```python
#Required Library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage import exposure
import os
import cv2
import numpy as np
from skimage import io, color, feature, exposure
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from sklearn.metrics import davies_bouldin_score
```

```python
# Function to extract HOG features from an image and visualize it
def extract_hog_features(image):
    # Convert the image to grayscale
    gray_image = color.rgb2gray(image)

    # Calculate HOG features
    hog_features, hog_image = feature.hog(gray_image, visualize=True)

    # Enhance the contrast of the HOG image for better visualization
    hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

    return hog_features, hog_image_rescaled
```

```python
# Path to the root folder of your dataset
class_folder = "D:\\collage\\third year\\first semester\\ML\\project
1\\image\\images\\part2"

# List all subdirectories (assuming each subdirectory corresponds to a class)
features_list=[]
labels_list=[]
age = []
gender = []
race = []
```

```
#date_time = []

for image_filename in os.listdir(class_folder):
    image_path = os.path.join(class_folder, image_filename)

    #append age and geder and race for each image
    p = (image_filename.split('.')[0]).split('_')
    age.append(p[0])
    gender.append(p[1])
    race.append(p[2])
    #date_time.append(p[3])

    # Load the image
    image = io.imread(image_path)
    #resize image into same saize 64*128
    image = cv2.resize(image, (64,64))

    # Extract HOG features and visualize
    hog_features, hog_image = extract_hog_features(image)

    # Display the original image and the HOG features
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(2, 2), sharex=True,
sharey=True)

    ax1.axis('off')
    ax1.imshow(image, cmap=plt.cm.gray)

    ax2.axis('off')
    ax2.imshow(hog_image, cmap=plt.cm.gray)

    plt.show()

    # Append HOG features to the features list
    features_list.append(hog_features)

    #not important
    # Append the label to the labels list
    labels_list.append(class_folder)
```

```python
# Convert lists to NumPy arrays
features_array = np.array(features_list)
labels_array = np.array(labels_list)

#not important
# Use LabelEncoder to convert class names into numeric labels
label_encoder = LabelEncoder()
numeric_labels = label_encoder.fit_transform(labels_array)

print("\n********************\n")
print(len(features_array))
```

```python
# Convert 'age' list to numpy array
age_array = np.array(age).reshape(-1, 1)  # Reshape as a column vector

# Convert 'gender' list to numpy array
gender_array = np.array(gender).reshape(-1, 1)  # Reshape as a column vector

# Convert 'race' list to numpy array
race_array = np.array(race).reshape(-1, 1)  # Reshape as a column vector

# Convert 'date_time' list to numpy array
#date_time_array = np.array(date_time).reshape(-1, 1)  # Reshape as a column
vector


# Create the dataset by horizontally stacking age_array with features_array
dataset = np.hstack((age_array,gender_array, race_array,features_array))

# Convert dataset to a pandas DataFrame
column_names = ['Age']+['Gender'] +['Race']  + [f'Feature_{i}' for i in
range(features_array.shape[1])]  # Creating column names
df = pd.DataFrame(dataset, columns=column_names)
```

```python
# save dataset in my drive
df.to_csv('age2.csv', index=False)
```

```python
#load data from my devise
dataset = pd.read_csv('age2.csv')
dataset.head()
```

```
#sum all image features
features_sum = np.sum(features_array, axis=1).tolist()
print(len(features_sum))
print(features_sum[:20])
print("\n-------------------\n")
print(age[:20])
```

```
# new dataset after sum image features
data = {'Age':age ,'Gender':gender ,'Race':race , 'Featurs': features_sum}
df = pd.DataFrame(data)

df.head(2)
```

```
# save dataset after sum all imge features in my drive
df.to_csv('age_after_sum.csv', index=False)
```

```
#load data from my devise
df = pd.read_csv('age_after_sum.csv')
df.head()
```

```
df.shape
```

```
df.info()
```

```
#fill NaN value by Medain
df = df.fillna(df.median())
```

```
df.info()
```

```
# normaliz Age and Features and Gender and race
df['Age'] = MinMaxScaler(feature_range=(0,1)).fit_transform(df[['Age']])
df['Featurs'] = MinMaxScaler(feature_range=(0,1)).fit_transform(df[['Featurs']])
```

```
df.head()
```

```
#spliot into featurs and target
data_x = df.drop(dataset.columns[1], axis=1)  #Featurs all expect gender
data_y = df.iloc[:, 1]   #Target only gender
```

```
# split into Train and Test
x_train, x_test, y_train, y_test =
train_test_split(data_x,data_y,test_size=0.20,random_state=44)
```

```
# K-Means
#useed orginal dataset not need to split dataset
km = KMeans(n_clusters=2, random_state=44)
y_pred = km.fit_predict(df)
```

```
# Calculating the silhouette score
silhouette_avg = silhouette_score(df, km.labels_)
print(f"The Inertia: {km.inertia_}")
print(f"The silhouette score is: {silhouette_avg}")
```

```
#Elbow Method
k=2
inertia = []
for k in range(1, 11):
    kmean = KMeans(n_clusters=k, random_state=42)
    kmean.fit(df)
    inertia.append(kmean.inertia_)

# Plotting the Elbow Method curve
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal k(loss curve)')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Inertia')
plt.xticks(range(1, 11))
plt.show()
```

```
# Visualize the cluster centers (representative images)
k=2
fig, ax = plt.subplots(1, k, figsize=(15, 3))
for i in range(k):
    center_image = km.cluster_centers_[i].reshape(2,2)  # Reshape to original
dimensions
    ax[i].imshow(center_image, cmap='gray')
    ax[i].axis('off')
    ax[i].set_title(f'Cluster {i}')

plt.show()
```