```
#import libraries
import os
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
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
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
import tensorflow as tf
```

```
#import  train test data
## loading training set
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)
```

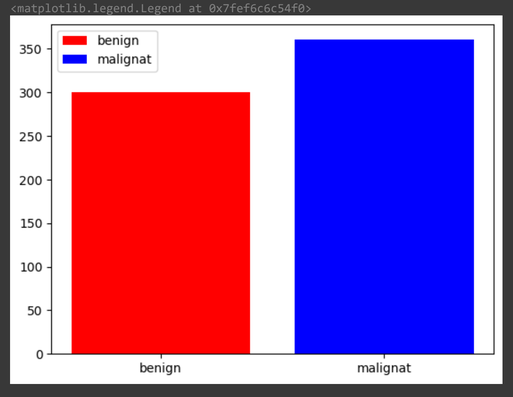
```
Found 2677 images belonging to 2 classes.
Found 660 images belonging to 2 classes.
```

```
#display classes names
class_names = ["Benign","Malignant"]
for i in class_names :
    print(class_names.index(i),"  ",i)
```

```
0    Benign
1    Malignant
```

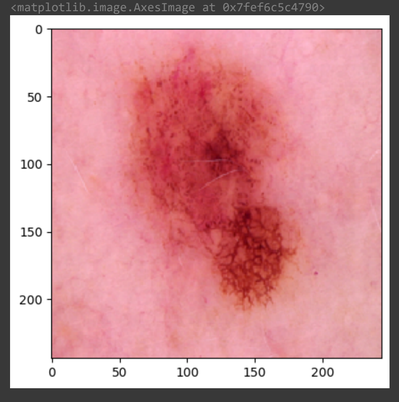
```
#visulaize test_data
fig, ax = plt.subplots()
ax.bar(["benign"],[300],color= "r",label='benign')
ax.bar(["malignat"],[360],color ="b",label ="malignat")
ax.legend()
```

```
<matplotlib.legend.Legend at 0x7fef6c6c54f0>
```



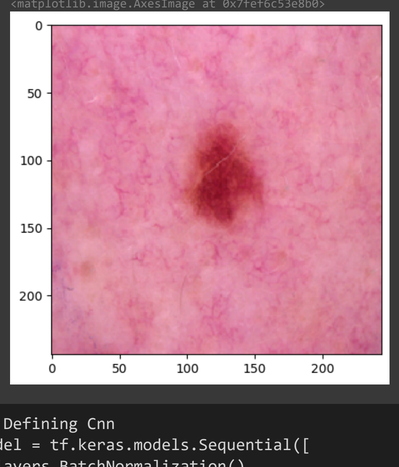

```
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/benign/1261.jpg"
new_img = image.load_img(image_path, target_size=(244, 244))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
print("benign")
plt.imshow(new_img)
```

```
benign
<matplotlib.image.AxesImage at 0x7fef6c5c4790>
```




```
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/benign/1080.jpg"
new_img = image.load_img(image_path, target_size=(244, 244))
img = image.img_to_array(new_img)
img = np.expand_dims(img, axis=0)
print("malignant")
plt.imshow(new_img)
```

```
malignant
<matplotlib.image.AxesImage at 0x7fef6c53e8b0>
```




```
## 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
import tensorflow as tf
model.compile(optimizer="adam", loss=keras.losses.SparseCategoricalCrossentropy(), metrics=['accuracy'])

#early stopping function
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
```

```
#fit the model
histroy=model.fit(train_data,
    validation_data = test_data,
    callbacks=[early],
    epochs = 15)
```

```
Epoch 1/15
84/84 [==============================] - 66s 774ms/step - loss: 0.3883 - accuracy: 0.8016 - val_loss: 0.3717 - val_accuracy: 0.8197
Epoch 2/15
84/84 [==============================] - 52s 625ms/step - loss: 0.4125 - accuracy: 0.7960 - val_loss: 0.3699 - val_accuracy: 0.8212
Epoch 3/15
84/84 [==============================] - 48s 564ms/step - loss: 0.3766 - accuracy: 0.8162 - val_loss: 0.3591 - val_accuracy: 0.8424
Epoch 4/15
84/84 [==============================] - 49s 586ms/step - loss: 0.3886 - accuracy: 0.8016 - val_loss: 0.3663 - val_accuracy: 0.8394
Epoch 5/15
84/84 [==============================] - 49s 584ms/step - loss: 0.3722 - accuracy: 0.8222 - val_loss: 0.3524 - val_accuracy: 0.8333
Epoch 6/15
84/84 [==============================] - 51s 605ms/step - loss: 0.3680 - accuracy: 0.8233 - val_loss: 0.3745 - val_accuracy: 0.8258
Epoch 7/15
84/84 [==============================] - 54s 633ms/step - loss: 0.3765 - accuracy: 0.8185 - val_loss: 0.3547 - val_accuracy: 0.8273
Epoch 8/15
84/84 [==============================] - 49s 584ms/step - loss: 0.3650 - accuracy: 0.8207 - val_loss: 0.3466 - val_accuracy: 0.8515
Epoch 9/15
84/84 [==============================] - 53s 624ms/step - loss: 0.3600 - accuracy: 0.8330 - val_loss: 0.3584 - val_accuracy: 0.8500
Epoch 10/15
84/84 [==============================] - 51s 600ms/step - loss: 0.3567 - accuracy: 0.8353 - val_loss: 0.3389 - val_accuracy: 0.8394
Epoch 11/15
84/84 [==============================] - 49s 579ms/step - loss: 0.3568 - accuracy: 0.8285 - val_loss: 0.3616 - val_accuracy: 0.8318
Epoch 12/15
84/84 [==============================] - 48s 574ms/step - loss: 0.3594 - accuracy: 0.8259 - val_loss: 0.3517 - val_accuracy: 0.8333
Epoch 13/15
84/84 [==============================] - 48s 567ms/step - loss: 0.3471 - accuracy: 0.8323 - val_loss: 0.3442 - val_accuracy: 0.8530
Epoch 14/15
84/84 [==============================] - 51s 608ms/step - loss: 0.3618 - accuracy: 0.8274 - val_loss: 0.3428 - val_accuracy: 0.8379
Epoch 15/15
84/84 [==============================] - 48s 571ms/step - loss: 0.3589 - accuracy: 0.8252 - val_loss: 0.3428 - val_accuracy: 0.8439
```

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

```
21/21 [==============================] - 6s 269ms/step - loss: 0.3428 - accuracy: 0.8439
[0.3427780270576477, 0.8439394235610962]
```

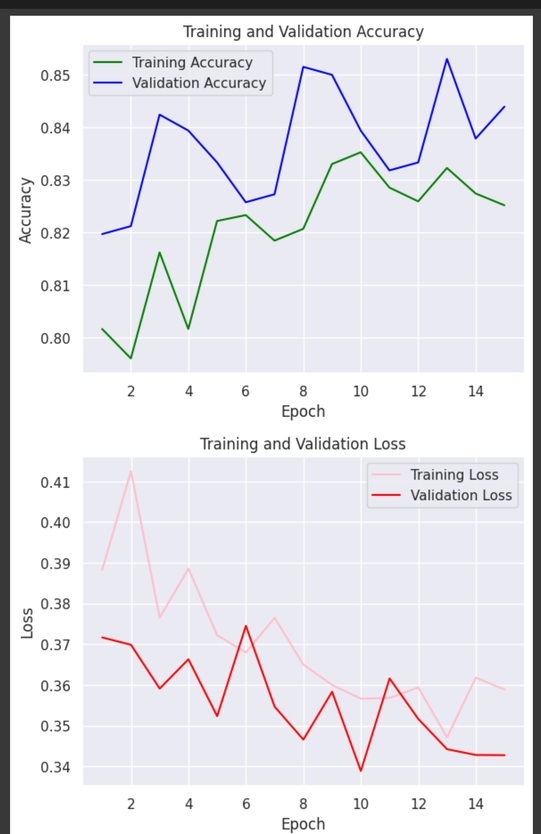
```
#plotting training values
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

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='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



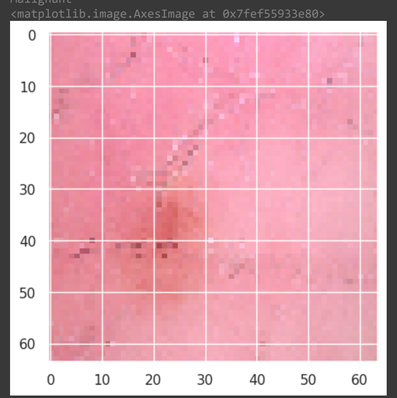

```
#predict val data
y_pred = model.predict(test_data)
y_pred =  np.argmax(y_pred,axis=1)

print(y_pred)
```

```
21/21 [==============================] - 7s 298ms/step
[1 0 1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 1 1
 0 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 1
 1 1 1 0 1 0 0 0 1 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 1 1 1 0 1 0 1 0 0 1 0 0 1
 0 0 1 1 1 0 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 0 0 0 0
 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 1 0 1 1 1 1 0 1 1 0 0 1 0 0 1 0
 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0
 0 0 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 1 1 1 1 0 0 0 1 1 1 1 0 1 1 1 0 0
 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1 1 1 0
 1 0 1 1 0 0 0 1 1 0 0 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0
 1 0 1 1 0 0 1 1 1 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1
 1 0 0 1 0 1 1 1 1 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 1 1 0 1 0 1
 0 0 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 0 0 1 0
 0 0 1 1 0 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 0 1
 0 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 0 1 0
 0 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 1 0 0 0 1 0 1 1
 0 1 0 0 1 0 1 1 0 0 1 1 0 0 0 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1 0 1 0 0 0 1
 0 0 1 0 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 1 0 0
 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 0 1 1 1 0]
```

```
#example 1
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/benign/1006.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(class_names[prediction[0]])
plt.imshow(new_img)
```

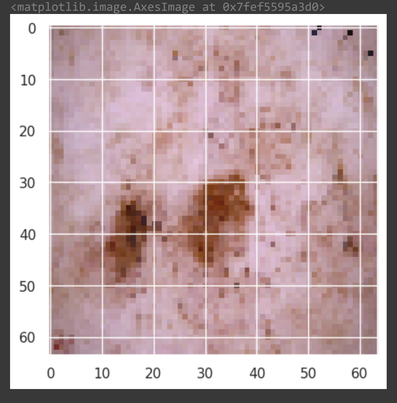
```
1/1 [==============================] - 0s 171ms/step
[1]
Malignant
<matplotlib.image.AxesImage at 0x7fef55933e80>
```



```
#example 2
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/train/malignant/1006.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)
```

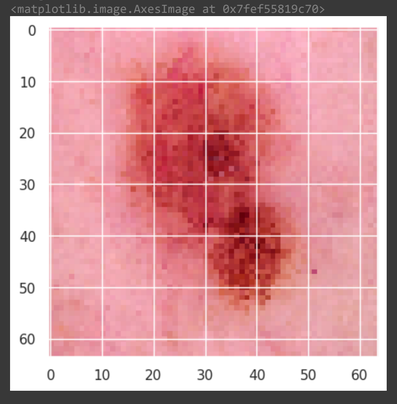
```
prediction = np.argmax(prediction,axis=1)
print(prediction)
print(class_names[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 27ms/step
[1]
Malignant
<matplotlib.image.AxesImage at 0x7fef5595a3d0>
```



```
#example 3
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/benign/1261.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(class_names[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 43ms/step
[0]
Benign
<matplotlib.image.AxesImage at 0x7fef55819c70>
```



```
#example 3=4
import keras.utils as image
import numpy as np
image_path = "/content/drive/MyDrive/data/test/malignant/1058.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(class_names[prediction[0]])
plt.imshow(new_img)
```

```
1/1 [==============================] - 0s 41ms/step
[1]
Malignant
<matplotlib.image.AxesImage at 0x7fef557504c0>
```