// CODE TO TRAIN THE MODEL

```
import pathlib
import matplotlib.pyplot as plt
import PIL
import os
import tensorflow as tf
import cv2


from tensorflow.keras.metrics import Recall
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau

path1 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\akiec\\"
path2 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\bcc\\"
path3 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\bkl\\"
path4 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\df\\"
path5 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\mel\\"
path6 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\nvv\\"
path7 = "C:\\Users\\abhic\\Desktop\\Data\\HAM_reorganized\\vasc\\"

data_dir1 = pathlib.Path(path1)
data_dir2 = pathlib.Path(path2)
data_dir3 = pathlib.Path(path3)
data_dir4 = pathlib.Path(path4)
data_dir5 = pathlib.Path(path5)
data_dir6 = pathlib.Path(path6)
data_dir7 = pathlib.Path(path7)
```

```
dict_image ={
    'akiec':list(data_dir1.glob('*')),
    'bcc':list(data_dir2.glob('*')),
    'bkl':list(data_dir3.glob('*')),
    'df':list(data_dir4.glob('*')),
    'mel':list(data_dir5.glob('*')),
    'nvv':list(data_dir6.glob('*')),
    'vasc':list(data_dir7.glob('*')),

}

dict_image_num ={
    'akiec' : 0,
    'bcc':1,
    'bkl':2,
    'df':3,
    'mel':4,
    'nvv' :5,
    'vasc':6
}

x =[]
y=[]

for key , path in dict_image.items():
    for sbpath in path:
        image = Image.open(str(sbpath))
        image = ImageOps.fit(image,(28,28))
```

```
        img = np.array(image)
        img_reshape = img[np.newaxis , ...]
        x.append(img)
        y.append(dict_image_num[key])

import numpy as np
x = np.array(x)
y = np.array(y)

from tensorflow.keras.preprocessing.image import ImageDataGenerator
data_generator=ImageDataGenerator(rotation_range=20, # rotate the image 20 degrees
                                  width_shift_range=0.10, # Shift the pic width by a max of 5%
                                  height_shift_range=0.10, # Shift the pic height by a max of 5%
                                  rescale=1/255, # Rescale the image by normalzing it.
                                  shear_range=0.1, # Shear means cutting away part of the image (max 10%)
                                  zoom_range=0.1, # Zoom in by 10% max
                                  horizontal_flip=True,
                                  vertical_flip=True,
                                  fill_mode='nearest')

data_generator.fit(x)

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2 ,random_state =42)

from tensorflow import keras


model1=keras.Sequential([
```

```
keras.layers.Conv2D(64,(2,2),input_shape=(28,28,3),activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.BatchNormalization(),

keras.layers.Conv2D(512,(2,2),input_shape=(28,28,3),activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.BatchNormalization(),
keras.layers.Dropout(0.3),

keras.layers.Conv2D(1024,(2,2),input_shape=(28,28,3),activation='relu'),
keras.layers.MaxPooling2D(pool_size=(2, 2)),
keras.layers.BatchNormalization(),
keras.layers.Dropout(0.3),

keras.layers.Conv2D(1024,(1,1),input_shape=(28,28,3),activation='relu'),
keras.layers.MaxPooling2D(pool_size=(1, 1)),
keras.layers.BatchNormalization(),
keras.layers.Dropout(0.3),

keras.layers.Conv2D(1024,(1,1),input_shape=(28,28,3),activation='relu'),
keras.layers.MaxPooling2D(pool_size=(1, 1)),
keras.layers.BatchNormalization(),
keras.layers.Dropout(0.3),


keras.layers.Flatten(),
keras.layers.Dense(256,activation = 'relu'),
keras.layers.Dropout(0.5),
```

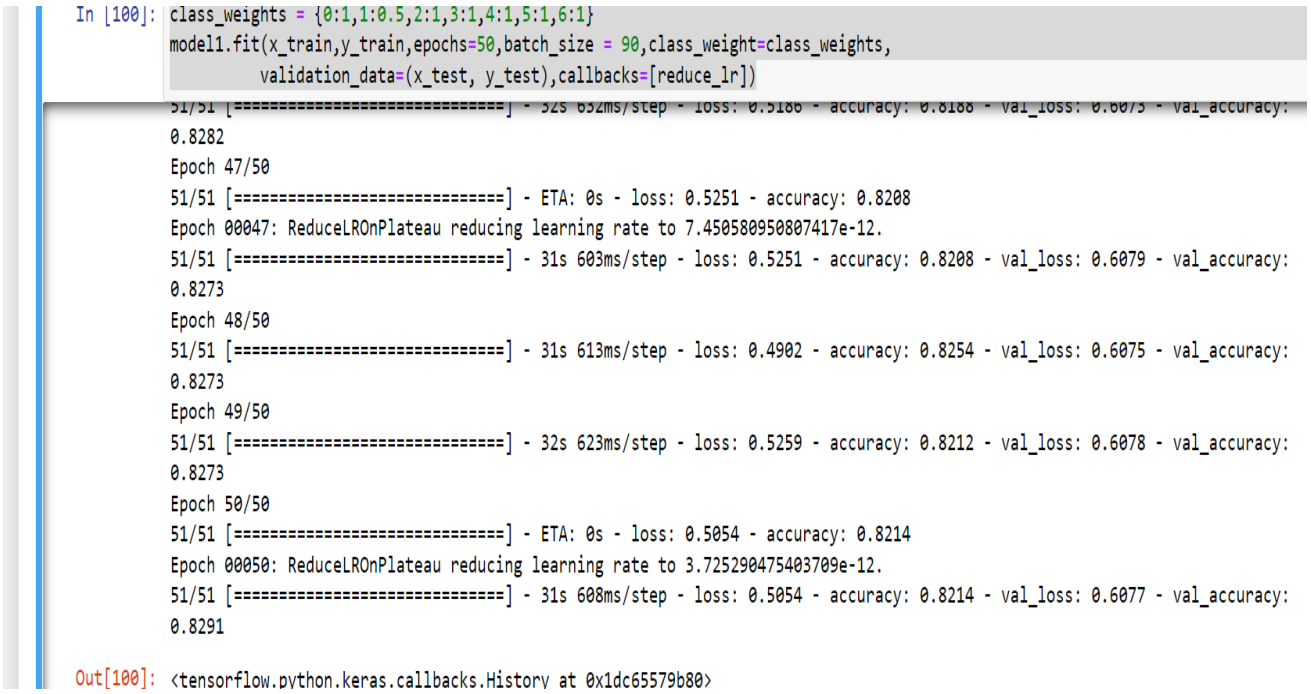
```
                keras.layers.Dense(7,activation='softmax') ])

# from keras.optimizers import adam
# opt = adam(lr = 0.01 )

model1.compile(optimizer= 'adam',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])

# # early=EarlyStopping(monitor='accuracy',patience=4,mode='auto')
reduce_lr = ReduceLROnPlateau(monitor='accuracy', factor=0.5, patience=2, verbose=1,
                    cooldown=0, mode='auto',min_delta=0.0001, min_lr=0.0)

class_weights = {0:1,1:0.5,2:1,3:1,4:1,5:1,6:1}
model1.fit(x_train,y_train,epochs=50,batch_size = 90,class_weight=class_weights,
       validation_data=(x_test, y_test),callbacks=[reduce_lr])
```



```
In [100]: class_weights = {0:1,1:0.5,2:1,3:1,4:1,5:1,6:1}
          model1.fit(x_train,y_train,epochs=50,batch_size = 90,class_weight=class_weights,
                   validation_data=(x_test, y_test),callbacks=[reduce_lr])
          51/51 [==============================] - 32s 652ms/step - loss: 0.5186 - accuracy: 0.8188 - val_loss: 0.6073 - val_accuracy:
          0.8282
          Epoch 47/50
          51/51 [==============================] - ETA: 0s - loss: 0.5251 - accuracy: 0.8208
          Epoch 00047: ReduceLROnPlateau reducing learning rate to 7.450580950807417e-12.
          51/51 [==============================] - 31s 603ms/step - loss: 0.5251 - accuracy: 0.8208 - val_loss: 0.6079 - val_accuracy:
          0.8273
          Epoch 48/50
          51/51 [==============================] - 31s 613ms/step - loss: 0.4902 - accuracy: 0.8254 - val_loss: 0.6075 - val_accuracy:
          0.8273
          Epoch 49/50
          51/51 [==============================] - 32s 623ms/step - loss: 0.5259 - accuracy: 0.8212 - val_loss: 0.6078 - val_accuracy:
          0.8273
          Epoch 50/50
          51/51 [==============================] - ETA: 0s - loss: 0.5054 - accuracy: 0.8214
          Epoch 00050: ReduceLROnPlateau reducing learning rate to 3.725290475403709e-12.
          51/51 [==============================] - 31s 608ms/step - loss: 0.5054 - accuracy: 0.8214 - val_loss: 0.6077 - val_accuracy:
          0.8291

Out[100]: <tensorflow.python.keras.callbacks.History at 0x1dc65579b80>
```

// Wep Application Using Streamlit.

```
import streamlit as st
import tensorflow as tf
import cv2
import numpy as np
import pandas as pd
from PIL import Image , ImageOps


# st.set_option("depreciation.showfileUploaderEncoding",False)
st.title("Skin Diseases Classification")

def load_model():
    model =
tf.keras.models.load_model("C:\\Users\\abhic\\Desktop\\Skin_Cancer_final_Model2")
    return model

st.write("""
         show skin classification
         """)


file = st.file_uploader("plz upload image ", type = ['jpg' , 'png'] )

def import_and_prdict(image_data, model):
    image = ImageOps.fit(image_data,(28,28))
    img = np.array(image)
    img_reshape = img[np.newaxis , ...]
    ar = model.predict(img_reshape)
    return ar


if file is  None:
    st.text("plz  upload the image")
else :
    image = Image.open(file)
    st.image(image,use_column_width=True)
    predictions = import_and_prdict(image, load_model())
```

```
    class_names = ['Actinic Keratoses' ,'Basal Cell Carcinoma','Benign Keratosis-
like Lessios','Dermatofibroma','Melanoma','Melanocylic nevi' ,'Vascular
Lessions']

    predictions = np.asarray(predictions)
    st.text(class_names[predictions.argmax(axis = 1)[0]])
```