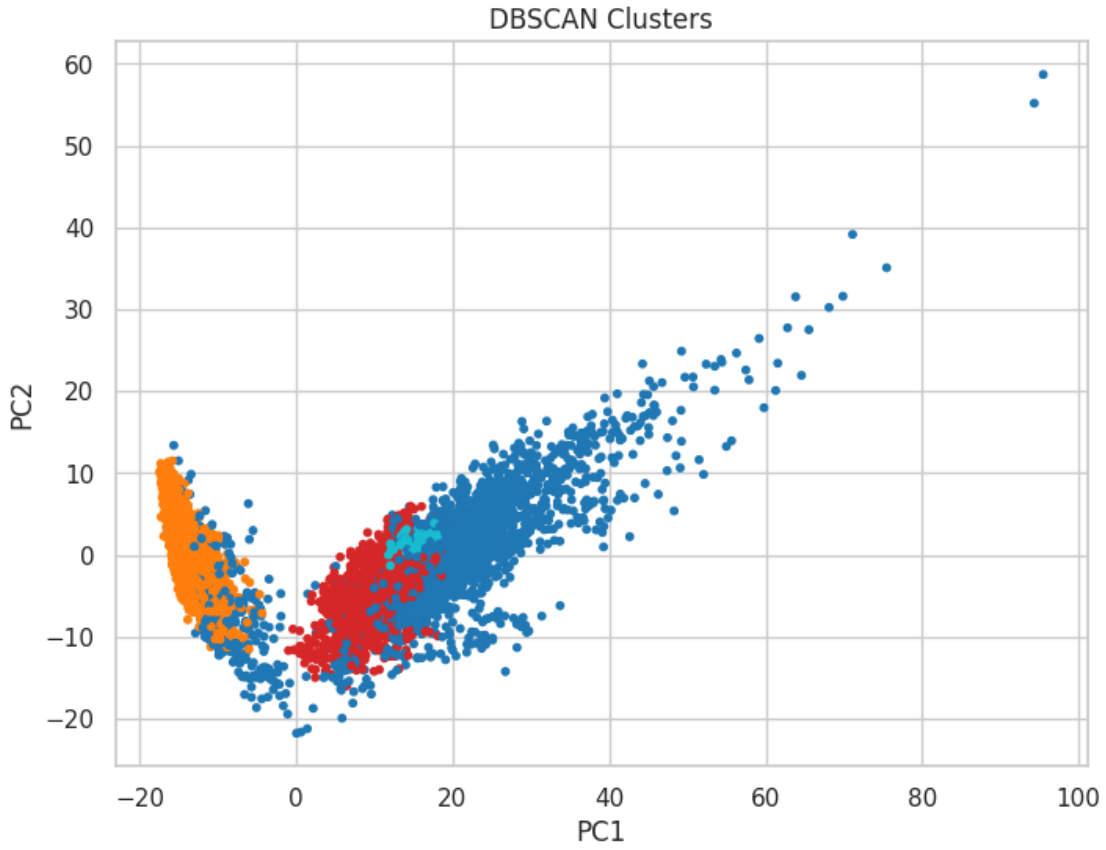


DBScan

[88]:
```
def tune_dbscan(X, eps_values, min_samples_values):
    best_score = -1
    best_params = None
    results = []

    for eps in eps_values:
        for min_s in min_samples_values:
            db = DBSCAN(eps=eps, min_samples=min_s).fit(X)
            labels = db.labels_

            # skip if DBSCAN assigns all points to noise or 1 cluster
            if len(set(labels)) <= 1:
                continue

            try:
                sil = silhouette_score(X, labels)
                dbi = davies_bouldin_score(X, labels)
                ch = calinski_harabasz_score(X, labels)
            except:
                continue

            results.append((eps, min_s, sil, dbi, ch))

            if sil > best_score:
                best_score = sil
                best_params = (eps, min_s)

    return best_params, results

# Example usage
eps_range = np.linspace(10, 15, 20, 25)
min_samples_range = [5, 10, 15, 20]

best_params, results = tune_dbscan(X_scaled, eps_range, min_samples_range)

if best_params is None:
    print("DBSCAN couldn't form valid clusters in the tested range.")
else:
    print("Best params:", best_params)
```

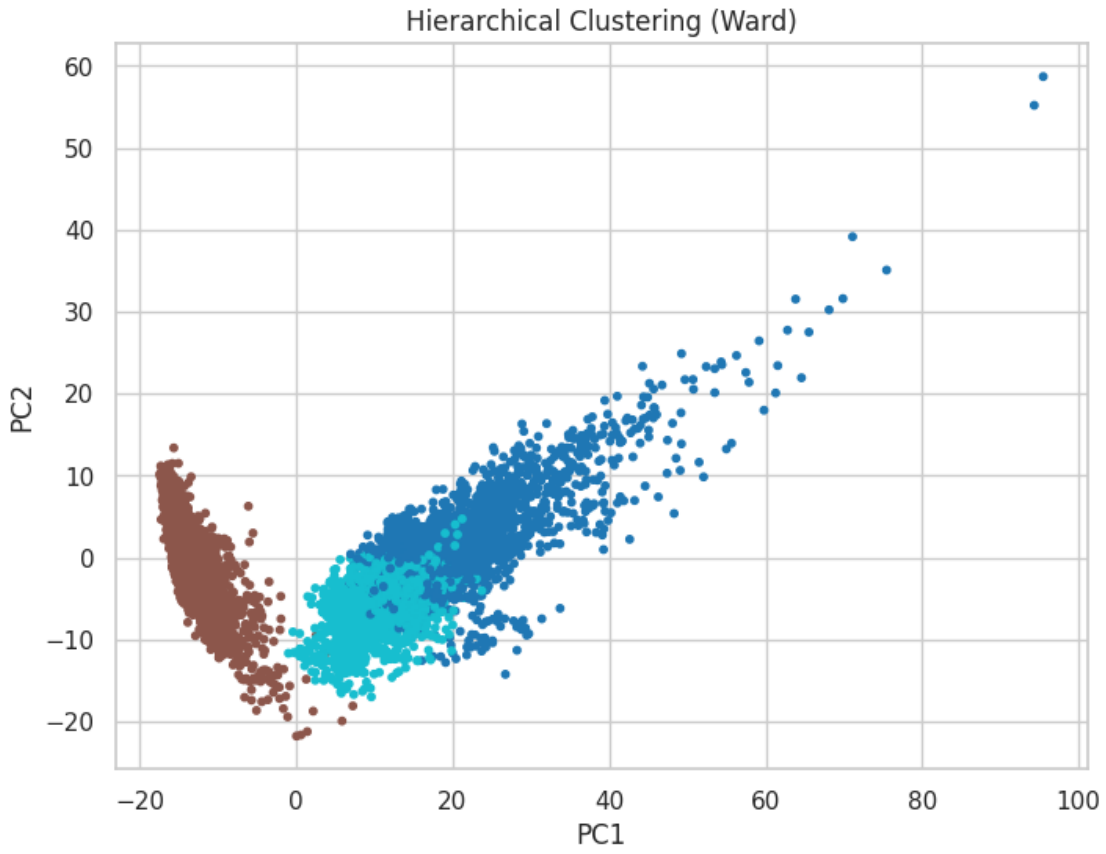
```
Best params: (np.float64(14.0), 20)
```

[91]:
```
dbscan = DBSCAN(eps=14, min_samples=20)
db_labels = dbscan.fit_predict(X_scaled)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=db_labels, cmap='tab10', s=10)
plt.title('DBSCAN Clusters')
plt.xlabel('PC1'); plt.ylabel('PC2')
plt.show()
```



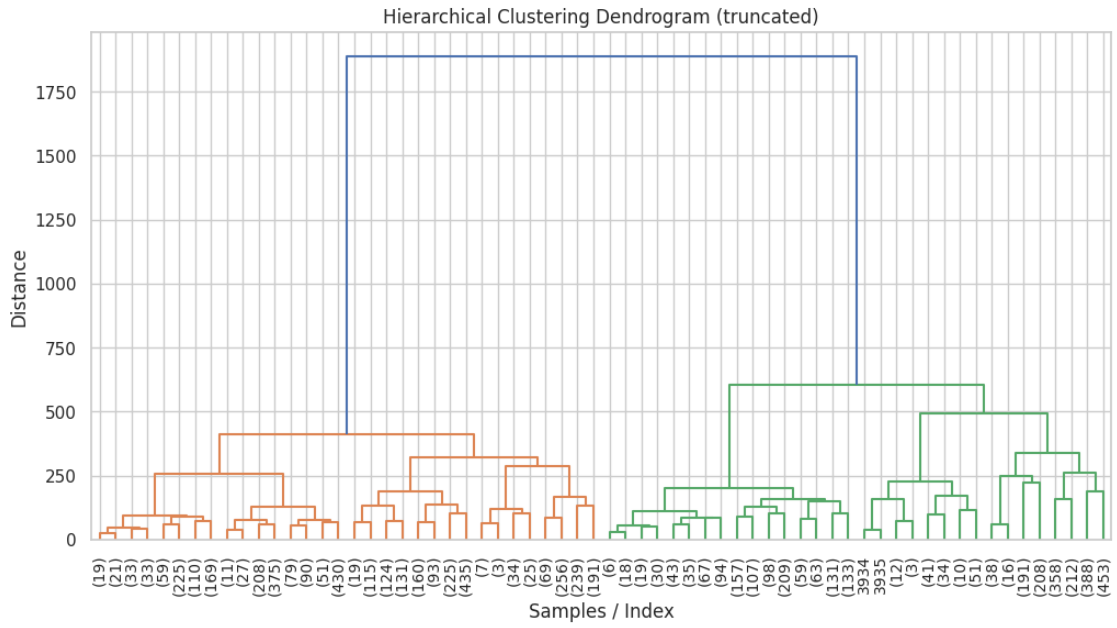
Hierarchical Clustering

[83]:
```
agg = AgglomerativeClustering(n_clusters=k_best, linkage='ward')
agg_labels = agg.fit_predict(X_scaled)

plt.figure(figsize=(8,6))
plt.scatter(X_pca[:,0], X_pca[:,1], c=agg_labels, cmap='tab10', s=10)
plt.title('Hierarchical Clustering (Ward)')
plt.xlabel('PC1'); plt.ylabel('PC2')
plt.show()
```



[93]:

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

# Using Ward linkage
Z = linkage(X_scaled, method='ward')

plt.figure(figsize=(12,6))
dendrogram(Z, truncate_mode='level', p=5, leaf_rotation=90, leaf_font_size=10)
plt.title("Hierarchical Clustering Dendrogram (truncated)")
plt.xlabel("Samples / Index")
plt.ylabel("Distance")
plt.show()
```

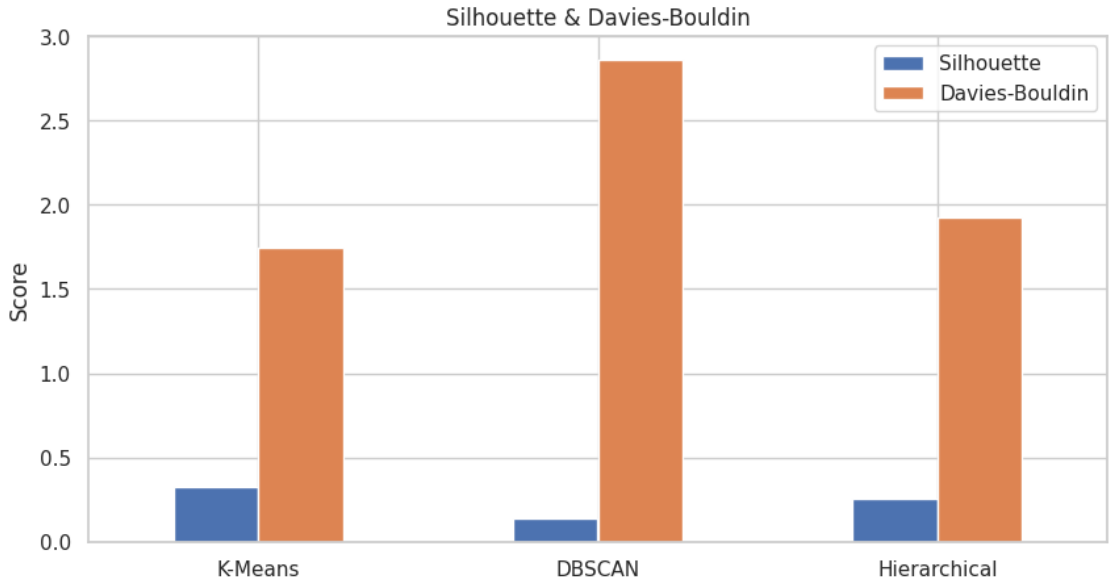


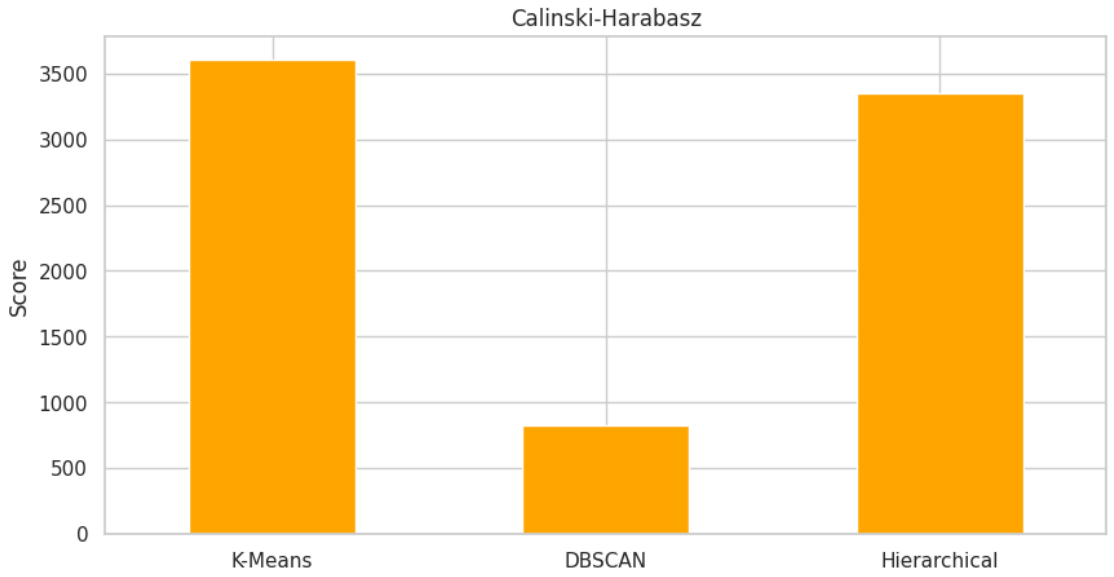
[92]:
```
# EVALUATION METRICES
def clustering_metrics(X, cluster_labels, true_labels=None,␣
 ↪algo_name="Algorithm"):
    metrics = {}
    n_clusters = len(set(cluster_labels)) - (1 if -1 in cluster_labels else 0)

    # Internal metrics (if at least 2 clusters)
    if n_clusters > 1:
        metrics['Silhouette'] = silhouette_score(X, cluster_labels)
        metrics['Davies-Bouldin'] = davies_bouldin_score(X, cluster_labels)
        metrics['Calinski-Harabasz'] = calinski_harabasz_score(X, cluster_labels)

    # External metrics (if true labels provided)
    if true_labels is not None:
        metrics['ARI'] = adjusted_rand_score(true_labels, cluster_labels)
        metrics['NMI'] = normalized_mutual_info_score(true_labels,␣
 ↪cluster_labels)

    return pd.DataFrame({algo_name: metrics}).T

y_true = y_train[0]  # ground truth
k_metrics = clustering_metrics(X_scaled, k_labels, y_true, "K-Means")
db_metrics = clustering_metrics(X_scaled, db_labels, y_true, "DBSCAN")
agg_metrics = clustering_metrics(X_scaled, agg_labels, y_true, "Hierarchical")

all_metrics = pd.concat([k_metrics, db_metrics, agg_metrics])
all_metrics
```

[92]:
```
              Silhouette  Davies-Bouldin  Calinski-Harabasz       ARI  \
K-Means         0.327408        1.745773        3605.263229  0.327923
DBSCAN          0.141134        2.859937         825.391505  0.275990
Hierarchical    0.254790        1.927494        3350.021309  0.342153

                    NMI
K-Means        0.516310
DBSCAN         0.426156
Hierarchical   0.539664
```

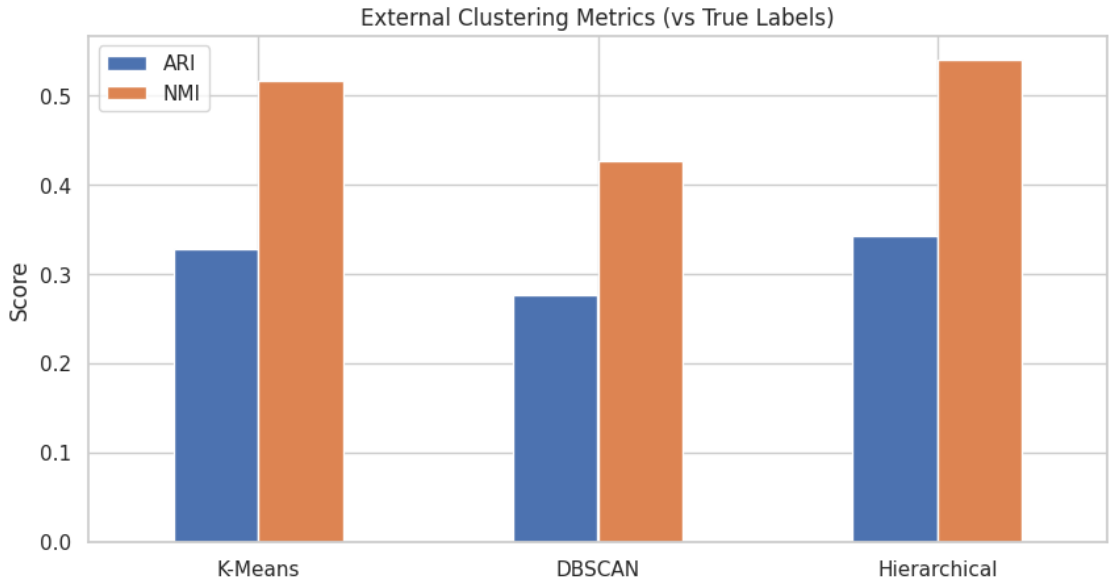
[99]:
```
internal_metrics =␣
↪all_metrics[['Silhouette','Davies-Bouldin','Calinski-Harabasz']]
external_metrics = all_metrics[['ARI','NMI']]
```

[102]:
```
# Plot Silhouette + Davies-Bouldin
internal_metrics[['Silhouette','Davies-Bouldin']].plot(kind='bar',␣
↪figsize=(10,5))
plt.title('Silhouette & Davies-Bouldin')
plt.ylabel('Score')
plt.xticks(rotation=0)
plt.show()

# Plot Calinski-Harabasz separately
internal_metrics['Calinski-Harabasz'].plot(kind='bar', figsize=(10,5),␣
↪color='orange')
plt.title('Calinski-Harabasz')
plt.ylabel('Score')
plt.xticks(rotation=0)
plt.show()
```





[98]:
```
# External
external_metrics.plot(kind='bar', figsize=(10,5))
plt.title('External Clustering Metrics (vs True Labels)')
plt.ylabel('Score')
plt.xticks(rotation=0)
plt.show()
```



4 Summary Tables

K-Means Elbow Method Results

Table 1: K-means Elbow Method Results

Number of Clusters (k)	WCSS (Inertia)	Silhouette Score
2	2.336224e+06	0.396505
3	2.081850e+06	0.327408
4	1.976453e+06	0.165163
5	1.875066e+06	0.130639
6	1.818790e+06	0.108641
7	1.776519e+06	0.111424
8	1.737555e+06	0.073861

Evaluation Metrics (Internal and External)

Table 2: Clustering Algorithm Evaluation Metrics

Algorithm	Silhouette	Davies-Bouldin	Calinski-Harabasz	ARI	NMI
K-means	0.327408	1.745773	3605.263229	0.327923	0.516310
DBSCAN	0.141134	2.859937	825.391505	0.275990	0.426156
Hierarchical	0.254790	1.927494	3350.021309	0.342153	0.539664

5 Observations

- **Which algorithm produced the most meaningful clusters? Why?**
  Hierarchical Clustering appears to produce the most meaningful clusters. It has reasonably good internal metrics (Silhouette: 0.2548, Davies-Bouldin: 1.9275, Calinski-Harabasz: 3350.0) and the highest external metrics (ARI: 0.3422, NMI: 0.5397), indicating a better match with ground truth activities compared to K-Means and DBSCAN.
- **How sensitive was K-Means to the choice of k?**
  K-Means is quite sensitive to k. Silhouette score decreased consistently as k increased beyond 2-3, indicating that choosing too many clusters reduces cluster cohesion. The Elbow method suggested 2-3 clusters, which roughly aligns with the higher Silhouette scores, but the dataset has 6 true activity classes, showing that K-Means struggles to separate all activities clearly.
- **Did DBSCAN detect noise or small clusters effectively?**
  DBSCAN detected some noise points, but the clusters were not very meaningful at default parameters. Internal metrics were low (Silhouette: 0.1411, Davies-Bouldin: 2.8599, Calinski-Harabasz: 825.4), and external metrics (ARI: 0.2760, NMI: 0.4262) were worse than Hierarchical Clustering, showing it struggled to capture all activity classes effectively.
- **How does linkage choice (single/complete/ward) affect hierarchical clustering?**
  Using Ward's linkage tends to produce compact, spherical clusters with better internal metrics, which is why Hierarchical Clustering with Ward's method performed well in this assignment. Single linkage can create long chains (less compact clusters), while complete linkage can be sensitive to outliers.
- **Which internal metric best matched your visual intuition of cluster quality?**
  Silhouette Score best matched the visual cluster quality. Higher Silhouette values correlated with clusters that were visually well-separated in PCA/t-SNE plots, making it easier to assess cluster cohesion and separation.

6 Learning Outcomes

- Understand and implement K-Means, DBSCAN, and Hierarchical Clustering algorithms on real datasets.
- Gain experience with cluster evaluation using internal metrics (Silhouette, Davies-Bouldin, Calinski-Harabasz) and external metrics (ARI, NMI).
- Interpret clustering results visually using PCA/t-SNE and understand how dimensionality reduction helps in analysis.
- Compare clustering algorithms and understand their strengths, weaknesses, and sensitivity to hyperparameters like k, ϵ, minPts, and linkage methods.
- Develop skills to choose optimal hyperparameters using techniques like the Elbow method and metric-based evaluation.

GitHub Repository:
https://github.com/vidarshanaa15/ml-expt-7