AI Capstone Project#1

Research question:

2020 年 11 月有則新聞報導:AI 攝影機誤把光頭裁判當成足球追蹤。因此我想觀察機器學習分辨足球與光頭的效果。

Documentation of dataset:

數據類型: jpg

數量: 足球照片 100 張，光頭照片 100 張

蒐集方式: 在 Unsplash 和 pexel 等免費圖庫網站蒐集有足球和有光頭的照片，再裁剪出足球跟光頭。用 python 把照片調整成相同大小(100*100)。最後製作一個 csv 檔，列出圖片檔案路徑和標籤，學習時會依據圖片的值讀出檔案並標上相對應的標籤，足球標上 0，光頭標上 1。


Method

Supervised method:

1. CNN: 一種深度學習模型，主要是利用 Convolutional Layer 和 Pooling Layer 學習圖像的特徵，並不斷調整模型內部的權重，進而實現圖像的分類。

   訓練流程：首先將資料取出，把圖片轉成 numpy array 作為 X，label 作為 y。接著 cross-validation，設成 3 folds。接著建立 CNN 模型，總共使用兩個 Conv2D、兩個 MaxPooling2D、一個 Flatten 加上一個用 relu 活化的 Dense，最後再用一個 Dense 當作輸出層，有兩個神經元，用 softmax 活化，以取得每項類別的概率。模型編譯時，使用 Adam optimizer，categorical_crossentropy 作為損失函數。再來拿模型跟分類好的資料進行訓練，並且在每層 fold 進行評估。

2. SVM: 透過尋找最佳的超平面，將不同類別的數據點分開，同時最大化類別之間的距離。

   訓練流程：將資料取出，把圖片進行 preprocessing，並轉成 numpy array

作為 X，label 作為 y。接著設定 cross-validation，設定成 3 folds。接著開始 cross-validation，用 HOG 提取特徵，接著進行 PCA 降低維度維，設 components =2 以便畫圖表分析。再來選擇 linear 的 SVM 進行分類，準備好後開始模型訓練，訓練後評估 model 的表現。

Unsupervised method:

K-Means clustering: 將數據分成 k 個 cluster。首先隨機設 k 個中心點，透過不斷計算平均，更新平均點，直到找到最佳的中心點。
訓練流程：首先將圖片數據轉成一維的 numpy array。用一個 array 儲存圖片的 numpy array，並且用 HOG 提取特徵。同時提取 label，label 是在 evaluation 時會用到，進行 cluster 時不會使用。接著用 PCA 降低維度，取 components = 2。拿 PCA 的結果進行 k-means cluster，由於數據分成兩種類別，所以分成兩個 clusters。分析時，比較數據原本的 label 和後來被分配到的 cluster，以及畫出圖表用 PCA 的兩個 component 作為座標，畫出數據點的分布。

Feature extractor& Dimensionality reduction

1. PCA: 可以降低數據的維度，並同時保留原始數據的變異性。它的運作過程是將數據進行標準化，計算數據 covariance matrix 並進行分解，選取前 k 個特徵值對應的特徵向量作為 principal components，將數據投影到由選取的 components 所構成的子空間中，而得到低維度的特徵。

2. HOG: 異種表示特徵的方法，主要用於圖像識別和目標檢測。它的運作過程是將圖像局部區域的梯度方向轉換為特徵向量，進而描述圖像中的紋理和形狀。

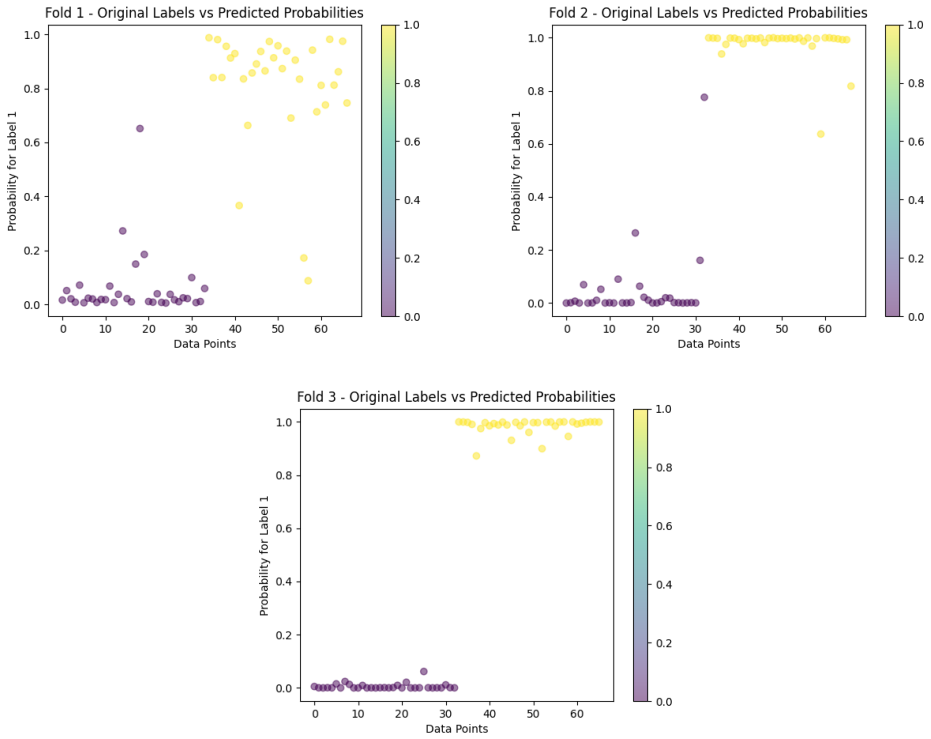
由於 CNN 的 Convolutional Layer 會自動學習特徵的表現方式，可以從原始圖像提取特徵，所以不需要另外使用 feature extractor。而 SVM 和 K-means cluster 在訓練的過程不會提取特徵。因此在這個 project 中，只有 SVM 和 cluster 使用 PCA 和 HOG，CNN 沒有。

Result evaluation

CNN

Fold	Accuracy	Precision	Recall	F1 Score	AUROC	Confusion Matrix
1	0.9402	0.9677	0.9090	0.9375	0.9919.	[[33 1] [3 30]]
2	0.98507	0.9714	1.0000	0.9855	0.9991	[[32 1] [0 34]]

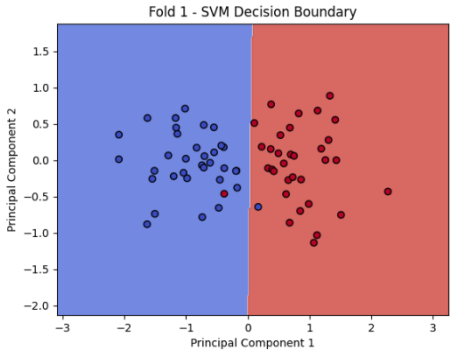
3	1.0000	1.0000	1.0000	1.0000	1.0000	[[33 0] [0 33]]

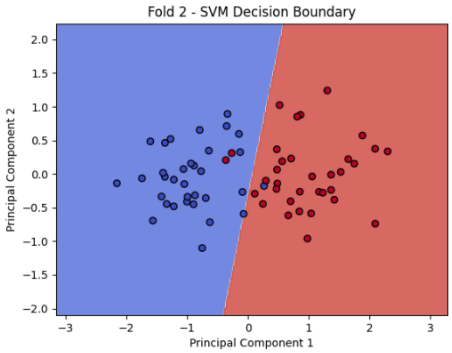


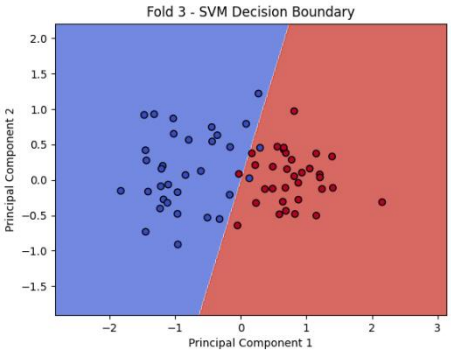
上列三個圖表是三個 folds 各自的結果。橫軸代表 data point，縱軸代表 data point 預測結果是 1 的機率，紫色點是 label 0 的 data point，黃色點是 label1。可以看到越後面的 fold，準確率提升，且 data point 被判斷成 1 的機率往 0 和 1 靠近。CNN 是 deep learning model，可以從數據中學習 feature，訓練多次後能夠發現更高層次的抽象特徵，模型會逐漸調整權重以擬合數據，因此可以看到分類的結果隨著訓練次數變多，分類的結果越好。

SVM

Fold	Accuracy	Precision	Recall	F1 Score	AUROC	Confusion Matrix
1	0.9701	0.9696	0.9696	0.9696	0.9955	[[33 1] [1 32]]
2	0.9552	0.9696	0.9411	0.9552	0.9893	[[32 1] [2 32]]
3	0.9545	0.9411	0.9696	0.9552	0.9963	[[31 2] [1 32]]







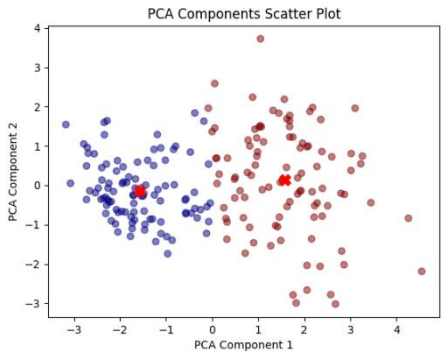
上列的三個圖表的橫軸和縱軸是 PCA 提取的兩個 components，中間的斜線是根據 SVM 學習後找到的邊界線。由於 SVM 跟 CNN 不同，不是 deep learning，所以分類的表現不會像 CNN 越後面的 fold 準確度越高。

Cluster:

Cluster / Label	0	1
0	95	5
1	5	95

Label 0 被分配到 cluster 0，label1 被分配到 cluster1

Accuracy: 0.95



可以從 clustering 的結果看到，label 0 幾乎都被分配到 cluster 0，label 1 幾乎都被分配到 cluster1。上面的圖表是以兩個 PCA component 做為橫軸跟縱軸，藍點

是 cluster 0，紅點是 cluster 1，兩個 cluster 中間的 x 是 cluster center。可以看到 cluster 0 比 cluster 1 更聚集，可能是因為足球之間的特徵比起光頭人像的特徵較為接近。整體來說，kmeans cluster 的分類結果能大致符合相對應的 label。

Experiment:

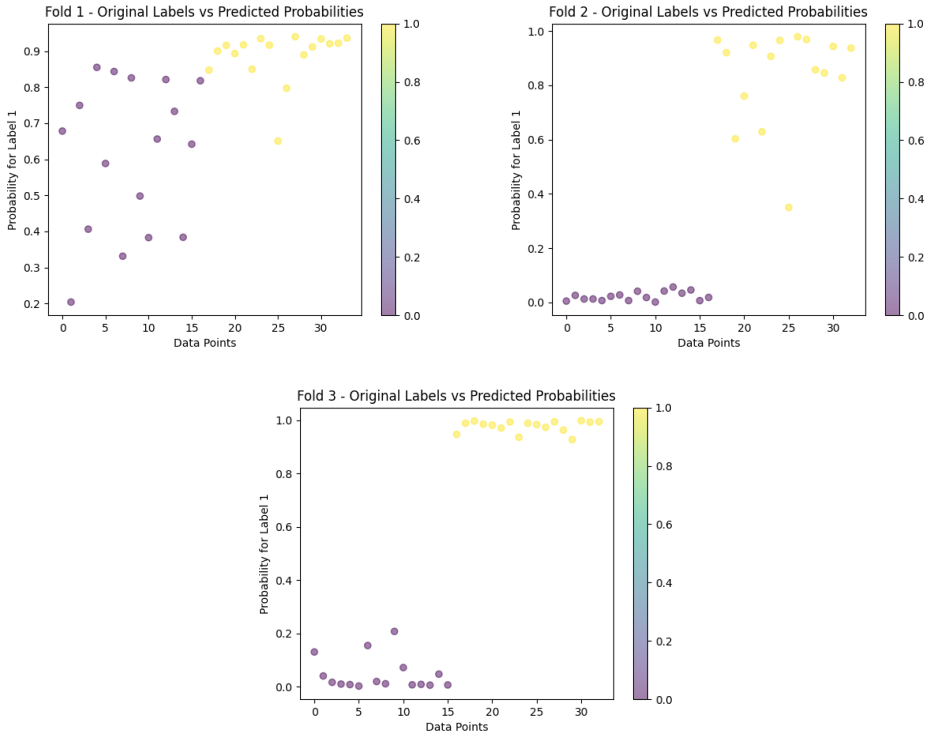
總共有三個 experiments：

1. 減少數據，從兩個類別各 100 張減少到 50 張
2. SVM 和 clustering 不使用 PCA
3. SVM 和 clustering 不使用 HOG

- Experiment 1. 減少數據，從兩個類別各 100 張減少到 50 張

CNN:

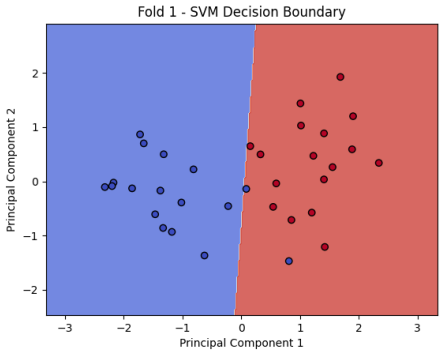
Fold	Accuracy	Precision	Recall	F1 Score	AUROC	Confusion Matrix
1	0.6764	0.6071	1.0000	0.7555	0.9446	[[6 11] [0 17]]
2	0.9696	1.0000	0.9375	0.9677	1.0000	[[17 0] [1 15]]
3	1.0000	1.0000	1.0000	1.0000	1.0000	[[16 0] [0 17]]

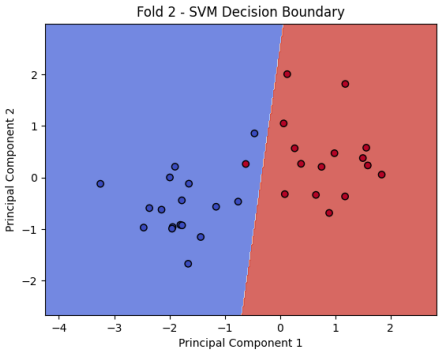


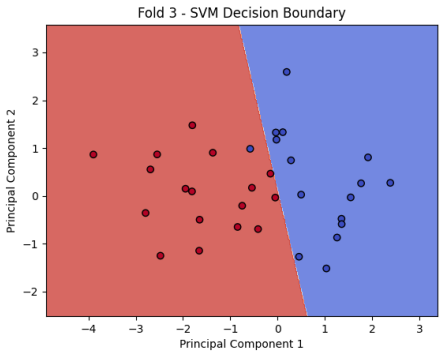
照片的數量減少後，fold 1 的表現有明顯變差(原本的 accurancy 是 0.9402)，但隨著訓練次數增加，fold 2 和 fold 3 的表現跟訓練 200 張照片時的表現差不多。fold 1 表現不好可能是因為數據太少而沒辦法學習足夠的特徵。

SVM:

Fold	Accuracy	Precision	Recall	F1 Score	AUROC	Confusion Matrix
1	0.9411	0.8947	1.0000	0.9444	0.9826	[[15 2] [0 17]]
2	0.9696	1.0	0.9375	0.9677	0.9963	[[17 0] [1 15]]
3	0.9696	0.9444	1.0	0.9714	0.9926	[[15 1] [0 17]]





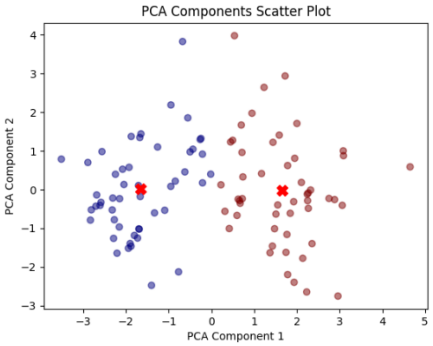


減少數據後的分類表現和減少前只有略為的差距(原本三次的 accuracy: 0.9701, 0.9552, 0.9545)，推測這樣的數據變化幅度對於 SVM 方法不會有太大的影響。

Cluster:

Cluster / Label	0	1
0	48	2
1	2	48

Accuracy: 0.96 (Label 0 被分配到 cluster 0，label1 被分配到 cluster1)



藍點是 cluster 0，紅點是 cluster 1。準確度和減少數據前的表現(0.95)差不多，比對 PCA component 的圖表，可以發現此時的 cluster 0 和 cluster 1 的集中程度相近，可能是在這個數據下足球之間的特徵接近程度和光頭的特徵之間的接近程度差不多。

- Experiment 2. SVM 和 cluster 不使用 PCA

(由於沒有使用 PCA，所以就沒有 PCA components 的圖表)

SVM:

Fold	Accuracy	Precision	Recall	F1 Score	AUROC	Confusion Matrix
1	1.0000	1.0000	1.0000	1.0000	1.0000	[[34 0] [0 33]]
2	0.9701	0.9444	1.0000	0.9714	0.9928	[[31 2] [1 32]]
3	0.9696	0.9696	0.9696	0.9696	0.9972	[[32 1] [1 32]]

整體的表現只有略微的差異，推測是否有使用 PCA 對 SVM 方法的影響不大。

Cluster:

Cluster / Label	0	1
0	7	93
1	1	99

Accuracy: 0.53(設正確分類為 label 0 分配到 cluster 0，label1 分配到 cluster1)

沒有用 PCA 分類時，準確度大幅下降(原本 0.95)。從實驗結果可以推測，k means cluster 在沒有用 PCA 降低維度的狀況下，可能受到其他 features 干擾，使得大多資料被分類到同一 cluster。有此可知，這 Project 中，若要使用 cluster，PCA 是一項相當重要的步驟。

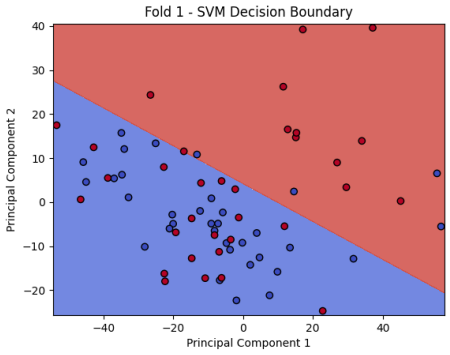
SVM 的結果沒有太大的差異，而 cluster 的結果有明顯的差異，可能是因為 SVM 能夠利用訓練數據中的標籤信息，更好地理解特徵之間的關係，並嘗

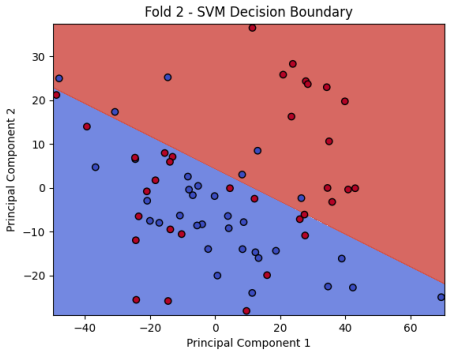
試構建一個能夠有效區分不同類別的邊界。而 cluster 沒有事先 label，不知道數據的結構或標籤，且它僅通過數據點之間的相似性來劃分數據，使得在沒有 PCA 降低維度的狀況下受到其他特徵干擾，而做出與預期不同的分類。

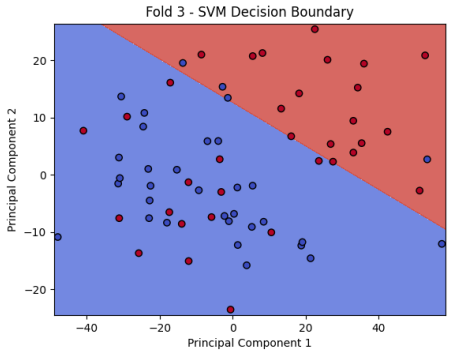
- Experiment 3. SVM 和 cluster 不使用 HOG:

SVM:

Fold	Accuracy	Precision	Recall	F1 Score	AUROC	Confusion Matrix
1	0.6268	0.7500	0.3636	0.4897	0.6114	[[30 4] [21 12]]
2	0.5970	0.6842	0.3823	0.4905	0.6737	[[27 6] [21 13]]
3	0.7121	0.8181	0.5454	0.6545	0.7079	[[29 4] [15 18]]





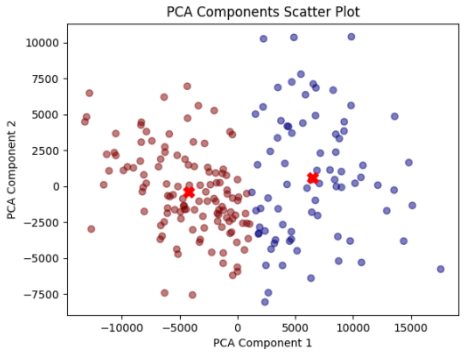


在沒有使用 HOG 的狀況下，SVM 方法的表現明顯變差(有使用時準確率都會達到 0.9 以上)。可以發現 HOG 提取特徵是一項相當關鍵的動作。

Cluster:

Cluster / Label	0	1
0	29	71
1	50	50

Accuracy: 0.605(label 0 分配到 cluster 1，label1 被分配到 cluster 0)



藍點是 cluster 0，紅點是 cluster 1。可以看到若沒有使用 HOG 的狀況下，cluster 方法的表現明顯變差，甚至 label 1 被平均分配到兩個 cluster。
SVM 和 cluster 在沒有 HOG 提取特徵的狀況下表現都明顯變差了。HOG 特徵通常用於捕捉圖像中的邊緣和紋理等局部特徵。如果沒有使用 HOG，可能使用的特徵無法充分捕捉圖像的結構和重要細節，從而導致特徵不具代表性。

Discussion:

在這個 project 中，我認為影響表現的因素主要可以分為數據和模型。數據因素包含數據的數量，圖片大小，數據的平衡性，還有提取數據特徵的方式。模型因素包含演算法，模型的結構，以及訓練的次數以及參數等等。而在這次 project 有做改變數據數量和提取特徵的方式，皆會對結果產生改變。
觀察實驗的結果，減少數據的實驗中，我預期表現會變差，因為資料變少比較難學到判斷的關鍵，CNN 表現如預期，SVM 和 cluster 沒有什麼變化在預料之外。
在改變提取特徵方式的實驗中，我預期有用 PCA 或 HOG 的表現會比較好，因為 PCA 降低維度可以去除冗餘訊息的干擾，而 HOG 對圖像的局部特徵有良好的辨別能力，但有這麼大的變化是在預料之外，這也讓我了解處理資料方式的重要性。
從這些實驗中，我學到數據數量和提取特徵方式對不同的學習方式有不同的影響。如果有更多的時間，我希望能蒐集更多資料。由於部分實驗結果前後沒有太大的變化，說不定數據規模大一些就可以看得出變化了。另外，我還想嘗試改變資料的平衡性或作 data augmentation 等實驗，了解這些變因對結果的影響。

Reference:

Unsplash: https://unsplash.com/
Pexel: https://www.pexels.com/zh-tw/
CNN: https://www.tensorflow.org/tutorials/images/cnn

SVM: https://scikit-learn.org/stable/modules/svm.html
Cluster (k mean):https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html
HOG: https://scikit-image.org/docs/stable/auto_examples/features_detection/plot_hog.html
PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

Code

CNN:

```
import os
import pandas as pd
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score,precision_score,
recall_score, f1_score, roc_auc_score
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt

# Load the dataset
dataset = pd.read_csv("dataset.csv")

# Image preprocessing function
def preprocess_image(image_path):
    img = load_img(image_path, target_size=(100, 100))
    img_array = img_to_array(img) / 255.0  # Normalize pixel values
    return img_array

# Preprocess images and convert labels to one-hot encoding
X = np.array([preprocess_image(path) for path in
dataset['Image_Path']])
y = to_categorical(dataset['Label'])

# Initialize StratifiedKFold
stratified_kfold = StratifiedKFold(n_splits=3, shuffle=True,
random_state=21)

# Define the CNN model
model = Sequential()
model.add(Conv2D(8, (3, 3), activation='relu', input_shape=(100, 100,
3)))
```

```
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(2, activation='softmax'))


# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])


# Perform cross-validation
fold_number = 1
for train_index, val_index in stratified_kfold.split(X, np.argmax(y,
axis=1)):
    X_train, X_val = X[train_index], X[val_index]
    y_train, y_val = y[train_index], y[val_index]


    # Train the model
    model.fit(X_train, y_train, epochs=5, validation_data=(X_val,
y_val))

    # Evaluate the model on the validation set
    y_pred_prob = model.predict(X_val)
    y_pred = np.argmax(y_pred_prob, axis=1)


    # Plot points with different colors for original labels
    plt.scatter(np.arange(len(y_val)), y_pred_prob[:, 1], c=y_val[:, 1],
cmap='viridis', alpha=0.5)
    plt.xlabel('Data Points')
    plt.ylabel('Probability for Label 1')
    plt.title(f'Fold {fold_number} - Original Labels vs Predicted
Probabilities')
    plt.colorbar()
```

```
    plt.savefig(f'image/CNN-Fold {fold_number}.png')
    plt.show()


    # Calculate evaluation metrics
    accuracy = accuracy_score(np.argmax(y_val, axis=1), y_pred)
    precision = precision_score(np.argmax(y_val, axis=1), y_pred)
    recall = recall_score(np.argmax(y_val, axis=1), y_pred)
    f1 = f1_score(np.argmax(y_val, axis=1), y_pred)
    auroc = roc_auc_score(y_val, y_pred_prob,multi_class='ovr')
    conf_matrix = confusion_matrix(np.argmax(y_val, axis=1), y_pred)

    print(f"Fold {fold_number} Metrics:")
    print(f"  Accuracy: {accuracy}")
    print(f"  Precision: {precision}")
    print(f"  Recall: {recall}")
    print(f"  F1 Score: {f1}")
    print(f"  AUROC: {auroc}")
    print(f"Fold {fold_number} Confusion Matrix:")
    print(conf_matrix)


    fold_number += 1
```

SVM:

```
import os
import pandas as pd
from sklearn.model_selection import StratifiedKFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score,confusion_matrix
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from skimage.feature import hog
from skimage.color import rgb2gray
```

```
# Load the dataset
dataset = pd.read_csv("dataset.csv")

# Image preprocessing function
def preprocess_image(image_path):
    img = load_img(image_path, target_size=(100, 100))
    img_array = img_to_array(img) / 255.0  # Normalize pixel values
    return img_array

# Preprocess images
X = np.array([preprocess_image(path) for path in
dataset['Image_Path']])
y = dataset['Label']

# Encode labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Initialize StratifiedKFold
stratified_kfold = StratifiedKFold(n_splits=3, shuffle=True,
random_state=21)

# Perform cross-validation
fold_number = 1
for train_index, val_index in stratified_kfold.split(X, y):
    X_train, X_val = X[train_index], X[val_index]
    y_train, y_val = y[train_index], y[val_index]

    X_train_gray = np.array([rgb2gray(img) for img in X_train])
    X_val_gray = np.array([rgb2gray(img) for img in X_val])

    X_train_hog = np.array([hog(img, orientations=9, pixels_per_cell=(8,
8), cells_per_block=(2, 2)) for img in X_train_gray])
    X_val_hog = np.array([hog(img, orientations=9, pixels_per_cell=(8,
8), cells_per_block=(2, 2)) for img in X_val_gray])

    # Apply PCA to reduce dimensionality to 2D
```

```
    pca = PCA(n_components=2)
    X_train_pca = pca.fit_transform(X_train_hog)
    X_val_pca = pca.transform(X_val_hog)

    # Define the SVM model
    svm_model = SVC(kernel='linear')



    # Train the SVM model
    svm_model.fit(X_train_pca, y_train)

    # Make predictions on the validation set
    y_pred = svm_model.predict(X_val_pca)



    accuracy = accuracy_score(y_val, y_pred)
    print(f"Fold {fold_number} Accuracy: {accuracy}")

    # Evaluate additional metrics
    precision = precision_score(y_val, y_pred)
    recall = recall_score(y_val, y_pred)
    f1 = f1_score(y_val, y_pred)
    auroc = roc_auc_score(y_val, svm_model.decision_function(X_val_pca))
    conf_matrix = confusion_matrix(y_val, y_pred)

    print(f"Fold {fold_number} Precision: {precision}")
    print(f"Fold {fold_number} Recall: {recall}")
    print(f"Fold {fold_number} F1 Score: {f1}")
    print(f"Fold {fold_number} AUROC: {auroc}")
    print(f"Fold {fold_number} Confusion Matrix:")
    print(conf_matrix)

    # Find the minimum and maximum values of the pac component to ensure
the boundary
    h = .02  # step size in the mesh
    x_min, x_max = X_val_pca[:, 0].min() - 1, X_val_pca[:, 0].max() + 1
    y_min, y_max = X_val_pca[:, 1].min() - 1, X_val_pca[:, 1].max() + 1
```

```
    xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min,
y_max, h))
    Z = svm_model.predict(np.c_[xx.ravel(), yy.ravel()])

    # Put the result into a color plot
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)

    # Plot data points
    plt.scatter(X_val_pca[:, 0], X_val_pca[:, 1], c=y_val, cmap =
  plt.cm.coolwarm, edgecolors='k', marker='o')

    # Set plot labels
    plt.title(f"Fold {fold_number} - SVM Decision Boundary")
    plt.xlabel("Principal Component 1")
    plt.ylabel("Principal Component 2")
    plt.savefig(f'image/SVM Fold{fold_number}.png')
    plt.show()

    fold_number += 1
```

cluster:

```
import os
import numpy as np
from PIL import Image
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd
import cv2
from skimage.feature import hog
from skimage import exposure

def extract_features(image_path, resize_to=(100, 100)):
    """
    Extract features from an image.
```

```
    Parameters:
    - image_path: Path to the image file.
    - resize_to: New size after resizing, represented as a tuple (width,
height).

    Returns:
    - 1D numpy array representing the features.
    """
    try:
        with Image.open(image_path) as img:
            img = img.resize(resize_to)

            img_array = np.array(img).flatten()



        img = cv2.imread(image_path)
        img = cv2.resize(img, resize_to)
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        fd, hog_img = hog(gray_img, orientations=9, pixels_per_cell=(8,
8), cells_per_block=(2, 2), visualize=True)

        hog_img_rescaled = exposure.rescale_intensity(hog_img,
in_range=(0, 10))
        hog_features = fd.flatten()

        combined_features = np.concatenate((img_array, hog_features))

        return hog_features
    except Exception as e:
        print(f"Error occurred while extracting features: {e}")
        return None

def cluster_images(csv_path, num_clusters=2):
    """
    Cluster images.

    Parameters:
```

```
    - csv_path: Path to the CSV file containing image paths and labels.
    - num_clusters: Number of clusters.

    Returns:
    - A dictionary containing the path of each image and its
corresponding cluster label.
    """
    # Read the CSV file
    dataset = pd.read_csv(csv_path)

    features = []
    image_paths = []
    labels = []

    # Iterate over all images in the dataset
    for index, row in dataset.iterrows():
        image_path = row['Image_Path']
        label = row['Label']



        # Check if the file is an image
        if os.path.isfile(image_path) and
image_path.lower().endswith(('.png', '.jpg', '.jpeg')):
            feature = extract_features(image_path)

            # Ensure successful feature extraction
            if feature is not None:
                features.append(feature)
                image_paths.append(image_path)
                labels.append(label)

    # Convert features to a NumPy array
    features_array = np.array(features)


    # Perform feature transformation using PCA
    num_pca_components=2
    pca = PCA(n_components=num_pca_components)
    pca_result = pca.fit_transform(features_array)
```

```
    kmeans = KMeans(n_clusters=num_clusters, random_state=21)  # Remove
'random_state' for randomness
    kmeans.fit(pca_result)
    centers = kmeans.cluster_centers_

    clusters = kmeans.predict(pca_result)

    # Associate each image with its cluster label, label, and PCA
components
    image_cluster_label_mapping = {'Image_Path': image_paths, 'Cluster':
clusters, 'Label': labels,
            'PCA_Component_1': pca_result[:, 0], 'PCA_Component_2':
pca_result[:, 1]}

    return image_cluster_label_mapping,centers


def visualize_clusters(image_cluster_mapping,centers):
    """
    Visualize clustering results.

    Parameters:
    - image_cluster_mapping: A dictionary containing the path of each
image and its corresponding cluster label.
    """
    # Convert the dictionary to separate lists
    image_paths = image_cluster_mapping['Image_Path']
    clusters = image_cluster_mapping['Cluster']
    labels=image_cluster_mapping['Label']

    # Visualize the clustering results
    df = pd.DataFrame(image_cluster_mapping)

    # Calculate the frequency of different labels assigned to different
clusters
    count_df = df.groupby( ['Label',
'Cluster'] ).size().reset_index(name='Count')
```

```
    print("Label Clustering Statistics:")
    print(count_df)



    # Plot the chart of PCA components
    pca_component_1 = image_cluster_mapping['PCA_Component_1']
    pca_component_2 = image_cluster_mapping['PCA_Component_2']

    # Create colors for different clusters
    unique_clusters = set(clusters)
    colors = [plt.cm.jet(cluster / float(len(unique_clusters) - 1)) for
cluster in clusters]

    # Plot the scatter plot
    plt.scatter(pca_component_1, pca_component_2, c=colors, marker='o',
alpha=0.5)
    plt.scatter(centers[:, 0], centers[:, 1], c='red', marker='X',
s=100, label='Cluster Centers')

    plt.xlabel('PCA Component 1')
    plt.ylabel('PCA Component 2')
    plt.title('PCA Components Scatter Plot')
    plt.savefig('image/PCA-scatter.png')
    plt.show()


csv_file_path = "dataset.csv"
num_clusters = 2


# Perform clustering
image_cluster_mapping,centers = cluster_images(csv_file_path,
num_clusters)

# Visualize clustering results
visualize_clusters(image_cluster_mapping,centers)
```