Notebook IndexError

```
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):
    # Added error handling for corrupted files
    try:
        with tarfile.open(dataset_path, "r:gz") as tar:
            tar.extractall(path=extract_path)
    except EOFError:
        print(f"Error: The file {dataset_path} is likely corrupted. Please re-download it.")
        # You might want to add code here to remove the corrupted file
        # os.remove(dataset_path)
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

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

```
Error: The file /content/lfw-funneled.tgz is likely corrupted. Please re-download it.
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)
```

```
!pip uninstall -y keras tensorflow
!pip install tensorflow==2.11.0
!pip install keras==2.11.0
!pip install keras_vggface #Install keras_vggface after installing compatible tensorflow and keras versions
```

```
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
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 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
```

```
Collecting keras-vggface
  Downloading keras_vggface-0.6-py3-none-any.whl.metadata (604 bytes)
Collecting keras-applications
  Downloading Keras_Applications-1.0.8-py3-none-any.whl.metadata (1.7 kB)
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) (0.15.0)
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
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras->keras-vggface) (2
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras->keras-v
Downloading keras_vggface-0.6-py3-none-any.whl (8.3 kB)
Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.7/50.7 kB 2.9 MB/s eta 0:00:00
Installing collected packages: keras-applications, keras-vggface
Successfully installed keras-applications-1.0.8 keras-vggface-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
```

```
!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
```

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

```
!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 3.8.0
Uninstalling keras-3.8.0:
  Successfully uninstalled keras-3.8.0
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
  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 17.4 MB/s eta 0:00:00
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 sourc
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
WARNING: The following packages were previously imported in this runtime:
  [keras]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
ERROR: Operation cancelled by user
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/cli/base_command.py", line 179, in exc_logging_wrapper
    status = run_func(*args)
             ^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/cli/req_command.py", line 67, in wrapper
    return func(self, options, args)
           ^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/commands/install.py", line 362, in run
    resolver = self.make_resolver(
               ^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/cli/req_command.py", line 177, in make_resolver
    return pip._internal.resolution.resolvelib.resolver.Resolver(
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/resolution/resolvelib/resolver.py", line 58, in __init__
    self.factory = Factory(
                   ^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/resolution/resolvelib/factory.py", line 127, in __init__
    self._installed_dists = {
                            ^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/resolution/resolvelib/factory.py", line 127, in <dictcomp>
    self._installed_dists = {
                            ^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/metadata/base.py", line 664, in <genexpr>
    return (d for d in it if d.canonical_name not in skip)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/metadata/base.py", line 612, in iter_all_distributions
    for dist in self._iter_distributions():
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/metadata/importlib/_envs.py", line 176, in _iter_distributions
    yield from finder.find(location)
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/metadata/importlib/_envs.py", line 79, in find
    for dist, info_location in self._find_impl(location):
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/metadata/importlib/_envs.py", line 64, in _find_impl
    raw_name = get_dist_name(dist)
               ^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/pip/_internal/metadata/importlib/_compat.py", line 52, in get_dist_name
    name = cast(Any, dist).name
           ^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/importlib/metadata/__init__.py", line 622, in name
```

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

```
Collecting facenet-pytorch
  Downloading facenet_pytorch-2.6.0-py3-none-any.whl.metadata (12 kB)
Collecting numpy<2.0.0,>=1.24.0 (from facenet-pytorch)
  Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (61 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.0/61.0 kB 3.8 MB/s eta 0:00:00
Collecting Pillow<10.3.0,>=10.2.0 (from facenet-pytorch)
  Downloading pillow-10.2.0-cp311-cp311-manylinux_2_28_x86_64.whl.metadata (9.7 kB)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in /usr/local/lib/python3.11/dist-packages (from facenet-pytorch) (2.32.3)
Collecting torch<2.3.0,>=2.2.0 (from facenet-pytorch)
  Downloading torch-2.2.2-cp311-cp311-manylinux1_x86_64.whl.metadata (25 kB)
Collecting torchvision<0.18.0,>=0.17.0 (from facenet-pytorch)
  Downloading torchvision-0.17.2-cp311-cp311-manylinux1_x86_64.whl.metadata (6.6 kB)
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->facen
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)
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-pyt
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3.0.0,>=2.0.0->facenet-pyt
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-
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
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.19.3 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl.metadata (1.7 kB)
Collecting triton==2.2.0 (from torch<2.3.0,>=2.2.0->facenet-pytorch)
  Downloading triton-2.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.4 kB)
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-p
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
Downloading facenet_pytorch-2.6.0-py3-none-any.whl (1.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 21.1 MB/s eta 0:00:00
Downloading numpy-1.26.4-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 28.2 MB/s eta 0:00:00
Downloading pillow-10.2.0-cp311-cp311-manylinux_2_28_x86_64.whl (4.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 32.8 MB/s eta 0:00:00
Downloading torch-2.2.2-cp311-cp311-manylinux1_x86_64.whl (755.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 755.6/755.6 MB 2.4 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 410.6/410.6 MB 4.1 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 67.9 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 50.3 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 823.6/823.6 kB 34.6 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.7/731.7 MB 2.8 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.6/121.6 MB 7.4 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 MB 13.1 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 MB 11.3 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.0/196.0 MB 6.4 MB/s eta 0:00:00
Downloading nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.0/166.0 MB 7.0 MB/s eta 0:00:00
Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 6.6 MB/s eta 0:00:00
Downloading triton-2.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (167.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 167.9/167.9 MB 6.7 MB/s eta 0:00:00
Downloading torchvision-0.17.2-cp311-cp311-manylinux1_x86_64.whl (6.9 MB)
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.9/6.9 MB 94.7 MB/s eta 0:00:00
Installing collected packages: triton, Pillow, nvidia-nvtx-cu12, nvidia-nccl-cu12, nvidia-cusparse-cu12, nvidia-curand-cu12, nvidia-cu
  Attempting uninstall: triton
    Found existing installation: triton 3.2.0
    Uninstalling triton-3.2.0:
      Successfully uninstalled triton-3.2.0
  Attempting uninstall: Pillow
    Found existing installation: pillow 11.1.0
    Uninstalling pillow-11.1.0:
      Successfully uninstalled pillow-11.1.0
  Attempting uninstall: nvidia-nvtx-cu12
    Found existing installation: nvidia-nvtx-cu12 12.4.127
    Uninstalling nvidia-nvtx-cu12-12.4.127:
      Successfully uninstalled nvidia-nvtx-cu12-12.4.127
  Attempting uninstall: nvidia-nccl-cu12
    Found existing installation: nvidia-nccl-cu12 2.21.5
    Uninstalling nvidia-nccl-cu12-2.21.5:
      Successfully uninstalled nvidia-nccl-cu12-2.21.5
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: numpy
    Found existing installation: numpy 2.0.2
    Uninstalling numpy-2.0.2:
      Successfully uninstalled numpy-2.0.2
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: torch
    Found existing installation: torch 2.6.0+cu124
    Uninstalling torch-2.6.0+cu124:
      Successfully uninstalled torch-2.6.0+cu124
  Attempting uninstall: torchvision
    Found existing installation: torchvision 0.21.0+cu124
    Uninstalling torchvision-0.21.0+cu124:
      Successfully uninstalled torchvision-0.21.0+cu124
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the sourc
thinc 8.3.6 requires numpy<3.0.0,>=2.0.0, but you have numpy 1.26.4 which is incompatible.
torchaudio 2.6.0+cu124 requires torch==2.6.0, but you have torch 2.2.2 which is incompatible.
tensorflow 2.18.0 requires keras>=3.5.0, but you have keras 2.11.0 which is incompatible.
Successfully installed Pillow-10.2.0 facenet-pytorch-2.6.0 numpy-1.26.4 nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nv
WARNING: The following packages were previously imported in this runtime:
  [PIL]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
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  [00:01<00:00,  71.6MB/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: 734 images from 39 people.
```

```
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 734 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)
```

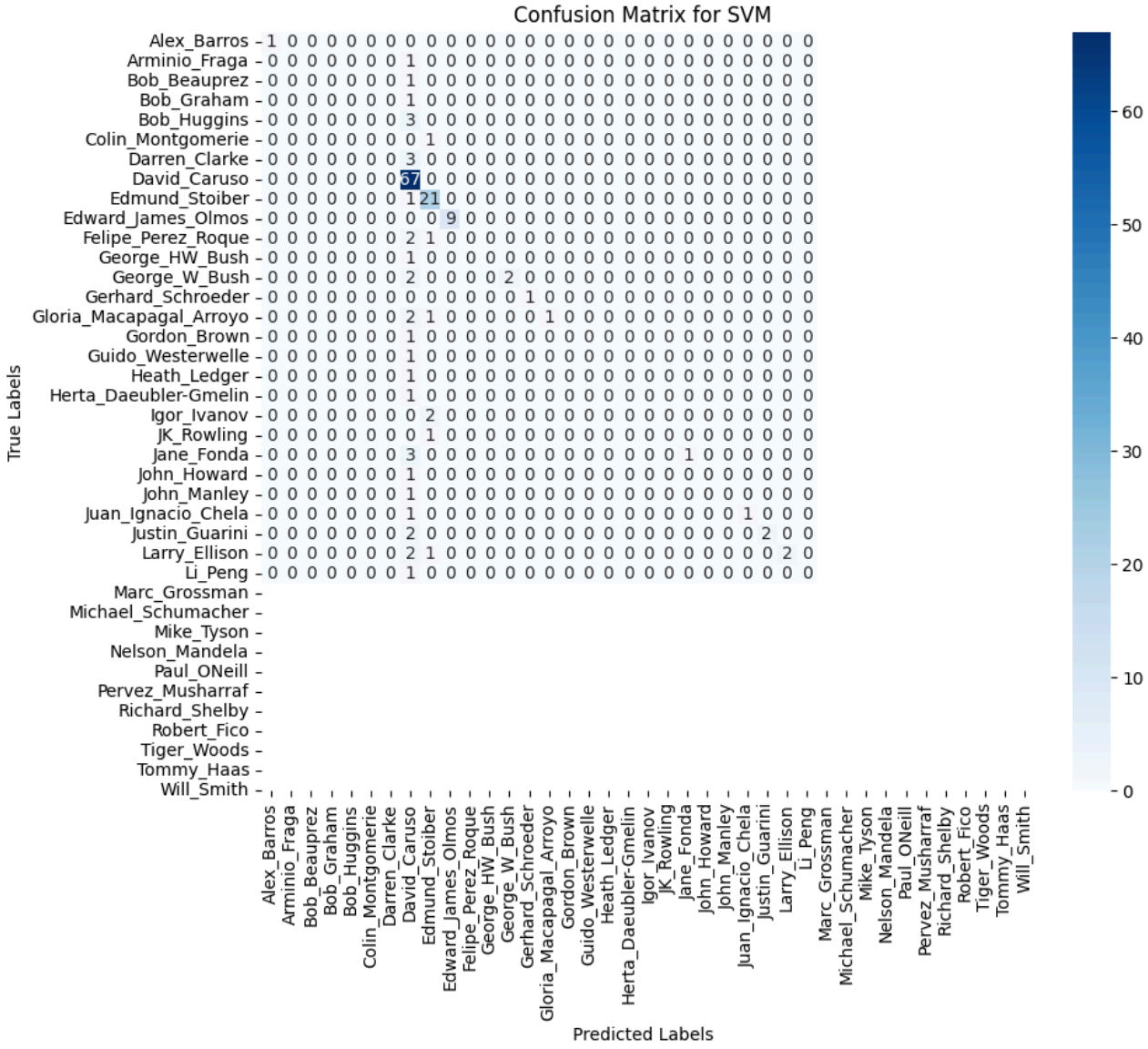
```
SVC
SVC(kernel='linear', probability=True)
```

```
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("SVM - 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))
```

```
    accuracy                           0.73       147
   macro avg       0.98      0.26      0.29       147
weighted avg       0.82      0.73      0.66       147


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  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  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  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 67  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 21  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  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  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  1  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  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  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  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  0  0]
 [ 0  0  0  0  0  0  0  3  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  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
   1  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  2  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
```

```
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)

# 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')
```

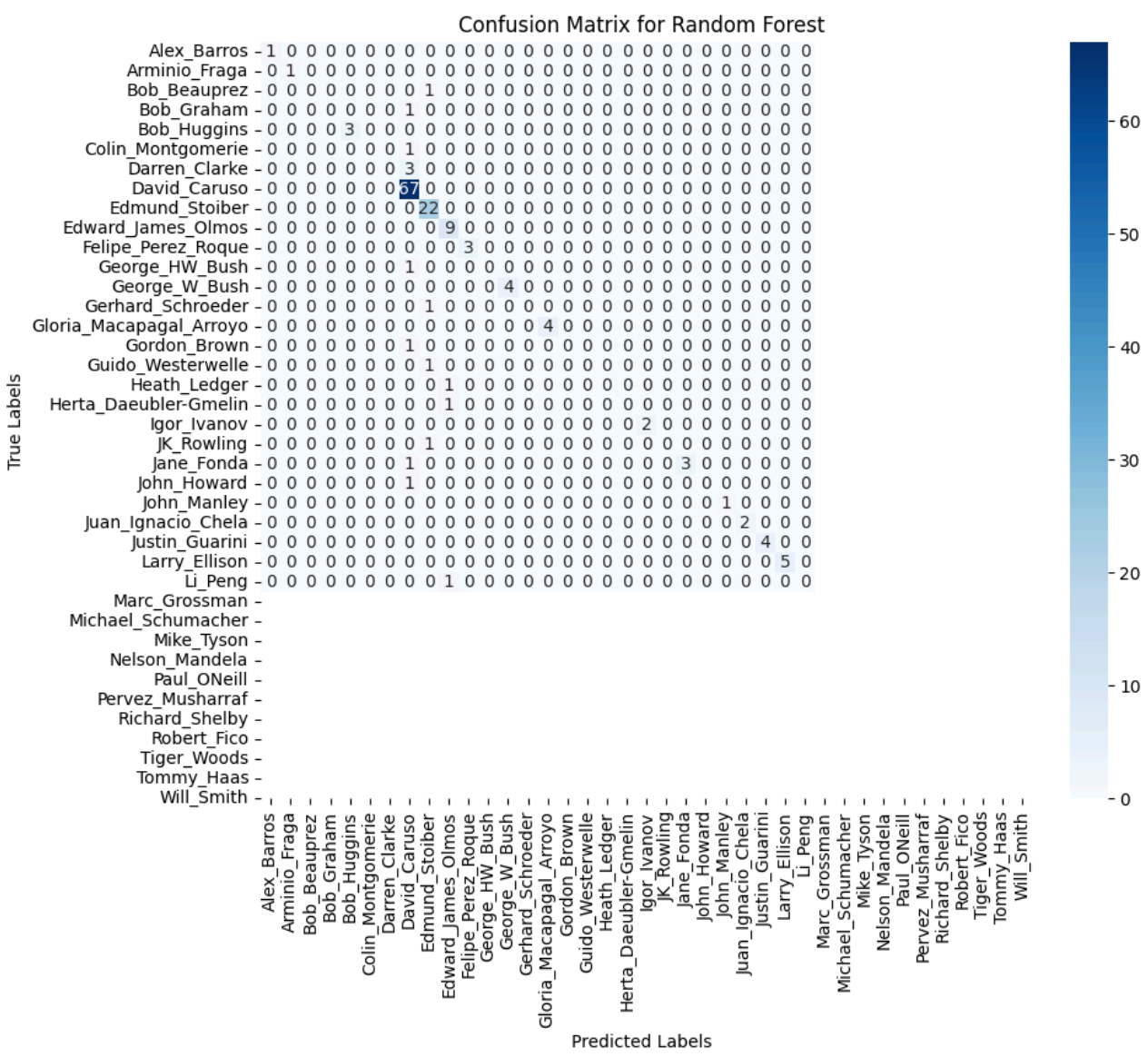
```
plt.ylabel('True Labels')
plt.title('Confusion Matrix for SVM')
plt.show()
```




```
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))
```

```
  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  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  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 67  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  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  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  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  4  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  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  2  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  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  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]
```

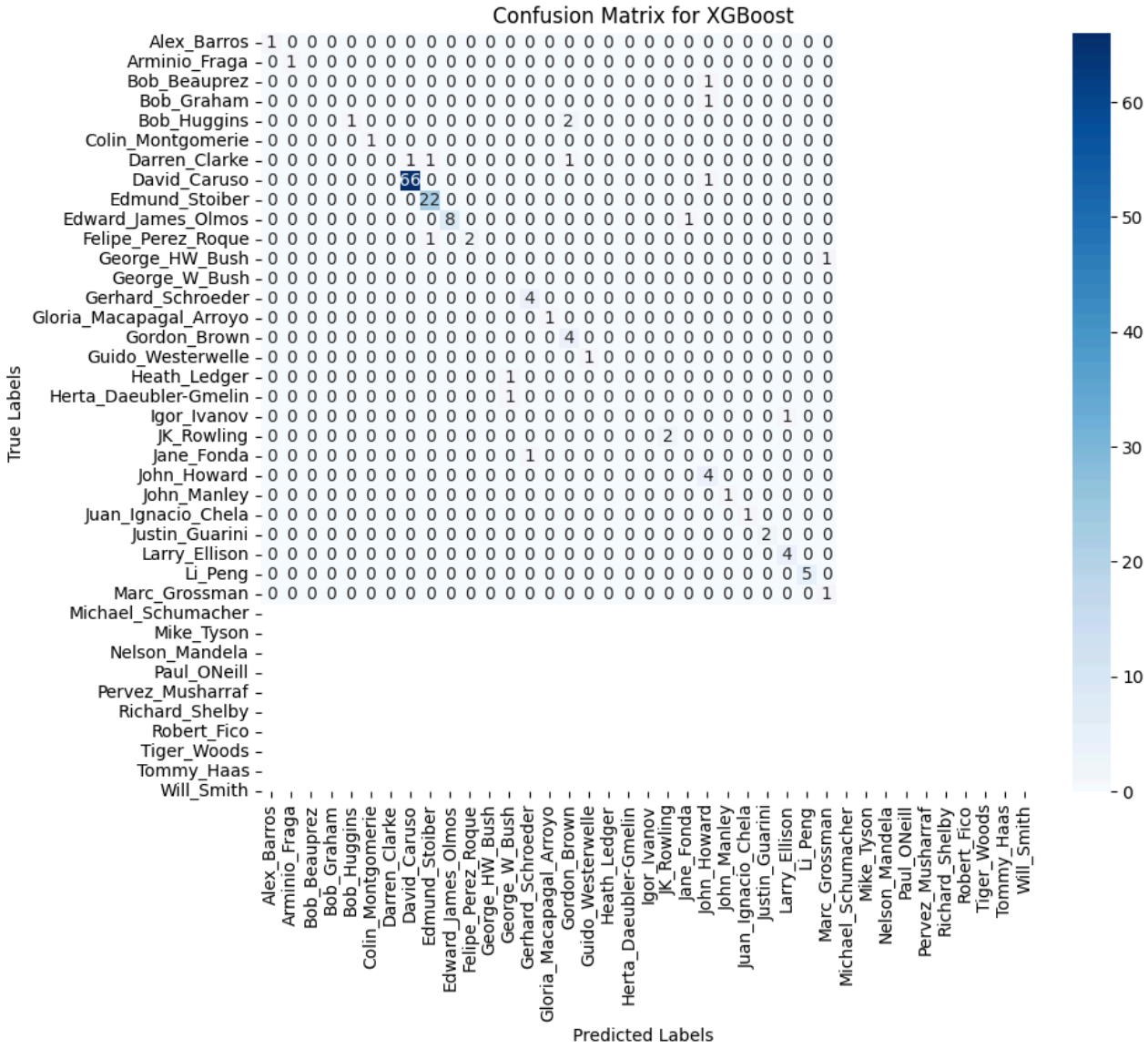
```
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt

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

# 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 Random Forest')
plt.show()
```




```
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))
```

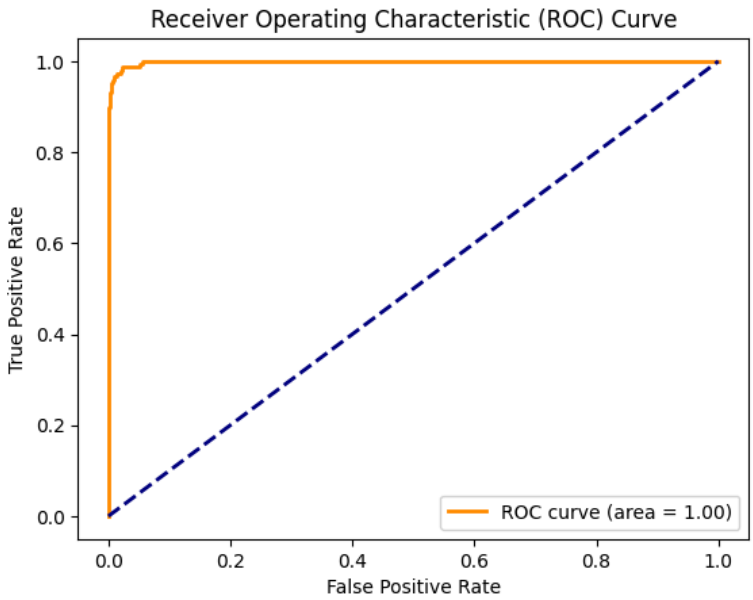
```
[ 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0]
[ 0  0  0  0  0  0  0  1  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
  0  0  0  0  0]
[ 0  0  0  0  0  0  0 66  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
  0  0  0  0  0]
[ 0  0  0  0  0  0  0  0 22  0  0  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  0  8  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  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  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  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  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  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  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  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  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  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  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  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
  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  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
  0  0  4  0  0]
```

```
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()
```

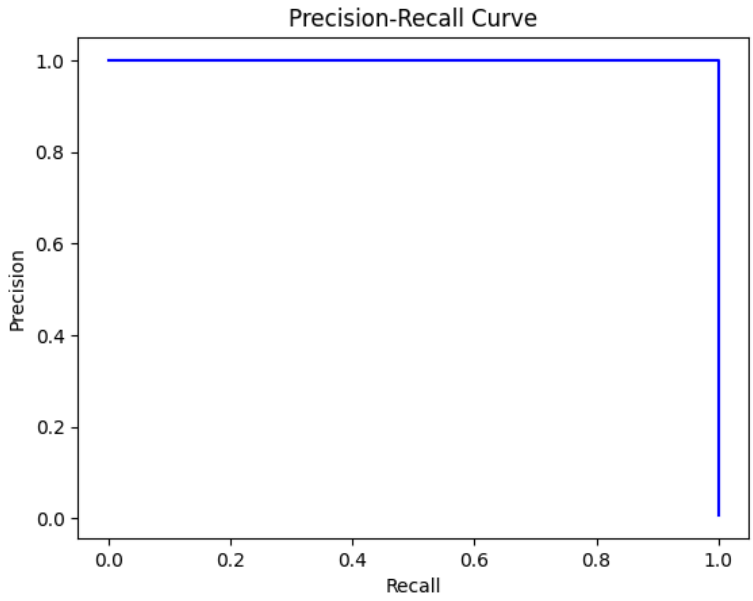



```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.preprocessing import label_binarize
import xgboost as xgb
from xgboost import XGBClassifier


# 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: [01:09:50] 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
Downloading skl2onnx-1.18.0-py2.py3-none-any.whl (300 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 300.3/300.3 kB 6.3 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 42.4 MB/s eta 0:00:00
Downloading onnxconverter_common-1.14.0-py2.py3-none-any.whl (84 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 84.5/84.5 kB 5.5 MB/s eta 0:00:00
Downloading protobuf-3.20.2-py2.py3-none-any.whl (162 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 162.1/162.1 kB 12.0 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 sourc
grpcio-status 1.71.0 requires protobuf<6.0dev,>=5.26.1, but you have protobuf 3.20.2 which is incompatible.
tensorflow-metadata 1.17.1 requires protobuf<6.0.0,>=4.25.2; python_version >= "3.11", but you have protobuf 3.20.2 which is incompati
ydf 0.11.0 requires protobuf<6.0.0,>=5.29.1, 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 3.2
Successfully installed onnx-1.17.0 onnxconverter-common-1.14.0 protobuf-3.20.2 skl2onnx-1.18.0
WARNING: The following packages were previously imported in this runtime:
  [google]
You must restart the runtime in order to use newly installed versions.
```

RESTART SESSION

```
!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
```

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')
```

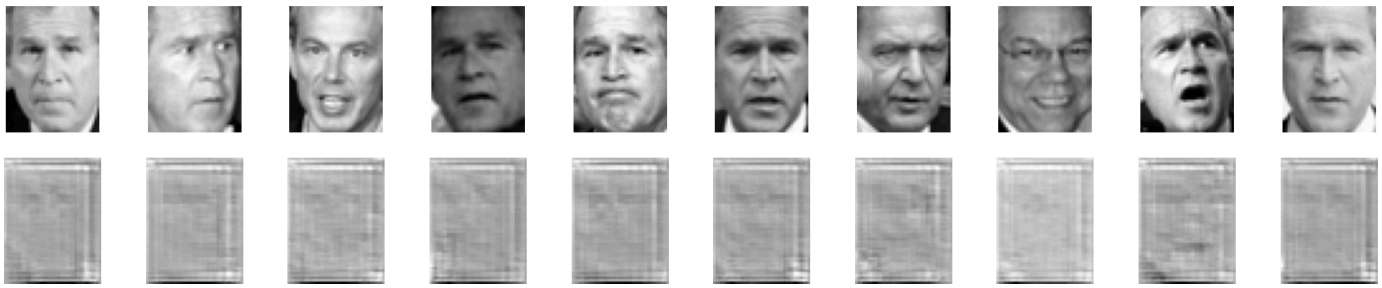
```
from sklearn.datasets import fetch_lfw_people
import numpy as np

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[:15])

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 173ms/step
```

Top: Original | Bottom: Reconstruction



```
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: 0.24804317951202393
Reconstruction error for deepfake image: 0.25763727626198735
```

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

def generate_heatmap(original_image, reconstructed_image):
    # Squeeze if needed
    if reconstructed_image.ndim > 2:
        reconstructed_image = reconstructed_image.squeeze()

    # Crop reconstructed image to match original image shape
    h, w = original_image.shape[:2]
    reconstructed_image = reconstructed_image[:h, :w]

    # Compute absolute difference
    diff = np.abs(original_image - reconstructed_image)
    diff = diff / np.max(diff)  # Normalize to [0, 1]
    return diff
```

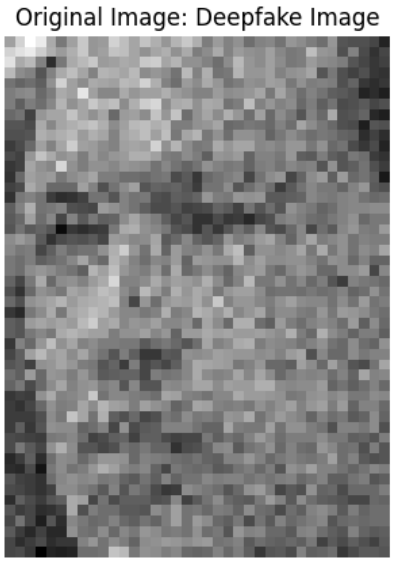
```
real_image = images[:10]  # Select 10 images as real images for demonstration
real_reconstructed = autoencoder.predict(
    real_image
)  # Reconstruct real images using autoencoder
# Create synthetic deepfake images by adding noise (for demonstration)
# In a real scenario, you would use your actual deepfake images here
deepfake_image = real_image + np.random.normal(0, 0.1, real_image.shape)
deepfake_reconstructed = autoencoder.predict(deepfake_image)
```

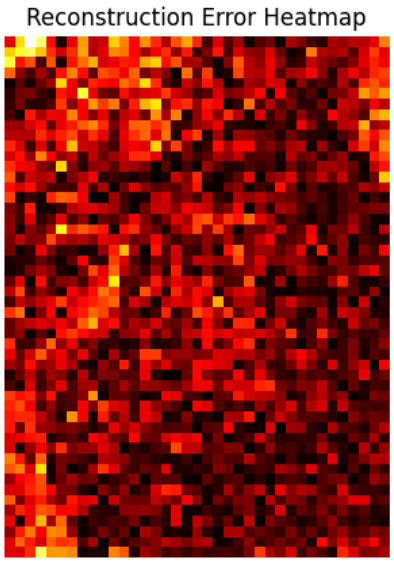
```
# Visualize the heatmap
def plot_image_with_heatmap(image, heatmap, title=""):
    plt.figure(figsize=(10, 5))
    plt.subplot(1, 2, 1)
    plt.imshow(image[0].reshape(50, 37), cmap="gray")  # Show the original image
    plt.title("Original Image: " + title)
    plt.axis("off")
    plt.subplot(1, 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 174ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 166ms/step
```





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

# Initialize MTCNN for face detection and alignment
mtcnn = MTCNN(keep_all=True, device='cuda:0' if torch.cuda.is_available() else 'cpu')
resnet = InceptionResnetV1(pretrained='vggface2').eval().to(mtcnn.device)

def get_aligned_facenet_embeddings(images):
    embeddings = []
    transform = transforms.Compose([
        transforms.ToPILImage(),
        transforms.Resize((160, 160)),
        transforms.ToTensor(),
        transforms.Normalize([0.5]*3, [0.5]*3)
    ])

    for img in images:
        # Detect and align faces
        faces = mtcnn(img)
        if faces is not None:
            # Get embeddings for each detected face
            face_embedding = resnet(faces.unsqueeze(0)).detach().cpu().numpy().flatten()
            embeddings.append(face_embedding)
        else:
            # Fallback to center crop if no face detected
            img_t = transform(img).unsqueeze(0).to(mtcnn.device)
            with torch.no_grad():
                embedding = resnet(img_t).cpu().numpy().flatten()
            embeddings.append(embedding)

    return np.array(embeddings)
```

```
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]
    }

    # Grid search for best parameters
    grid_svm = GridSearchCV(svm, param_grid_svm, cv=3, n_jobs=-1, verbose=1)
    grid_rf = GridSearchCV(rf, param_grid_rf, cv=3, n_jobs=-1, verbose=1)
    grid_xgb = GridSearchCV(xgb, param_grid_xgb, cv=3, n_jobs=-1, verbose=1)

    grid_svm.fit(X_train, y_train)
    grid_rf.fit(X_train, y_train)
    grid_xgb.fit(X_train, y_train)

    # Stacking classifier with best estimators
    estimators = [
        ('svm', grid_svm.best_estimator_),
        ('rf', grid_rf.best_estimator_),
        ('xgb', grid_xgb.best_estimator_)
    ]

    stack_clf = StackingClassifier(
        estimators=estimators,
        final_estimator=XGBClassifier(),
        stack_method='predict_proba',
        n_jobs=-1
    )

    stack_clf.fit(X_train, y_train)

    # Evaluation
    y_pred = stack_clf.predict(X_test)
    print("Stacking Classifier Accuracy:", accuracy_score(y_test, y_pred))
    print("\nClassification Report:\n", classification_report(y_test, y_pred))

    return stack_clf
```

```
import cv2
from PIL import Image
import numpy as np

def real_time_face_recognition(model, facenet_model, mtcnn, label_encoder):
    cap = cv2.VideoCapture(0)

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Convert to RGB
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # Detect faces
        boxes, _ = mtcnn.detect(rgb_frame)

        if boxes is not None:
            for box in boxes:
                x1, y1, x2, y2 = map(int, box)
                face = rgb_frame[y1:y2, x1:x2]

                # Get embedding
                face_img = Image.fromarray(face)
                face_tensor = mtcnn(face_img)
                if face_tensor is not None:
                    embedding = facenet_model(face_tensor.unsqueeze(0)).detach().cpu().numpy()

                    # Extract traditional features
                    traditional_features = extract_enhanced_features([face])

                    # Combine features
                    combined_features = np.hstack([embedding.flatten(), traditional_features[0]])

                    # Predict
                    pred = model.predict([combined_features])[0]
                    proba = model.predict_proba([combined_features])[0]
                    confidence = np.max(proba)

                    if confidence > 0.7:  # Confidence threshold
                        name = label_encoder.inverse_transform([pred])[0]
                        cv2.putText(frame, f"{name} ({confidence:.2f})", (x1, y1-10),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,255,0), 2)
                        cv2.rectangle(frame, (x1,y1), (x2,y2), (0,255,0), 2)
                    else:
                        cv2.putText(frame, "Unknown", (x1, y1-10),
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)
                        cv2.rectangle(frame, (x1,y1), (x2,y2), (0,0,255), 2)

        cv2.imshow('Face Recognition', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()


from albumentations import (
    Compose, RandomBrightnessContrast, HorizontalFlip,
    Rotate, GaussNoise, OpticalDistortion
)

augmentation = Compose([
    HorizontalFlip(p=0.5),
    RandomBrightnessContrast(p=0.2),
    Rotate(limit=20, p=0.3),
    GaussNoise(var_limit=(10.0, 50.0), p=0.2),
    OpticalDistortion(p=0.3)
])

def augment_image(image):
    augmented = augmentation(image=image)['image']
    return augmented
```

```
<ipython-input-24-5c8e00307353>:10: UserWarning: Argument(s) 'var_limit' are not valid for transform GaussNoise
  GaussNoise(var_limit=(10.0, 50.0), p=0.2),
```

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Lambda
from tensorflow.keras.models import Model

def build_siamese_model(input_shape, embedding_size=128):
    # Shared embedding network
    input_layer = Input(shape=input_shape)
    x = tf.keras.layers.Conv2D(64, (3,3), activation='relu')(input_layer)
    x = tf.keras.layers.MaxPooling2D(pool_size=(2,2))(x)
    x = tf.keras.layers.Conv2D(128, (3,3), activation='relu')(x)
    x = tf.keras.layers.MaxPooling2D(pool_size=(2,2))(x)
    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dense(embedding_size, activation=None)(x)
    x = tf.keras.layers.Lambda(lambda x: tf.math.l2_normalize(x, axis=1))(x)

    return Model(input_layer, x)

def triplet_loss(y_true, y_pred, alpha=0.2):
    anchor, positive, negative = y_pred[0], y_pred[1], y_pred[2]

    pos_dist = tf.reduce_sum(tf.square(anchor - positive), axis=-1)
    neg_dist = tf.reduce_sum(tf.square(anchor - negative), axis=-1)

    basic_loss = pos_dist - neg_dist + alpha
    loss = tf.reduce_sum(tf.maximum(basic_loss, 0.0))

    return loss


def predict_with_confidence(model, X_test, threshold=0.7):
    probas = model.predict_proba(X_test)
    max_proba = np.max(probas, axis=1)
    preds = np.where(max_proba > threshold,
                     model.predict(X_test),
                     -1)  # -1 for unknown
    return preds


from matplotlib import pyplot as plt
import numpy as np
import random
from sklearn.datasets import fetch_lfw_people
from sklearn.model_selection import train_test_split

# Fetch the LFW dataset (this was likely done in a previous step)
lfw_people = fetch_lfw_people(min_faces_per_person=70, resize=0.4)
images = lfw_people.images
# Assuming you've preprocessed or normalized 'images' as needed

# Split into training and testing sets (this was likely done in a previous step)
X_train, X_test, y_train, y_test = train_test_split(
    images, lfw_people.target, test_size=0.2, random_state=42
)

def plot_sample_faces(images, labels, num_samples=10):
    plt.figure(figsize=(15, 4))
    indices = random.sample(range(len(images)), num_samples)
    for i, idx in enumerate(indices):
        plt.subplot(1, num_samples, i + 1)
        img = images[idx]
        if img.max() <= 1.0:
            img = (img * 255).astype('uint8')  # Rescale if normalized
        else:
            img = img.astype('uint8')
        plt.imshow(img)
        plt.title(labels[idx], fontsize=8)
        plt.axis('off')
    plt.suptitle("Sample Images from LFW Dataset", fontsize=14)
    plt.tight_layout()
    plt.show()

plot_sample_faces(X_train, y_train)
```