```
#importing libraries
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from glob import glob
import seaborn as sns
from PIL import Image
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
import keras
import tensorflow as tf
from keras.applications import VGG19,Xception,VGG16
from keras.models import Sequential,Model
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from tensorflow.keras.layers import BatchNormalization
from keras.optimizers import Adam, RMSprop
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.resnet50 import ResNet50
from keras import layers
```

```
#importing train test data
#loading training data
train_datagen=ImageDataGenerator(rescale=1/255,shear_range=0.2,
                                 zoom_range=0.2,vertical_flip=True,
                                 rotation_range=40,
                                 brightness_range=(0.5,1.5),
                                 horizontal_flip=True)
train_data= train_datagen.flow_from_directory('/content/drive/MyDrive/data/train',
                                              target_size=(64,64),
                                              class_mode='sparse',
                                              shuffle=True,seed=1)
#loading validation dataset
test_datagen=ImageDataGenerator(rescale=1/255)
test_data=test_datagen.flow_from_directory('/content/drive/MyDrive/data/test',
                                           target_size=(64,64),
                                           class_mode='sparse',
                                           shuffle=True,seed=1)
```

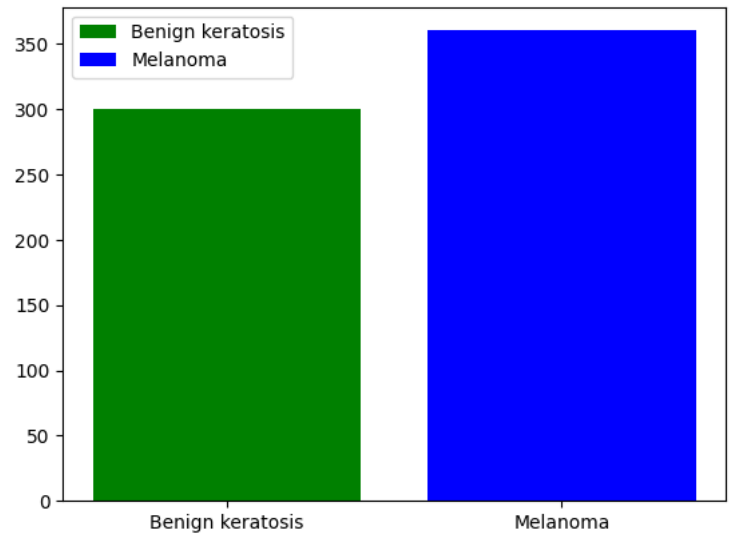
```
Found 2677 images belonging to 2 classes.
Found 660 images belonging to 2 classes.
```

```
#displaying the class names
names_class=['Benign keratosis','Melanoma']
for i in names_class:
  print(names_class.index(i)," ",i)
```

```
0   Benign keratosis
1   Melanoma
```

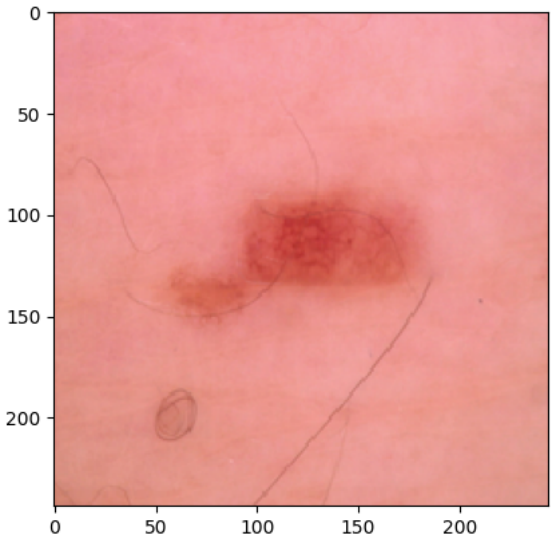
```
#visualizing the test_Data
fig,ax=plt.subplots()
ax.bar(['Benign keratosis'],[300],color='g',label='Benign keratosis')
ax.bar(['Melanoma'],[360],color='b',label='Melanoma')
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7f4ce9b08610>
```



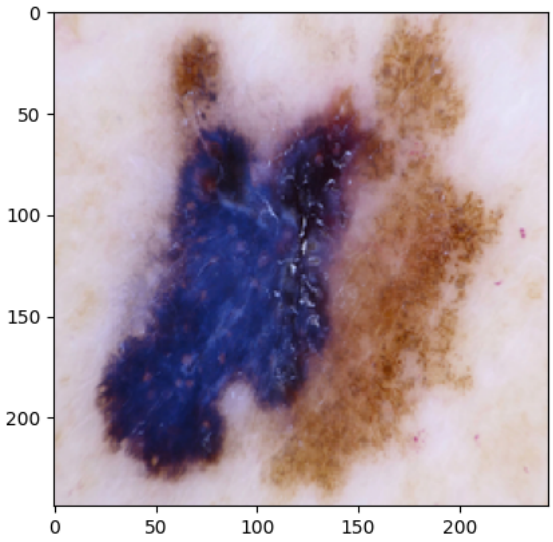
```
import keras.utils as image
img_path='/content/drive/MyDrive/data/train/benign/1030.jpg'
new_img=image.load_img(img_path,target_size=(244,244))
img=image.img_to_array(new_img)
img=np.expand_dims(img,axis=0)
print("Benign keratosis")
plt.imshow(new_img)
```

```
Benign keratosis
<matplotlib.image.AxesImage at 0x7f4c63ab06a0>
```




```
img_path='/content/drive/MyDrive/data/train/malignant/1284.jpg'
new_img=image.load_img(img_path,target_size=(244,244))
img=image.img_to_array(new_img)
img=np.expand_dims(img,axis=0)
print("Melanoma")
plt.imshow(new_img)
```

```
Melanoma
<matplotlib.image.AxesImage at 0x7f4c63a2e100>
```



Defining CNN Model

```
#defining CNN
model=tf.keras.models.Sequential([
    layers.BatchNormalization(),
    layers.Conv2D(32,3,activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64,3,activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.3),
    layers.Conv2D(128, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.2),
```

```
    layers.Conv2D(256, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dropout(0.15),
    layers.Dense(2, activation= 'softmax')
])


#compile the model
model.compile(optimizer='adam',loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])
#early stopping function
early=tf.keras.callbacks.EarlyStopping(monitor='loss',patience=3)
#early=tf.keras.callbacks.EarlyStopping(monitor='accuracy',patience=3)
#early=tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=3)


#fitting the model
histroy=model.fit(train_data,
                  epochs=19,
                  verbose=1,
                  validation_data=test_data
                  ,callbacks=[early])
```

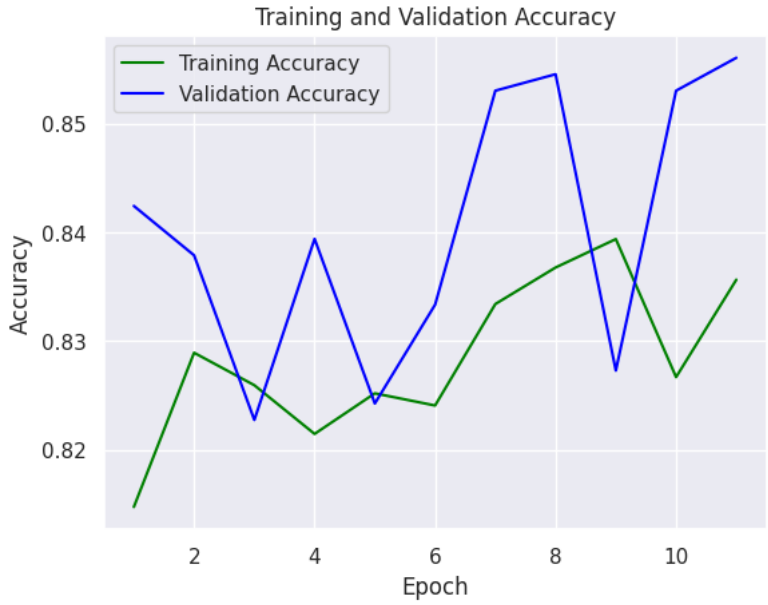
```
Epoch 1/19
84/84 [==============================] - 43s 504ms/step - loss: 0.3897 - accuracy: 0.8147 - val_loss: 0.3340 - val_accuracy: 0.8424
Epoch 2/19
84/84 [==============================] - 41s 485ms/step - loss: 0.3556 - accuracy: 0.8289 - val_loss: 0.3402 - val_accuracy: 0.8379
Epoch 3/19
84/84 [==============================] - 41s 490ms/step - loss: 0.3577 - accuracy: 0.8259 - val_loss: 0.3546 - val_accuracy: 0.8227
Epoch 4/19
84/84 [==============================] - 40s 470ms/step - loss: 0.3545 - accuracy: 0.8214 - val_loss: 0.3359 - val_accuracy: 0.8394
Epoch 5/19
84/84 [==============================] - 41s 490ms/step - loss: 0.3581 - accuracy: 0.8252 - val_loss: 0.3518 - val_accuracy: 0.8242
Epoch 6/19
84/84 [==============================] - 40s 466ms/step - loss: 0.3501 - accuracy: 0.8241 - val_loss: 0.3668 - val_accuracy: 0.8333
Epoch 7/19
84/84 [==============================] - 43s 509ms/step - loss: 0.3477 - accuracy: 0.8334 - val_loss: 0.3326 - val_accuracy: 0.8530
Epoch 8/19
84/84 [==============================] - 43s 507ms/step - loss: 0.3357 - accuracy: 0.8368 - val_loss: 0.3255 - val_accuracy: 0.8545
Epoch 9/19
84/84 [==============================] - 41s 486ms/step - loss: 0.3494 - accuracy: 0.8394 - val_loss: 0.3442 - val_accuracy: 0.8273
Epoch 10/19
84/84 [==============================] - 41s 484ms/step - loss: 0.3468 - accuracy: 0.8267 - val_loss: 0.3410 - val_accuracy: 0.8530
Epoch 11/19
84/84 [==============================] - 41s 483ms/step - loss: 0.3439 - accuracy: 0.8356 - val_loss: 0.3116 - val_accuracy: 0.8561
```

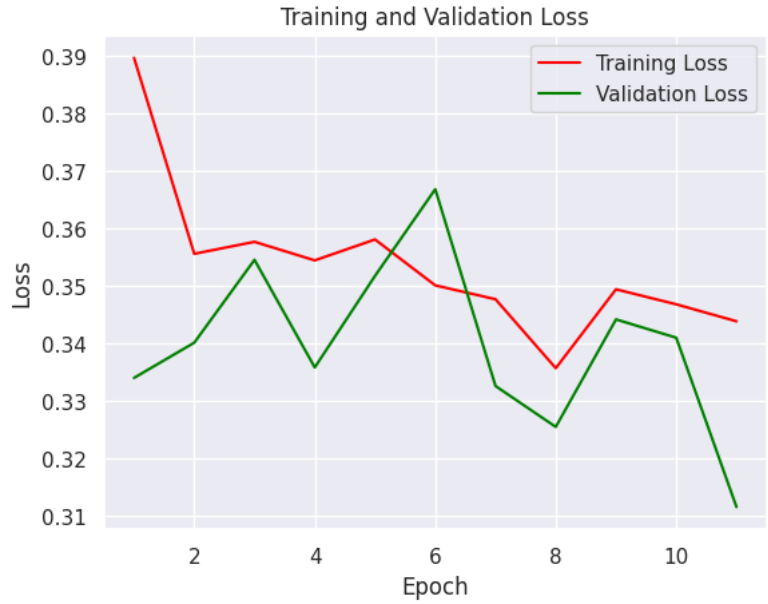
```
#evaluate the model
model.evaluate(test_data)
```

```
21/21 [==============================] - 4s 173ms/step - loss: 0.3116 - accuracy: 0.8561
[0.3116269111633301, 0.8560606241226196]
```

```
#plotting training values
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

acc = histroy.history['accuracy']
val_acc = histroy.history['val_accuracy']
loss = histroy.history['loss']
val_loss = histroy.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='red', label='Training Loss')
plt.plot(epochs, val_loss, color='green', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
```

```
plt.show()
```





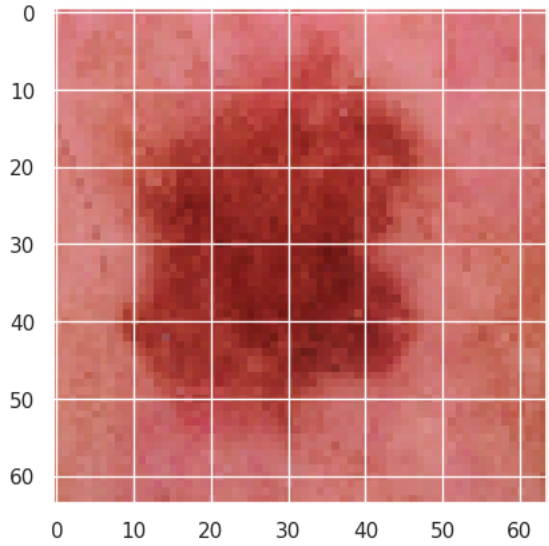

```
#predict validation data
y_pred = model.predict(test_data)
y_pred =  np.argmax(y_pred,axis=1)

print(y_pred)
```

```
21/21 [==============================] - 4s 167ms/step
[1 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0
 1 1 0 1 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1
 0 1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0 1 0 0
 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 1 0 1
 0 1 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1 0 1 1
 0 0 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1
 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 1 0 1 1 1 0 0 1 1 0
 1 0 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0
 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1 1 1 0 0 1 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 1 1
 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1
 0 1 1 1 0 0 0 1 0 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0
 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 0 1 1 1 0 0 1 0 1 0 1 1 0 0 0 0
 0 0 0 1 1 0 0 0 1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 0 0 1 1 0 0 0 1
 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1 1
 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 1 0 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1 0
 0 1 1 0 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1
 1 1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 1 0 1 0 1 0]
```

```
#example 1
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/benign/1253.jpg"
```

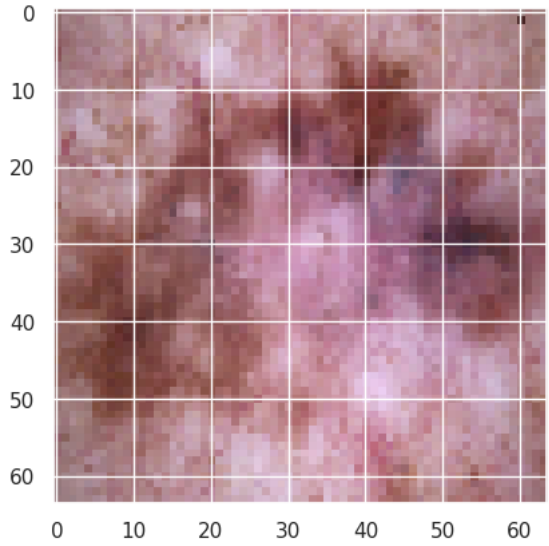
```
new_img = image.load_img(image_path, target_size=(64, 64))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(names_class[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 34ms/step
[0]
Benign keratosis
<matplotlib.image.AxesImage at 0x7f4c50b5a910>
```




```
#example 2
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/malignant/1119.jpg"
new_img = image.load_img(image_path, target_size=(64, 64))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(names_class[prediction[0]])
plt.imshow(new_img)
```

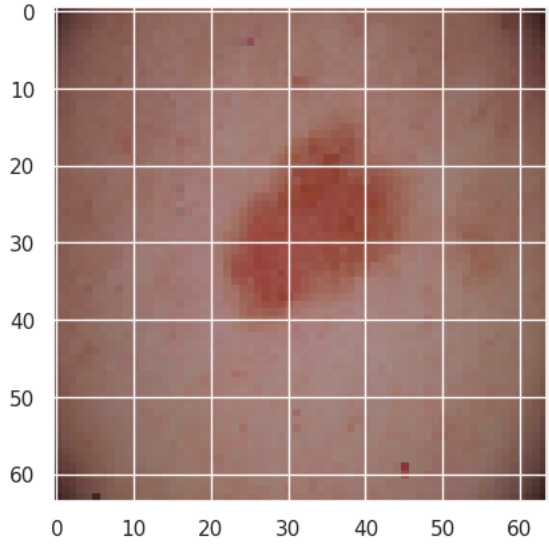
```
1/1 [==============================] - 0s 98ms/step
[1]
Melanoma
<matplotlib.image.AxesImage at 0x7f4c5070f640>
```




```
#example 3
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/benign/1288.jpg"
new_img = image.load_img(image_path, target_size=(64, 64))
```

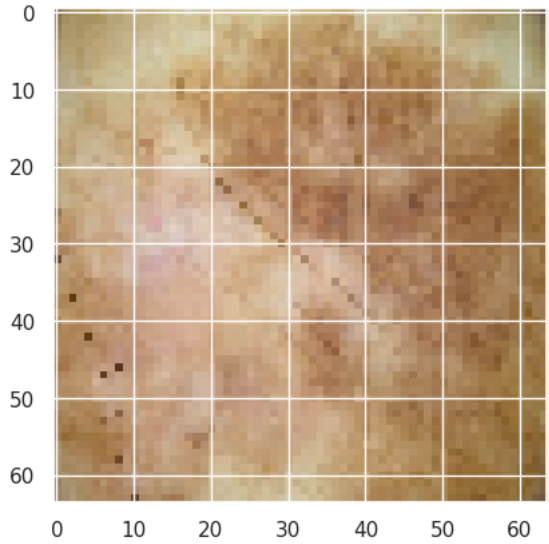
```
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(names_class[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 23ms/step
[0]
Benign keratosis
<matplotlib.image.AxesImage at 0x7f4c50d593d0>
```




```
#example 4
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/malignant/46.jpg"
new_img = image.load_img(image_path, target_size=(64, 64))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
prediction = model.predict(img)
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(names_class[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 47ms/step
[1]
Melanoma
<matplotlib.image.AxesImage at 0x7f4c509d3eb0>
```




```
model_version=1
model.save(f"../models/{model_version}")
```

```
ıch as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op, _updat
```

✓ 1s completed at 8:22 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.