```
import os
import tarfile
from pathlib import Path

# Upload lfw-funneled.tgz manually to Colab or use drive
dataset_path = "/content/lfw-funneled.tgz"
extract_path = "/content/lfw_dataset"

# Extract the dataset
if not os.path.exists(extract_path):
    with tarfile.open(dataset_path, "r:gz") as tar:
        tar.extractall(path=extract_path)

print(f"Dataset extracted to: {extract_path}")
```

```
Dataset extracted to: /content/lfw_dataset
```

```
def load_images(image_dir, image_size=(160, 160)):
    X, y = [], []
    full_dir = Path(image_dir, "lfw_funneled")
    print(f"Checking directory: {full_dir}")

    for person_dir in full_dir.iterdir():
        if person_dir.is_dir():
            label = person_dir.name
            for img_path in person_dir.glob("*.jpg"):
                img = cv2.imread(str(img_path))
                if img is None:
                    continue
                img = cv2.resize(img, image_size)
                X.append(img_to_array(img))
                y.append(label)
    return np.array(X), np.array(y)
```

```python
!pip install scikit-image opencv-python
import os
import tarfile
from pathlib import Path
import cv2
import numpy as np
from skimage.feature import local_binary_pattern, hog
from keras.utils import img_to_array
from sklearn.model_selection import train_test_split

# Upload lfw-funneled.tgz manually to Colab or use drive
dataset_path = "/content/lfw-funneled.tgz"
extract_path = "/content/lfw_dataset"

# Extract the dataset
if not os.path.exists(extract_path):
    with tarfile.open(dataset_path, "r:gz") as tar:
        tar.extractall(path=extract_path)

print(f"Dataset extracted to: {extract_path}")

def load_images(image_dir, image_size=(160, 160)):
    X, y = [], []
    full_dir = Path(image_dir, "lfw_funneled")
    print(f"Checking directory: {full_dir}")

    for person_dir in full_dir.iterdir():
        if person_dir.is_dir():
            label = person_dir.name
            for img_path in person_dir.glob("*.jpg"):
                img = cv2.imread(str(img_path))
                if img is None:
                    continue
                img = cv2.resize(img, image_size)
                X.append(img_to_array(img))
                y.append(label)
    return np.array(X), np.array(y)

# Load the dataset
X, y = load_images(extract_path)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


def extract_lbp(image):
    gray = cv2.cvtColor(image.astype('uint8'), cv2.COLOR_RGB2GRAY)
    lbp = local_binary_pattern(gray, P=8, R=1.0, method='uniform')
    return lbp.flatten()

def extract_hog(image):
    gray = cv2.cvtColor(image.astype('uint8'), cv2.COLOR_RGB2GRAY)
    hog_features, _ = hog(gray, pixels_per_cell=(8, 8), cells_per_block=(2, 2), visualize=True)
    return hog_features

def fuse_features(images):
    features = []
    for img in images:
        hog_f = extract_hog(img)
        lbp_f = extract_lbp(img)
        fused = np.concatenate([hog_f, lbp_f])
        features.append(fused)
    return np.array(features)

# Extract features and fuse them
X_train_fused = fuse_features(X_train)
X_test_fused = fuse_features(X_test)
```

```
Requirement already satisfied: scikit-image in /usr/local/lib/python3.11/dist-packages (0.25.2)
Requirement already satisfied: opencv-python in /usr/local/lib/python3.11/dist-packages (4.11.0.86)
Requirement already satisfied: numpy>=1.24 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (2.0.2)
Requirement already satisfied: scipy>=1.11.4 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (1.14.1)
Requirement already satisfied: networkx>=3.0 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (3.4.2)
Requirement already satisfied: pillow>=10.1 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (11.1.0)
Requirement already satisfied: imageio!=2.35.0,>=2.33 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (2025.3.30)
Requirement already satisfied: packaging>=21 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (24.2)
Requirement already satisfied: lazy-loader>=0.4 in /usr/local/lib/python3.11/dist-packages (from scikit-image) (0.4)
```

```
Dataset extracted to: /content/lfw_dataset
Checking directory: /content/lfw_dataset/lfw_funneled
```

```
!pip install keras-vggface keras-applications --upgrade
```

```
Requirement already satisfied: keras-vggface in /usr/local/lib/python3.11/dist-packages (0.6)
Requirement already satisfied: keras-applications in /usr/local/lib/python3.11/dist-packages (1.0.8)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (2.0.2)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (1.14.1)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (3.13.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (11.1.0)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (3.8.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (1.17.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (6.0.2)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras->keras-vggface) (1.4.0)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras->keras-vggface) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras->keras-vggface) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras->keras-vggface) (0.14.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras->keras-vggface) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras->keras-vggface) (24.2)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from optree->keras->keras-vggface) (
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->keras-vggface) (3.0.0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->keras-vggface) (2.1
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras->keras-vgg
```

```
!pip install keras==2.11.0 #Downgrade keras to a compatible version
!pip install keras_vggface --no-deps #Install keras_vggface without automatically installing dependencies
```

```
Collecting keras==2.11.0
  Downloading keras-2.11.0-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 17.9 MB/s eta 0:00:00
Installing collected packages: keras
  Attempting uninstall: keras
    Found existing installation: keras 3.8.0
    Uninstalling keras-3.8.0:
      Successfully uninstalled keras-3.8.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
tensorflow 2.18.0 requires keras>=3.5.0, but you have keras 2.11.0 which is incompatible.
Successfully installed keras-2.11.0
Requirement already satisfied: keras_vggface in /usr/local/lib/python3.11/dist-packages (0.6)
```

```
!pip install keras==2.11.0 #Downgrade keras to a compatible version
!pip install keras_vggface --no-deps #Install keras_vggface without automatically installing dependencies
```

```
Requirement already satisfied: keras==2.11.0 in /usr/local/lib/python3.11/dist-packages (2.11.0)
Requirement already satisfied: keras_vggface in /usr/local/lib/python3.11/dist-packages (0.6)
```

```
!pip install tensorflow==2.11.0
!pip install keras==2.11.0
!pip install keras_vggface
```

```
ERROR: Could not find a version that satisfies the requirement tensorflow==2.11.0 (from versions: 2.12.0rc0, 2.12.0rc1, 2.12.0, 2.12.1,
ERROR: No matching distribution found for tensorflow==2.11.0
Requirement already satisfied: keras==2.11.0 in /usr/local/lib/python3.11/dist-packages (2.11.0)
Requirement already satisfied: keras_vggface in /usr/local/lib/python3.11/dist-packages (0.6)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (2.0.2)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (1.14.1)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (3.13.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (11.1.0)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (2.11.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (1.17.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from keras_vggface) (6.0.2)
```

```
!pip uninstall -y keras keras-nightly keras-Preprocessing keras-vis
!pip install keras==2.11
!pip install keras-vggface keras-applications
import os
os.kill(os.getpid(), 9)  # Restart runtime
```

```
Found existing installation: keras 2.11.0
Uninstalling keras-2.11.0:
  Successfully uninstalled keras-2.11.0
WARNING: Skipping keras-nightly as it is not installed.
```

```
WARNING: Skipping keras-Preprocessing as it is not installed.
WARNING: Skipping keras-vis as it is not installed.
Collecting keras==2.11
  Using cached keras-2.11.0-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
Installing collected packages: keras
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
tensorflow 2.18.0 requires keras>=3.5.0, but you have keras 2.11.0 which is incompatible.
Successfully installed keras-2.11.0
Requirement already satisfied: keras-vggface in /usr/local/lib/python3.11/dist-packages (0.6)
Requirement already satisfied: keras-applications in /usr/local/lib/python3.11/dist-packages (1.0.8)
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (2.0.2)
Requirement already satisfied: scipy>=0.14 in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (1.14.1)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (3.13.0)
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (11.1.0)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (2.11.0)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (1.17.0)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.11/dist-packages (from keras-vggface) (6.0.2)
```

```
!pip install facenet-pytorch
```

```
Requirement already satisfied: facenet-pytorch in /usr/local/lib/python3.11/dist-packages (2.6.0)
Requirement already satisfied: numpy<2.0.0,>=1.24.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (1.26.4)
Requirement already satisfied: Pillow<10.3.0,>=10.2.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (10.2.0)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (2.32.3)
Requirement already satisfied: torch<2.3.0,>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (2.2.2)
Requirement already satisfied: torchvision<0.18.0,>=0.17.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (0.17.2)
Requirement already satisfied: tqdm<5.0.0,>=4.0.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (4.67.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->facenet
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->facenet-pytorch) (3
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->facenet-pytor
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->facenet-pytor
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-pytorch) (3.18.0)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-py
Requirement already satisfied: sympy in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-pytorch) (1.13.1)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-pytorch) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-pytorch) (3.1.6)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-pytorch) (2025.3.2)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->fa
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->fa
Requirement already satisfied: nvidia-cudnn-cu12==8.9.2.26 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet
Requirement already satisfied: nvidia-cublas-cu12==12.1.3.1 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facene
Requirement already satisfied: nvidia-cufft-cu12==11.0.2.54 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facene
Requirement already satisfied: nvidia-curand-cu12==10.3.2.106 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->face
Requirement already satisfied: nvidia-cusolver-cu12==11.4.5.107 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->fa
Requirement already satisfied: nvidia-cusparse-cu12==12.1.0.106 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->fa
Requirement already satisfied: nvidia-nccl-cu12==2.19.3 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-py
Requirement already satisfied: nvidia-nvtx-cu12==12.1.105 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.11/dist-packages (from torch<2.3.0,>=2.2.0->facenet-pytorch) (2.2
Requirement already satisfied: nvidia-nvjitlink-cu12 in /usr/local/lib/python3.11/dist-packages (from nvidia-cusolver-cu12==11.4.5.107->
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/dist-packages (from jinja2->torch<2.3.0,>=2.2.0->facenet-pyt
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from sympy->torch<2.3.0,>=2.2.0->facenet-p
from facenet_pytorch import InceptionResnetV1
import torch
from torchvision import transforms
from PIL import Image

# Load pretrained FaceNet
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
facenet = InceptionResnetV1(pretrained='vggface2').eval().to(device)

# Preprocessing for FaceNet
transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((160, 160)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5]*3, std=[0.5]*3)
])

def get_facenet_embeddings(images):
    embeddings = []
    for img in images:
        img_t = transform(img.astype(np.uint8)).unsqueeze(0).to(device)
        with torch.no_grad():
            embedding = facenet(img_t).cpu().numpy().flatten()
        embeddings.append(embedding)
    return np.array(embeddings)
```

```
100%                                        107M/107M [00:00<00:00, 144MB/s]
```

```
from collections import Counter

# Count each label's occurrences
label_counts = Counter(y)

# Keep only labels with at least 2 images
valid_labels = {label for label, count in label_counts.items() if count >= 2}

# Filter images and labels
X_filtered = []
y_filtered = []
for img, label in zip(X, y):
    if label in valid_labels:
        X_filtered.append(img)
        y_filtered.append(label)

X_filtered = np.array(X_filtered)
y_filtered = np.array(y_filtered)

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y_filtered)

# Train/test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    X_filtered, y_encoded, test_size=0.2, stratify=y_encoded, random_state=42
)

print(f"Filtered dataset: {len(X_filtered)} images from {len(set(y_filtered))} people.")
```

```
Filtered dataset: 677 images from 39 people.
```

```python
import os
import cv2
import numpy as np
from pathlib import Path
from collections import defaultdict, Counter
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

def load_images_filtered(image_dir, image_size=(160, 160), min_images_per_class=2):
    X, y = [], []
    full_dir = Path(image_dir, "lfw_funneled")

    label_image_map = defaultdict(list)

    # First collect all images and group by label
    for person_dir in full_dir.iterdir():
        if person_dir.is_dir():
            label = person_dir.name
            images = list(person_dir.glob("*.jpg"))
            if len(images) >= min_images_per_class:
                for img_path in images:
                    img = cv2.imread(str(img_path))
                    if img is None:
                        continue
                    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                    img = cv2.resize(img, image_size)
                    X.append(np.array(img))
                    y.append(label)

    return np.array(X), np.array(y)

# Load images (only people with >= 2 images)
X, y = load_images_filtered("/content/lfw_dataset", min_images_per_class=2)

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Train/test split with stratification
X_train, X_test, y_train, y_test = train_test_split(
    X, y_encoded, test_size=0.2, stratify=y_encoded, random_state=42
)

print(f"Loaded {len(X)} images from {len(le.classes_)} people (each with ≥2 images).")
```

```
Loaded 677 images from 39 people (each with ≥2 images).
```

```
from facenet_pytorch import InceptionResnetV1
import torch
from torchvision import transforms

# Load FaceNet
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
facenet = InceptionResnetV1(pretrained='vggface2').eval().to(device)

# Preprocessing
transform = transforms.Compose([
    transforms.ToPILImage(),
    transforms.Resize((160, 160)),
    transforms.ToTensor(),
    transforms.Normalize([0.5]*3, [0.5]*3)
])

def get_facenet_embeddings(images):
    embeddings = []
    for img in images:
        img_t = transform(img.astype(np.uint8)).unsqueeze(0).to(device)
        with torch.no_grad():
            emb = facenet(img_t).cpu().numpy().flatten()
        embeddings.append(emb)
    return np.array(embeddings)

# Extract FaceNet features
X_train_deep = get_facenet_embeddings(X_train)
X_test_deep = get_facenet_embeddings(X_test)
```

```
from skimage.feature import local_binary_pattern, hog

def extract_lbp_hog_features(images):
    features = []
    for img in images:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

        # LBP
        lbp = local_binary_pattern(gray, P=8, R=1, method='uniform')
        lbp_hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, 11), density=True)

        # HOG
        hog_feat = hog(gray, pixels_per_cell=(8, 8), cells_per_block=(2, 2), feature_vector=True)

        # Combine LBP + HOG
        combined = np.hstack((lbp_hist, hog_feat))
        features.append(combined)
    return np.array(features)

X_train_traditional = extract_lbp_hog_features(X_train)
X_test_traditional = extract_lbp_hog_features(X_test)
```

```
# Fuse Deep + Traditional features
X_train_combined = np.hstack((X_train_deep, X_train_traditional))
X_test_combined = np.hstack((X_test_deep, X_test_traditional))
```

```
from sklearn.svm import SVC

# SVM Classifier
clf = SVC(kernel='linear', probability=True)
clf.fit(X_train_combined, y_train)
```

```
SVC
SVC(kernel='linear', probability=True)
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Predictions
y_pred = clf.predict(X_test_combined)

# Metrics with zero_division=1 to avoid undefined precision
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=1))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
 [[ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0 56  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0 22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  2  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  1  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  2  1  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  3  0  0]
 [ 0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  2  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]]
```

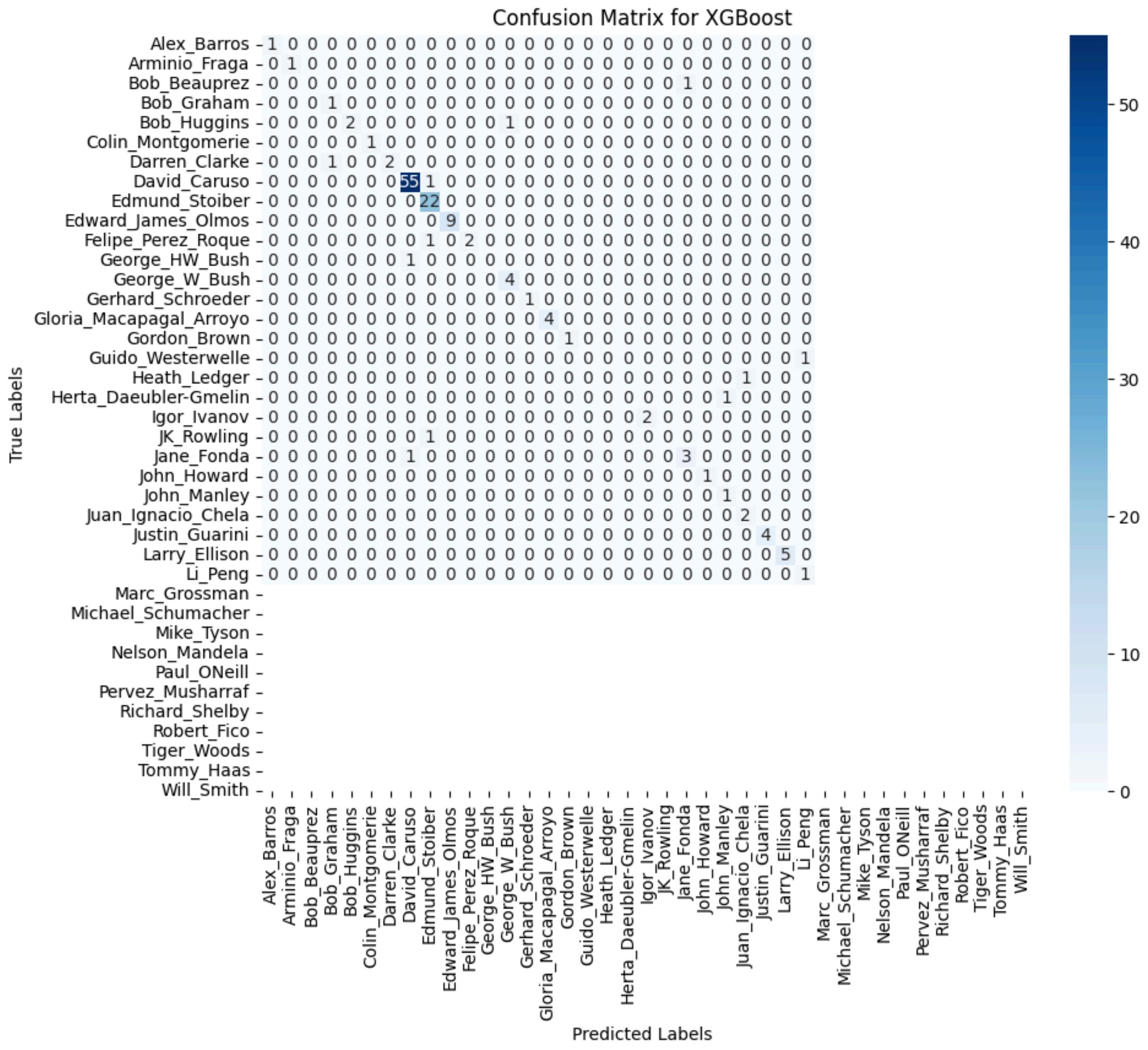
```
from sklearn.ensemble import RandomForestClassifier

# Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100, random_state=42)
rf_clf.fit(X_train_combined, y_train)

# Predictions
y_pred_rf = rf_clf.predict(X_test_combined)

# Metrics
print("Random Forest - Accuracy:", accuracy_score(y_test, y_pred_rf))
print("\nClassification Report:\n", classification_report(y_test, y_pred_rf, zero_division=1))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
```

```
Random Forest - Accuracy: 0.8529411764705882

Classification Report:
               precision    recall  f1-score   support

           1       1.00      1.00      1.00         1
           3       1.00      1.00      1.00         1
           4       1.00      0.00      0.00         1
           5       1.00      0.00      0.00         1
           8       0.50      0.67      0.57         3
          10       1.00      0.00      0.00         1
          11       1.00      0.33      0.50         3
          12       0.85      1.00      0.92        56
          13       0.79      1.00      0.88        22
          14       0.90      1.00      0.95         9
          15       1.00      0.67      0.80         3
          17       1.00      0.00      0.00         1
          19       1.00      1.00      1.00         4
          20       1.00      0.00      0.00         1
          22       1.00      1.00      1.00         4
          23       1.00      0.00      0.00         1
          24       1.00      0.00      0.00         1
          25       1.00      0.00      0.00         1
          26       1.00      0.00      0.00         1
          27       1.00      0.50      0.67         2
          28       1.00      0.00      0.00         1
          29       1.00      0.75      0.86         4
          30       1.00      0.00      0.00         1
          31       1.00      0.00      0.00         1
          32       1.00      0.50      0.67         2
          33       1.00      1.00      1.00         4
          36       0.83      1.00      0.91         5
          37       1.00      0.00      0.00         1

    accuracy                           0.85       136
   macro avg       0.96      0.44      0.45       136
weighted avg       0.88      0.85      0.81       136


Confusion Matrix:
 [[ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0 56  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0 22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
```

```
import xgboost as xgb
from xgboost import XGBClassifier

# XGBoost Classifier
xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42)
xgb_clf.fit(X_train_combined, y_train)

# Predictions
y_pred_xgb = xgb_clf.predict(X_test_combined)

# Metrics
print("XGBoost - Accuracy:", accuracy_score(y_test, y_pred_xgb))
print("\nClassification Report:\n", classification_report(y_test, y_pred_xgb, zero_division=1))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred_xgb))
```

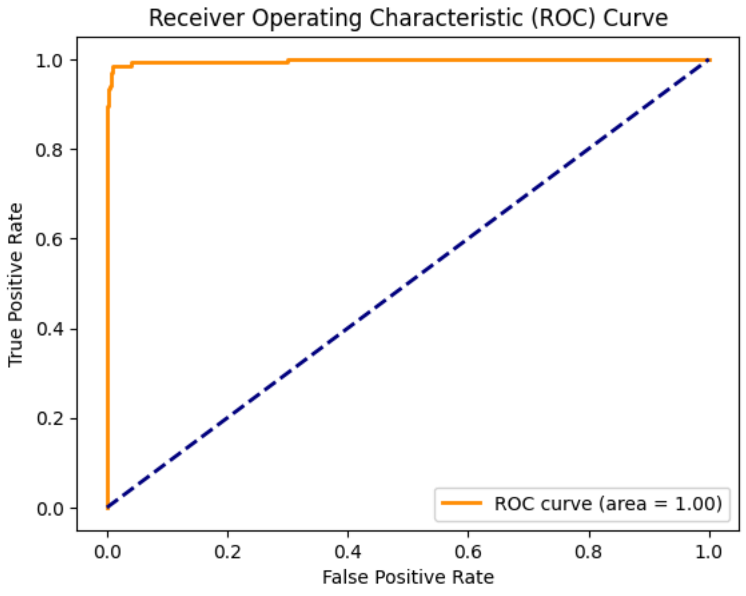
```
Confusion Matrix:
 [[ 1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
   0  0  0  0]
 [ 0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  2  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  1  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0 55  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0 22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  1]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   1  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
   0  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   2  0  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  4  0  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  5  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
   0  0  0  1]]
```

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred_xgb)

# Plot the confusion matrix
plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=le.classes_, yticklabels=le.classes_)
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix for XGBoost')
plt.show()
```

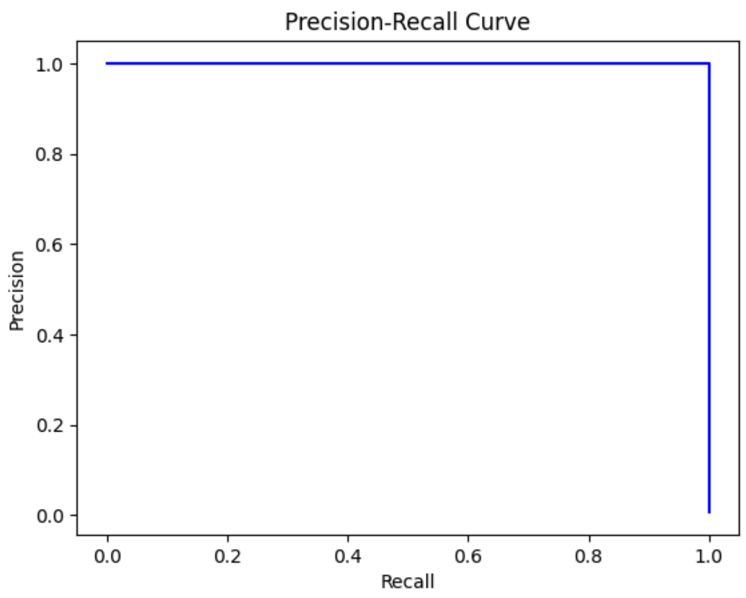


```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize
import xgboost as xgb
from xgboost import XGBClassifier


# Binarize the labels for multi-class classification
y_test_bin = label_binarize(y_test, classes=range(len(le.classes_)))

# XGBoost Classifier
xgb_clf = XGBClassifier(use_label_encoder=False, eval_metric='mlogloss', random_state=42)
xgb_clf.fit(X_train_combined, y_train) # Fit the model before predicting

# Get ROC curve for each class
# Use xgb_clf instead of best_xgb_model
fpr, tpr, _ = roc_curve(y_test_bin.ravel(), xgb_clf.predict_proba(X_test_combined).ravel())
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [18:12:51] WARNING: /workspace/src/learner.cc:740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```



```
from sklearn.metrics import precision_recall_curve
import matplotlib.pyplot as plt
import xgboost as xgb
from xgboost import XGBClassifier

# Precision-Recall curve for one class (e.g., class 1)
precision, recall, _ = precision_recall_curve(y_test_bin[:, 1], xgb_clf.predict_proba(X_test_combined)[:, 1])

plt.plot(recall, precision, color='blue')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve')
plt.show()
```



Deployment & Optimization

```
import joblib
from sklearn.decomposition import PCA

# want to keep 95% of the variance
pca = PCA(n_components=0.95)

# Fit PCA on your training data (e.g., X_train_combined)
pca.fit(X_train_combined)

# Save trained SVM model
joblib.dump(clf, "svm_face_recognition.pkl")

# Save PCA transformer
joblib.dump(pca, "pca_transform.pkl")
```

```
['pca_transform.pkl']
```

```
!pip install skl2onnx
```

```
Collecting skl2onnx
  Downloading skl2onnx-1.18.0-py2.py3-none-any.whl.metadata (3.2 kB)
Collecting onnx>=1.2.1 (from skl2onnx)
  Downloading onnx-1.17.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (16 kB)
Requirement already satisfied: scikit-learn>=1.1 in /usr/local/lib/python3.11/dist-packages (from skl2onnx) (1.6.1)
Collecting onnxconverter-common>=1.7.0 (from skl2onnx)
  Downloading onnxconverter_common-1.14.0-py2.py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.11/dist-packages (from onnx>=1.2.1->skl2onnx) (1.26.4)
Requirement already satisfied: protobuf>=3.20.2 in /usr/local/lib/python3.11/dist-packages (from onnx>=1.2.1->skl2onnx) (5.29.4)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from onnxconverter-common>=1.7.0->skl2onnx) (24.2)
Collecting protobuf>=3.20.2 (from onnx>=1.2.1->skl2onnx)
  Downloading protobuf-3.20.2-py2.py3-none-any.whl.metadata (720 bytes)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.1->skl2onnx) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.1->skl2onnx) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.1->skl2onnx) (3.6.0
Downloading skl2onnx-1.18.0-py2.py3-none-any.whl (300 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.3/300.3 kB 7.0 MB/s eta 0:00:00
Downloading onnx-1.17.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.0/16.0 MB 81.2 MB/s eta 0:00:00
Downloading onnxconverter_common-1.14.0-py2.py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.5/84.5 kB 5.7 MB/s eta 0:00:00
Downloading protobuf-3.20.2-py2.py3-none-any.whl (162 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.1/162.1 kB 13.6 MB/s eta 0:00:00
Installing collected packages: protobuf, onnx, onnxconverter-common, skl2onnx
  Attempting uninstall: protobuf
    Found existing installation: protobuf 5.29.4
    Uninstalling protobuf-5.29.4:
      Successfully uninstalled protobuf-5.29.4
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
tensorflow-metadata 1.16.1 requires protobuf<6.0.0dev,>=4.25.2; python_version >= "3.11", but you have protobuf 3.20.2 which is incompat
grpcio-status 1.71.0 requires protobuf<6.0dev,>=5.26.1, but you have protobuf 3.20.2 which is incompatible.
tensorflow 2.18.0 requires keras>=3.5.0, but you have keras 2.11.0 which is incompatible.
tensorflow 2.18.0 requires protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3, but you have protobuf 3.20.
Successfully installed onnx-1.17.0 onnxconverter-common-1.14.0 protobuf-3.20.2 skl2onnx-1.18.0
```

```
!pip uninstall -y keras tensorflow
!pip install tensorflow==2.11.0
!pip install keras==2.11.0
```

```
Found existing installation: keras 2.11.0
Uninstalling keras-2.11.0:
  Successfully uninstalled keras-2.11.0
Found existing installation: tensorflow 2.18.0
Uninstalling tensorflow-2.18.0:
  Successfully uninstalled tensorflow-2.18.0
ERROR: Could not find a version that satisfies the requirement tensorflow==2.11.0 (from versions: 2.12.0rc0, 2.12.0rc1, 2.12.0, 2.12.1,
ERROR: No matching distribution found for tensorflow==2.11.0
Collecting keras==2.11.0
  Using cached keras-2.11.0-py2.py3-none-any.whl.metadata (1.4 kB)
Using cached keras-2.11.0-py2.py3-none-any.whl (1.7 MB)
Installing collected packages: keras
Successfully installed keras-2.11.0
```

```
!pip uninstall keras -y
```

```
Found existing installation: keras 2.11.0
Uninstalling keras-2.11.0:
  Successfully uninstalled keras-2.11.0
```

```
import os
os.kill(os.getpid(), 9)
```

```
!pip install tensorflow
```

```
Collecting tensorflow
  Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.1 kB)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Collecting protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 (from tensorflow)
```

```
    Downloading protobuf-5.29.4-cp38-abi3-manylinux2014_x86_64.whl.metadata (592 bytes)
  Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
  Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)
  Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
  Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.0.1)
  Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.13.0)
  Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
  Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
  Collecting tensorboard~=2.19.0 (from tensorflow)
    Downloading tensorboard-2.19.0-py3-none-any.whl.metadata (1.8 kB)
  Collecting keras>=3.5.0 (from tensorflow)
    Downloading keras-3.9.2-py3-none-any.whl.metadata (6.1 kB)
  Requirement already satisfied: numpy<2.2.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.26.4)
  Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
  Collecting ml-dtypes<1.0.0,>=0.5.1 (from tensorflow)
    Downloading ml_dtypes-0.5.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (21 kB)
  Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37
  Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (0.4
  Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
  Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.8)
  Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.14.1)
  Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorfl
  Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
  Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2
  Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2
  Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.7)
  Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard~=2.1
  Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard~=2.19.0->tensorflow) (3.1
  Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard~=2.19.0
  Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow)
  Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow
  Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->
  Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (644.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 644.9/644.9 MB 840.2 kB/s eta 0:00:00
  Downloading keras-3.9.2-py3-none-any.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 44.9 MB/s eta 0:00:00
  Downloading ml_dtypes-0.5.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 62.5 MB/s eta 0:00:00
  Downloading protobuf-5.29.4-cp38-abi3-manylinux2014_x86_64.whl (319 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 319.7/319.7 kB 20.3 MB/s eta 0:00:00
  Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 67.6 MB/s eta 0:00:00
  Installing collected packages: protobuf, ml-dtypes, tensorboard, keras, tensorflow
    Attempting uninstall: protobuf
      Found existing installation: protobuf 3.20.2
```

Autoencoder for Deepfake Detection

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, UpSampling2D

input_img = Input(shape=(50, 37, 1))  # example shape, adjust to your image size
x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

# Decoder
# Adjusted padding and strides in UpSampling2D & Conv2D to maintain size
x = Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Conv2D(16, (3, 3), activation='relu', padding='same')(x)  # <-- fixed
x = UpSampling2D((2, 2))(x)
decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

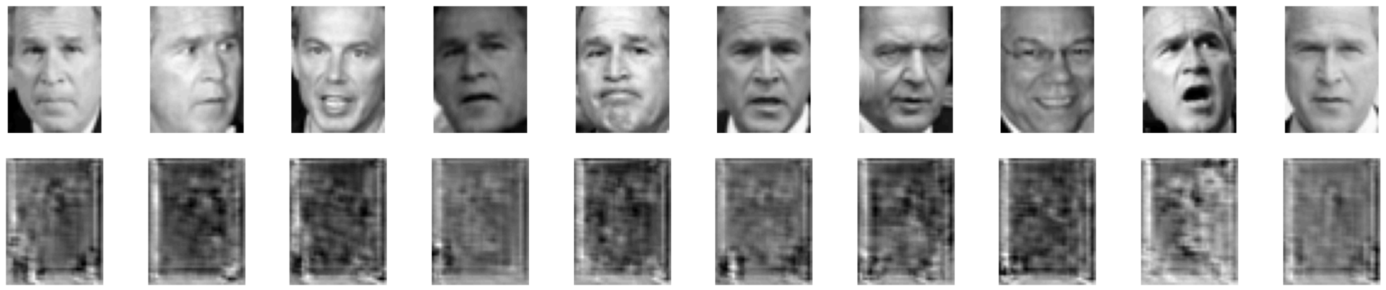
```
from sklearn.datasets import fetch_lfw_people
import numpy as np

lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
images = lfw_people.images
images = images / 255.0  # normalize
images = images.reshape(-1, 50, 37, 1)  # reshape to match autoencoder input
```

Visualize Reconstruction vs. Original

```
import matplotlib.pyplot as plt

decoded_imgs = autoencoder.predict(x_test[:10])

n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    # Original
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(50, 37), cmap='gray')
    plt.axis('off')

    # Reconstructed
    ax = plt.subplot(2, n, i + 1 + n)
    # Reshape to (52, 40) instead of (50, 37)
    plt.imshow(decoded_imgs[i].reshape(52, 40), cmap='gray')
    plt.axis('off')
plt.suptitle("Top: Original | Bottom: Reconstruction", fontsize=16)
plt.show()
```

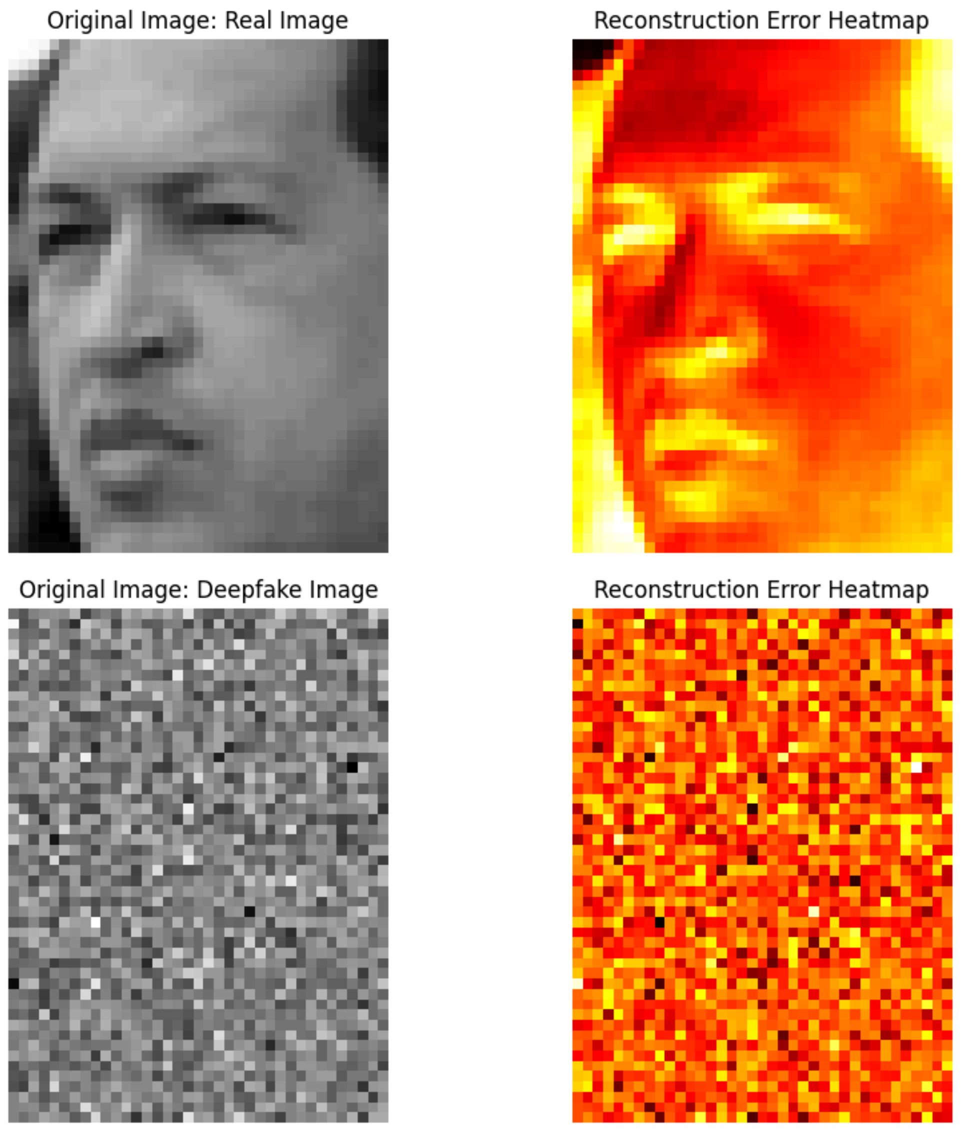
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step
```

Top: Original | Bottom: Reconstruction



```
import matplotlib.pyplot as plt
import numpy as np

def generate_heatmap(original_image, reconstructed_image):
    # Resize the reconstructed image to match the original image shape
    reconstructed_image = reconstructed_image[:, :original_image.shape[1], :]

    # Calculate the absolute difference between the original and the resized reconstructed image
    diff = np.abs(original_image - reconstructed_image)
    diff = np.mean(diff, axis=-1)  # Average across RGB channels to get a single channel heatmap

    # Normalize to [0, 1]
    diff = diff / np.max(diff)
    return diff

# Assume 'images' from previous cell is the real images
real_image = images[:10]  # Select 10 images as real images for demonstration
real_reconstructed = autoencoder.predict(real_image)  # Reconstruct real images using autoencoder

# Create synthetic deepfake images by adding noise (for demonstration)
# In a real scenario, you would use your actual deepfake images here
deepfake_image = real_image + np.random.normal(0, 0.1, real_image.shape)
deepfake_reconstructed = autoencoder.predict(deepfake_image)

# Visualize the heatmap
def plot_image_with_heatmap(image, heatmap, title=''):
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(image[0].reshape(50, 37), cmap='gray')  # Show the original image
    plt.title('Original Image: ' + title)
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(heatmap, cmap='hot')  # Show the heatmap
    plt.title('Reconstruction Error Heatmap')
    plt.axis('off')

    plt.show()

# Plot for real and deepfake images
plot_image_with_heatmap(real_image, generate_heatmap(real_image[0], real_reconstructed[0]), 'Real Image')
plot_image_with_heatmap(deepfake_image, generate_heatmap(deepfake_image[0], deepfake_reconstructed[0]), 'Deepfake Image')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 82ms/step
```



```
def get_reconstruction_error(original_image, reconstructed_image):
    # Resize the reconstructed image to match the original image shape before flattening
    reconstructed_image = reconstructed_image[:, :original_image.shape[1], :original_image.shape[2]]
    # Sliced along both dimensions to match original image shape

    # Flatten both images and compute mean squared error
    from sklearn.metrics import mean_squared_error  # Import mean_squared_error here
    return mean_squared_error(original_image.flatten(), reconstructed_image.flatten())

# Calculate reconstruction error for both real and deepfake images
real_error = get_reconstruction_error(real_image, real_reconstructed)
deepfake_error = get_reconstruction_error(deepfake_image, deepfake_reconstructed)

print("Reconstruction error for real image:", real_error)
print("Reconstruction error for deepfake image:", deepfake_error)
```

```
Reconstruction error for real image: 0.24808216094970703
Reconstruction error for deepfake image: 0.2637921103552475
```

Enhanced Feature Extraction Pipeline

```
from skimage.feature import local_binary_pattern, hog
from skimage.transform import pyramid_gaussian
import cv2
import numpy as np

def extract_enhanced_features(images):
    features = []
    for img in images:
        gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)

        # Multi-scale LBP
        radii = [1, 2, 3]
        n_points = [8, 16, 24]
        lbp_features = []
        for radius, n_point in zip(radii, n_points):
            lbp = local_binary_pattern(gray, n_point, radius, method='uniform')
            hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, n_point + 3), range=(0, n_point + 2))
            lbp_features.extend(hist)

        # Multi-scale HOG
        hog_features = []
        for scale in np.linspace(0.5, 1.5, 3):
            resized = cv2.resize(gray, (0,0), fx=scale, fy=scale)
            fd = hog(resized, orientations=9, pixels_per_cell=(8,8),
                    cells_per_block=(2,2), transform_sqrt=True, feature_vector=True)
            hog_features.extend(fd)

        # Gabor features
        gabor_features = []
        kernels = []
        for theta in np.arange(0, np.pi, np.pi/4):
            for sigma in (1, 3):
                for frequency in (0.05, 0.25):
                    kernel = cv2.getGaborKernel((21,21), sigma, theta, frequency, 0.5, 0, ktype=cv2.CV_32F)
                    kernels.append(kernel)

        for kernel in kernels:
            filtered = cv2.filter2D(gray, cv2.CV_8UC3, kernel)
            gabor_features.extend([filtered.mean(), filtered.std()])

        # Combine all features
        combined = np.hstack([lbp_features, hog_features, gabor_features])
        features.append(combined)

    return np.array(features)
```

```
from facenet_pytorch import MTCNN, InceptionResnetV1
import torch
from torchvision import transforms
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import StackingClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score

def train_advanced_classifier(X_train, y_train, X_test, y_test):
    # Base models
    svm = SVC(probability=True)
    rf = RandomForestClassifier()
    xgb = XGBClassifier(eval_metric='mlogloss', use_label_encoder=False)

    # Hyperparameter grids
    param_grid_svm = {
        'C': [0.1, 1, 10, 100],
        'gamma': ['scale', 'auto', 0.1, 1],
        'kernel': ['linear', 'rbf']
    }

    param_grid_rf = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10]
    }

    param_grid_xgb = {
        'n_estimators': [100, 200],
        'max_depth': [3, 6, 9],
        'learning_rate': [0.01, 0.1, 0.2]
    }
```