In [ ]:

```python
import pickle
from sklearn.cluster import KMeans
from sklearn.metrics import (
    silhouette_score,
    calinski_harabasz_score,
    confusion_matrix,
    accuracy_score,
)
from metrics import beta_cv

# Load processed data
with open("processed_images_data_3.pickle", "rb") as f:
    images_data = pickle.load(f)

with open("processed_hogs_data_3.pickle", "rb") as f:
    hogs_data = pickle.load(f)
```

In [ ]:

```python
images_features = images_data[:, :-1]
hogs_features = hogs_data[:, :-1]
images_labels = images_data[:, -1]
hogs_labels = hogs_data[:, -1]

features = hogs_features  # Replace with hogs_features if needed
labels = hogs_labels  # Replace with hogs_labels if needed

# Create and fit KMeans model
kmeans = KMeans(
    n_clusters=4, random_state=0, max_iter=1000
)  # random_state for reproducibility
kmeans.n_iter_ = 500  # Set number of iterations
print(len(features))
kmeans.fit(features)
```

```
3852
```

Out[ ]:

KMeans

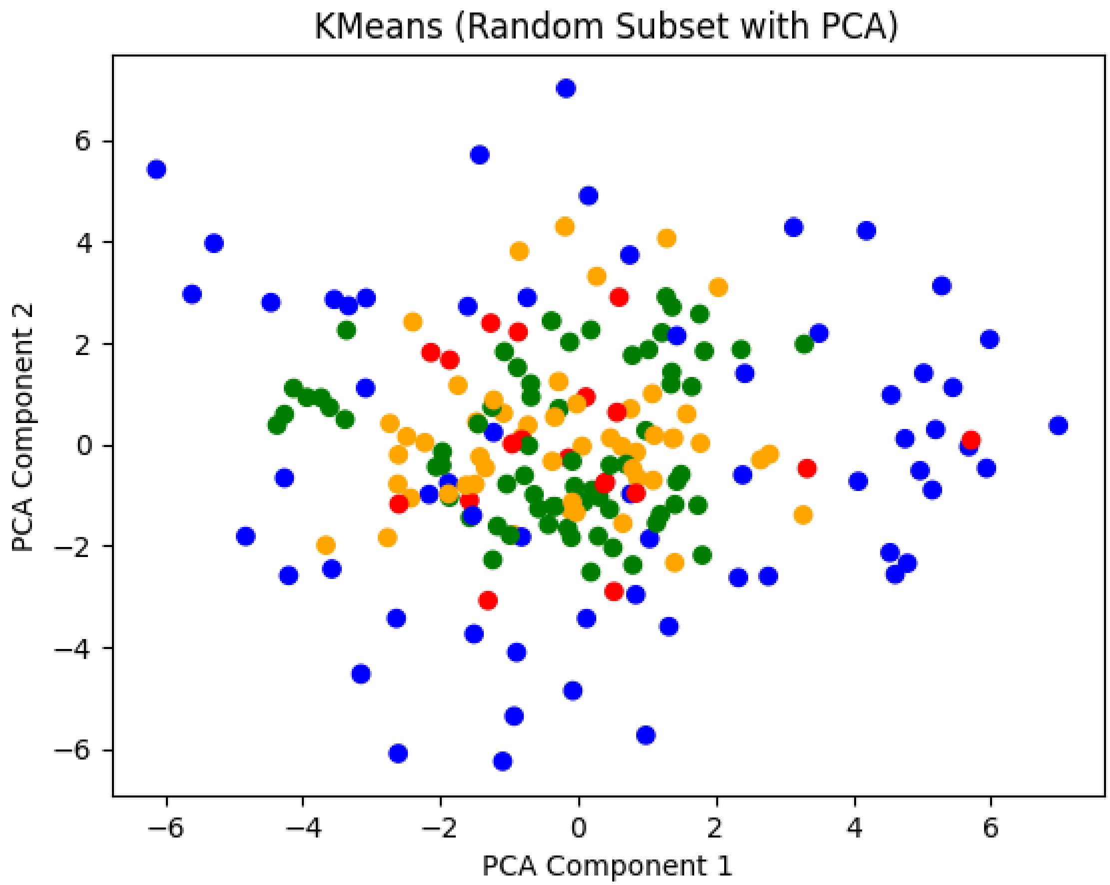
```
KMeans(max_iter=1000, n_clusters=4, random_state=0)
```

In [ ]:
```python
# Predict cluster labels
predicted_labels = kmeans.predict(features)

# Evaluate clustering performance
silhouette_avg = silhouette_score(features, predicted_labels)
calinski_harabasz = calinski_harabasz_score(features, predicted_labels)

print("Silhouette Coefficient:", silhouette_avg)
print("Calinski-Harabasz Index:", calinski_harabasz)


cm = confusion_matrix(labels, predicted_labels)
accuracy = accuracy_score(labels, predicted_labels)
# beta_cv_score = beta_cv(labels, predicted_labels)

# print("Beta CV Score:", beta_cv_score)
print("Confusion Matrix:\n", cm)
# Analyze the confusion matrix:
# - High diagonal values == better
# - Off-diagonal values == misclassifications.

print("Accuracy:", accuracy)  # higher == better
```

```
Silhouette Coefficient: 0.08140483623935373
Calinski-Harabasz Index: 338.03792500609984
Confusion Matrix:
 [[353 185 337 287]
 [603 239 229 121]
 [261 110  93  49]
 [458 208 215 104]]
Accuracy: 0.20482866043613707
```

In [ ]:
```python
import random
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import numpy as np

# Select a random subset of data
subset_size = 200  # Adjust the subset size as needed
random_indices = random.sample(range(len(features)), subset_size)
subset_features = features[random_indices]
```

```python
# Create and fit KMeans model on the subset
kmeans = KMeans(n_clusters=4, random_state=0)
kmeans.fit(subset_features)

# Predict cluster labels for the subset
predicted_labels = kmeans.predict(subset_features)

# Reduce dimensionality using PCA for visualization
pca = PCA(n_components=2)
reduced_features = pca.fit_transform(subset_features)

# Plot clusters
colors = ["blue", "green", "red", "orange"]
colors = np.array(colors)
for i in range(4):
    cluster_data = reduced_features[predicted_labels == i]
    cluster_labels = labels[random_indices][predicted_labels == i]

    # reshape
    cluster_data = np.hstack((cluster_data, np.zeros((cluster_data.shape[0], 1))))
    cluster_data[:, 2] = np.array(cluster_labels)

    plt.scatter(
        cluster_data[:, 0],
        cluster_data[:, 1],
        c=colors[cluster_data[:, 2].astype(int)],
        label=f"Cluster {i}",
    )

plt.title("KMeans (Random Subset with PCA)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.show()
```



Results

Images

Silhouette Coefficient: 0.07437938050979319

Calinski-Harabasz Index: 349.41544056745397

Confusion Matrix:

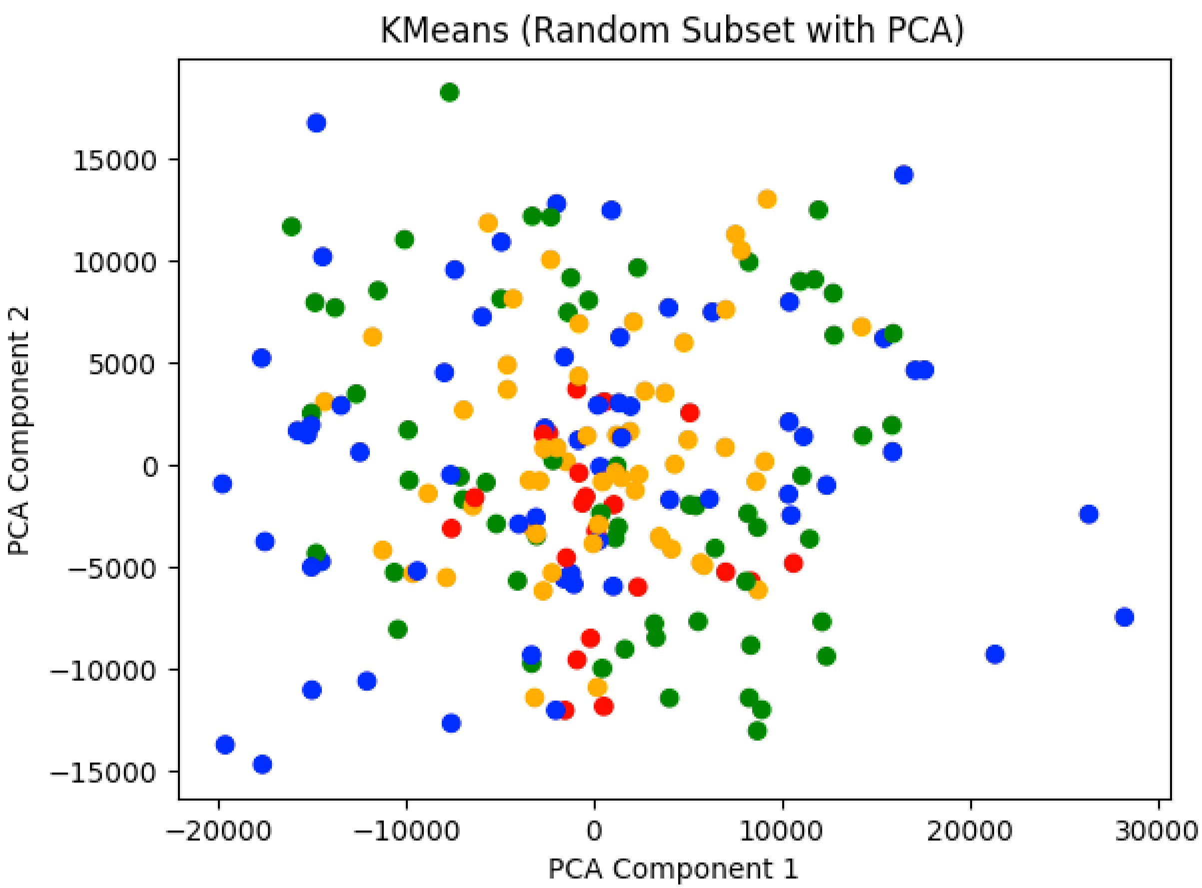
[[258 417 238 249]

[378 362 207 245]

[207 99 29 178]

[383 200 78 324]]

Accuracy: 0.2525960539979232



HOGS (Histogram of Oriented Gradients)

Silhouette Coefficient: 0.08140483623935373

Calinski-Harabasz Index: 338.03792500609984

Confusion Matrix:

[[353 185 337 287]

[603 239 229 121]

[261 110 93 49]

[458 208 215 104]]

Accuracy: 0.20482866043613707