```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Double-click (or enter) to edit

```
!unzip /content/drive/MyDrive/mod2diseases.zip
```

```
Streaming output truncated to the last 5000 lines.
  inflating: Melanoma/ISIC_0014157_downsampled.jpg
  inflating: Melanoma/ISIC_0014160_downsampled.jpg
  inflating: Melanoma/ISIC_0014163_downsampled.jpg
  inflating: Melanoma/ISIC_0014171_downsampled.jpg
  inflating: Melanoma/ISIC_0014173_downsampled.jpg
  inflating: Melanoma/ISIC_0014181_downsampled.jpg
  inflating: Melanoma/ISIC_0014183_downsampled.jpg
  inflating: Melanoma/ISIC_0014186_downsampled.jpg
  inflating: Melanoma/ISIC_0014187_downsampled.jpg
  inflating: Melanoma/ISIC_0014189_downsampled.jpg
  inflating: Melanoma/ISIC_0014190_downsampled.jpg
  inflating: Melanoma/ISIC_0014217_downsampled.jpg
  inflating: Melanoma/ISIC_0014219_downsampled.jpg
  inflating: Melanoma/ISIC_0014221_downsampled.jpg
  inflating: Melanoma/ISIC_0014222_downsampled.jpg
  inflating: Melanoma/ISIC_0014229_downsampled.jpg
  inflating: Melanoma/ISIC_0014233_downsampled.jpg
  inflating: Melanoma/ISIC_0014238_downsampled.jpg
  inflating: Melanoma/ISIC_0014249_downsampled.jpg
  inflating: Melanoma/ISIC_0014255_downsampled.jpg
  inflating: Melanoma/ISIC_0014270_downsampled.jpg
  inflating: Melanoma/ISIC_0014284_downsampled.jpg
  inflating: Melanoma/ISIC_0014288_downsampled.jpg
  inflating: Melanoma/ISIC_0014289_downsampled.jpg
  inflating: Melanoma/ISIC_0014290_downsampled.jpg
  inflating: Melanoma/ISIC_0014291_downsampled.jpg
  inflating: Melanoma/ISIC_0014302_downsampled.jpg
  inflating: Melanoma/ISIC_0014308_downsampled.jpg
  inflating: Melanoma/ISIC_0014316_downsampled.jpg
  inflating: Melanoma/ISIC_0014319_downsampled.jpg
  inflating: Melanoma/ISIC_0014324_downsampled.jpg
  inflating: Melanoma/ISIC_0014325_downsampled.jpg
  inflating: Melanoma/ISIC_0014327_downsampled.jpg
  inflating: Melanoma/ISIC_0014331_downsampled.jpg
  inflating: Melanoma/ISIC_0014336_downsampled.jpg
  inflating: Melanoma/ISIC_0014337_downsampled.jpg
  inflating: Melanoma/ISIC_0014347_downsampled.jpg
  inflating: Melanoma/ISIC_0014349_downsampled.jpg
  inflating: Melanoma/ISIC_0014357_downsampled.jpg
  inflating: Melanoma/ISIC_0014360_downsampled.jpg
  inflating: Melanoma/ISIC_0014361_downsampled.jpg
  inflating: Melanoma/ISIC_0014366_downsampled.jpg
  inflating: Melanoma/ISIC_0014369_downsampled.jpg
  inflating: Melanoma/ISIC_0014372_downsampled.jpg
  inflating: Melanoma/ISIC_0014385_downsampled.jpg
  inflating: Melanoma/ISIC_0014395_downsampled.jpg
  inflating: Melanoma/ISIC_0014422_downsampled.jpg
  inflating: Melanoma/ISIC_0014423_downsampled.jpg
  inflating: Melanoma/ISIC_0014428_downsampled.jpg
  inflating: Melanoma/ISIC_0014434_downsampled.jpg
  inflating: Melanoma/ISIC_0014454_downsampled.jpg
  inflating: Melanoma/ISIC_0014476_downsampled.jpg
  inflating: Melanoma/ISIC_0014478_downsampled.jpg
  inflating: Melanoma/ISIC_0014480_downsampled.jpg
  inflating: Melanoma/ISIC_0014486_downsampled.jpg
  inflating: Melanoma/ISIC_0014489_downsampled.jpg
  inflating: Melanoma/ISIC_0014501_downsampled.jpg
```

```
import os
import shutil
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
#import os

# Your code goes here...


# Define the path to the original dataset
original_data_path = '/content'

# Define the path where augmented data will be stored
augmented_data_path = '/content/augmented_data'

# Create the augmented data directory if it doesn't exist
if not os.path.exists(augmented_data_path):
    os.makedirs(augmented_data_path)

# Define the categories
categories = ['Actinic keratosis', 'Basal cell carcinoma', 'Benign keratosis', 'Dermatofibroma',
              'Melanocytic nevus', 'Melanoma', 'Squamous cell carcinoma', 'Vascular lesion']

# Define the target number of samples for each folder (e.g., the maximum number of samples)
target_num_samples = 12900

# Perform data augmentation for each category
for category in categories:
    # Create a directory for the current category in the augmented data path
    category_augmented_path = os.path.join(augmented_data_path, category)
    if not os.path.exists(category_augmented_path):
        os.makedirs(category_augmented_path)

    # Check the number of samples in the original folder
    original_folder_path = os.path.join(original_data_path, category)
    num_original_samples = len(os.listdir(original_folder_path))

    # Calculate the number of augmentation steps needed to reach the target number of samples
    augmentation_steps = target_num_samples // num_original_samples

    # Create an ImageDataGenerator for data augmentation
    datagen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )

    # Load images from the original folder and perform data augmentation
    for img_name in os.listdir(original_folder_path):
        img_path = os.path.join(original_folder_path, img_name)
        img = load_img(img_path)
        img_array = img_to_array(img)
        img_array = img_array.reshape((1,) + img_array.shape)  # Reshape for flow method

        # Generate augmented images
        i = 0
        for batch in datagen.flow(img_array, batch_size=1, save_to_dir=category_augmented_path, save_prefix='aug', save_format='jpg'):
            i += 1
            if i >= augmentation_steps:
                break  # Break after reaching the target number of samples

# Zip the augmented data folder
shutil.make_archive('/content/skindiseases_augmenteds', 'zip', augmented_data_path)
```

Start coding or generate with AI.

```
!unzip /content/skindiseases_augmenteds.zip
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.applications import MobileNetV2
from keras.models import Sequential
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from keras.optimizers import Adam
#from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras.applications import ResNet50
```

```
# Mount Google Drive if your dataset is stored there
```

```
# Define path to your dataset
data_path = '/content/skindiseases_augmenteds'

# Define the categories (normal and skin disease)
categories = ['Actinic keratosis', 'Basal cell carcinoma', 'Benign keratosis', 'Dermatofibroma',
              'Melanocytic nevus', 'Melanoma', 'Squamous cell carcinoma', 'Vascular lesion']
```

```
# Resize images to match MobileNetV2 input size
img_size = 112

# Load images and labels
data = []

for category in categories:
    path = os.path.join(data_path, category)
    label = categories.index(category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path, img))
        img_array = cv2.resize(img_array, (img_size, img_size))
        data.append([img_array, label])

# Shuffle the data
np.random.shuffle(data)

# Split the data into features and labels
X = []
y = []

for features, label in data:
    X.append(features)
    y.append(label)

# Convert features and labels to numpy arrays
X = np.array(X)
y = np.array(y)

# Normalize the data
X = X / 255.0

# Split the data 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)

# Convert labels to one-hot encoding
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)

# Load pre-trained MobileNetV2 model without the top classification layer
base_model = MobileNetV2(input_shape=(img_size, img_size, 3), include_top=False, weights='imagenet')

# Freeze the base model layers
for layer in base_model.layers:
    layer.trainable = False

# Initialize Sequential model
model = Sequential()

# Add base model
model.add(base_model)

# Add Flatten layer to convert 2D feature map to 1D feature vector
model.add(Flatten())

# Add Dense layer with 128 neurons and ReLU activation
model.add(Dense(128, activation='relu'))

# Add Dropout layer with dropout rate of 0.5
model.add(Dropout(0.5))

# Additional Convolutional blocks
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.25))

# Dense block with three Dense layers
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(8, activation='softmax'))
```

```
# Compile the model
```

```
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
model.save('skinmod2.h5')
```

```
# /content/skindiseasesall/normal/123123.jpg

import numpy as np
import cv2
from keras.models import load_model

# Load the trained model
model = load_model('/content/skinmod2.h5')

# Define categories
categories =['Actinic keratosis', 'Basal cell carcinoma', 'Benign keratosis', 'Dermatofibroma',
              'Melanocytic nevus', 'Melanoma', 'Squamous cell carcinoma', 'Vascular lesion']
# Function to preprocess and predict an image
def predict_image(image_path):
    img_size = 112
    img_array = cv2.imread(image_path)
    img_array = cv2.resize(img_array, (img_size, img_size))
    img_array = np.expand_dims(img_array, axis=0) / 255.0  # Normalize
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    confidence = prediction[0][predicted_class]
    predicted_category = categories[predicted_class]
    return predicted_category, confidence

# Path to the image you want to predict
image_path = '/content/skindiseasesall/normal/1231234asd.jpg'

# Predict the image
predicted_category, confidence = predict_image(image_path)
#if confidence < 0.5:
# print('not skin')
print("Predicted category:", predicted_category)
print("Confidence:", confidence)
```

```
!pip install gradio
```

```
import gradio as gr
from PIL import Image
import numpy as np
import cv2
from keras.models import load_model

# Load the trained model
model = load_model('/content/skinmod2.h5')

# Define categories
categories = ['Actinic keratosis', 'Basal cell carcinoma', 'Benign keratosis', 'Dermatofibroma',
              'Melanocytic nevus', 'Melanoma', 'Squamous cell carcinoma', 'Vascular lesion']

def predict_image(image_array):
    img_size = 112
    # Resize the image to the required size
    img_array = cv2.resize(image_array, (img_size, img_size))
    img_array = np.expand_dims(img_array, axis=0) / 255.0  # Normalize
    prediction = model.predict(img_array)
    predicted_class = np.argmax(prediction)
    confidence = prediction[0][predicted_class]
    predicted_category = categories[predicted_class]
    return predicted_category, confidence

def process_image(image_array):
    # Convert the PIL Image to NumPy array if necessary
    if isinstance(image_array, Image.Image):
```