In [1]:
```
import os
import numpy as np
import pandas as pd
from skimage import io,color
from skimage import filters
from skimage import exposure
from sklearn.decomposition import PCA
from sklearn.cluster import BisectingKMeans
from sklearn.cluster import SpectralClustering,AgglomerativeClustering,DBSCAN
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
Dog_images = r'Sreenu_images'
```

In [3]:
```
#Image edge histogram conversion and normalize
```

In [4]:
```
#as specified in assignment 2
def angle(dx, dy):
    return np.mod(np.arctan2(dy, dx), np.pi)

dog_hist = []
dog_breeds     = []
for index,breed_name in enumerate(os.listdir(Dog_images)):
    path=os.path.join(Dog_images,breed_name)
    for image_file in os.listdir(path):
        image    = io.imread(os.path.join(path,image_file.strip()))
        image=color.rgb2gray(image)
        image = angle(filters.sobel_h(image),filters.sobel_v(image))
        hist,_=exposure.histogram(image, nbins=36)
        dog_hist.append(hist/np.sum(hist))
        dog_breeds.append(index)
dog_hist=np.array(dog_hist)
dog_breeds=np.array(dog_breeds)
```

In [5]:
```
# dimensionality reduction using PCA as per assignment 1
pca_algorithm=PCA(n_components=2)
#conversion
dog_hist=pca_algorithm.fit_transform(dog_hist)
```

In [6]:
```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import fowlkes_mallows_score, silhouette_score

# Create an empty DataFrame
m = {'algorithm':[], 'fowlkes_score':[], 'silhouette_score':[]}
eval_scores = pd.DataFrame(m)
kmeans_random = KMeans(n_clusters=4, random_state=42, init="random").fit(dog_hist)
# Calculate evaluation scores and append to the DataFrame
eval_scores = pd.concat([eval_scores, pd.DataFrame({
    'algorithm': ['random'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, kmeans_random.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, kmeans_random.labels_)]
})], ignore_index=True)
```

In [7]:
```
print(type(eval_scores))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [8]:
```
# Initialize and fit KMeans with k-means++ initialization
kmeans_plusplus = KMeans(n_clusters=4, random_state=42, init="k-means++").fit(dog_hist)

# Calculate evaluation scores and append to the DataFrame
eval_scores = pd.concat([eval_scores, pd.DataFrame({
    'algorithm': ['k-means++'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds,
kmeans_plusplus.labels_)],
    'silhouette_score': [silhouette_score(dog_hist,
kmeans_plusplus.labels_)]
})], ignore_index=True)
```

In [9]:
```
# Assuming you've imported BisectingKMeans correctly

# Initialize and fit Bisecting KMeans
bise = BisectingKMeans(n_clusters=4, random_state=42, init="random").fit(dog_hist)

eval_scores = pd.concat([eval_scores, pd.DataFrame({
    'algorithm': ['bisecting means'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, bise.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, bise.labels_)]
})], ignore_index=True)
```

In [10]:
```
# Initialize and fit Spectral Clustering
spc = SpectralClustering(n_clusters=4).fit(dog_hist)

new_eval_scores = pd.DataFrame({
    'algorithm': ['spectral'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, spc.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, spc.labels_)]
```

```
})

# Concatenate the new evaluation scores with the existing DataFrame
eval_scores = pd.concat([eval_scores, new_eval_scores], ignore_index=True)
```

In [11]:
```
# Initialize and fit DBSCAN
dbscan = DBSCAN(eps=0.03, min_samples=2).fit(dog_hist)


new_eval_scores = pd.DataFrame({
    'algorithm': ['dbscan'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, dbscan.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, dbscan.labels_) if -1 in dbscan.labels_ else "Not applicable"]
})

# Concatenate the new evaluation scores with the existing DataFrame
eval_scores = pd.concat([eval_scores, new_eval_scores], ignore_index=True)
```

eps =0.03 and min_samples =2 to generate 4 clusters

In [12]:
```
# Fit Agglomerative Clustering with single linkage
aggsingle = AgglomerativeClustering(n_clusters=4, linkage='single').fit(dog_hist)


new_eval_scores = pd.DataFrame({
    'algorithm': ['single'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, aggsingle.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, aggsingle.labels_)]
})

# Concatenate the new evaluation scores with the existing DataFrame
eval_scores = pd.concat([eval_scores, new_eval_scores], ignore_index=True)
```

In [13]:
```
aggcomplete = AgglomerativeClustering(n_clusters=4, linkage='complete').fit(dog_hist)

# Create a DataFrame for the current evaluation scores
new_eval_scores = pd.DataFrame({
    'algorithm': ['complete'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, aggcomplete.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, aggcomplete.labels_)]
})

# Concatenate the new evaluation scores with the existing DataFrame
eval_scores = pd.concat([eval_scores, new_eval_scores], ignore_index=True)
```

In [14]:
```
aggaverage = AgglomerativeClustering(n_clusters=4, linkage='average').fit(dog_hist)

# Create a DataFrame for the current evaluation scores
new_eval_scores = pd.DataFrame({
    'algorithm': ['average'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, aggaverage.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, aggaverage.labels_)]
})

# Concatenate the new evaluation scores with the existing DataFrame
eval_scores = pd.concat([eval_scores, new_eval_scores], ignore_index=True)
```

In [15]:
```
aggward = AgglomerativeClustering(n_clusters=4, linkage='ward').fit(dog_hist)

# Create a DataFrame for the current evaluation scores
new_eval_scores = pd.DataFrame({
    'algorithm': ['ward'],
    'fowlkes_score': [fowlkes_mallows_score(dog_breeds, aggward.labels_)],
    'silhouette_score': [silhouette_score(dog_hist, aggward.labels_)]
})

# Concatenate the new evaluation scores with the existing DataFrame
eval_scores = pd.concat([eval_scores, new_eval_scores], ignore_index=True)
```

In [16]:
```
eval_scores = eval_scores.drop_duplicates()
```

In [17]:
```
eval_scores
```

Out[17]:

	algorithm	fowlkes_score	silhouette_score
0	random	0.308929	0.395957
1	k-means++	0.310482	0.393838
2	bisecting means	0.323851	0.367088
3	spectral	0.353100	-0.035180
4	dbscan	0.501188	0.708168
5	single	0.499856	0.701080
6	complete	0.407996	0.316655
7	average	0.491786	0.616372
8	ward	0.310739	0.354997

best to worst

In [18]:
```
# fowlkes mallows index
eval_scores.sort_values(by='fowlkes_score', ascending=False)['algorithm']
```

Out[18]:
```
4            dbscan
5            single
7           average
6          complete
3          spectral
2    bisecting means
8              ward
1         k-means++
0            random
Name: algorithm, dtype: object
```

In [19]:
```
# silhouette coefficient
eval_scores.sort_values(by='silhouette_score', ascending=False)['algorithm']
```

Out[19]:
```
4            dbscan
5            single
7           average
0            random
1         k-means++
2    bisecting means
8              ward
6          complete
3          spectral
Name: algorithm, dtype: object
```

reference : https://scikit-learn.org/stable/modules/clustering.html

In [ ]: