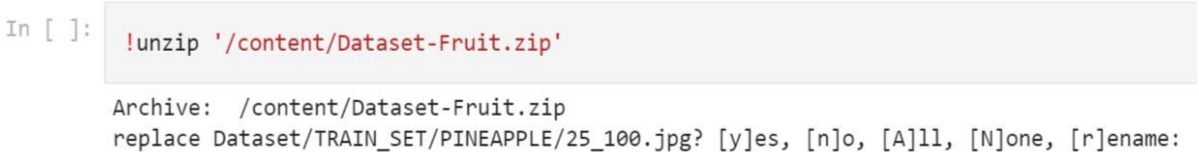
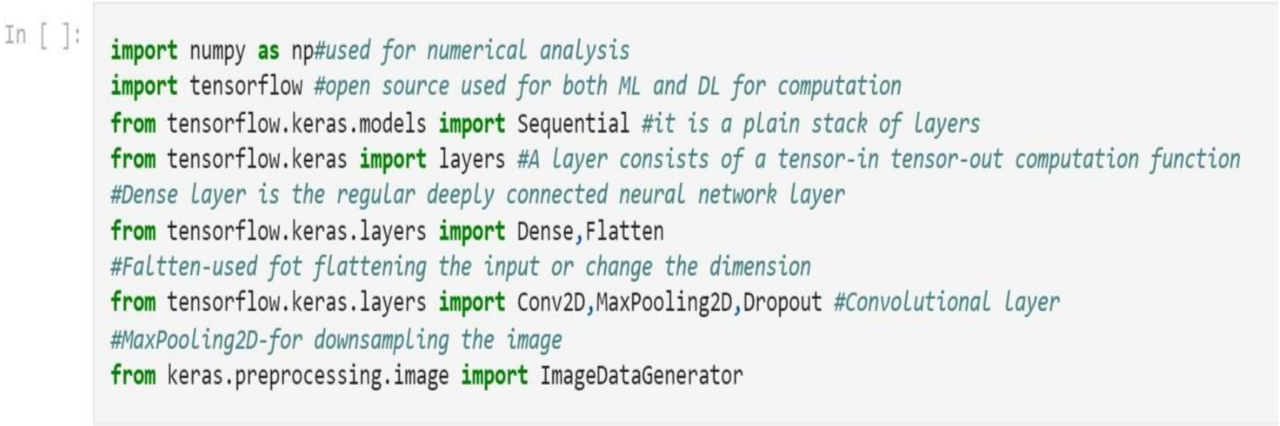
DATA COLLECTION AND PREPROCESSING

Nutrition image Analysis using CNN



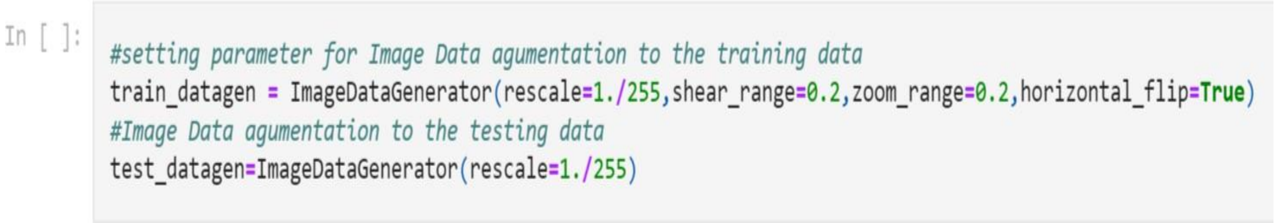
```
In [ ]: !unzip '/content/Dataset-Fruit.zip'

        Archive:  /content/Dataset-Fruit.zip
        replace Dataset/TRAIN_SET/PINEAPPLE/25_100.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

Importing Necessary Libraries



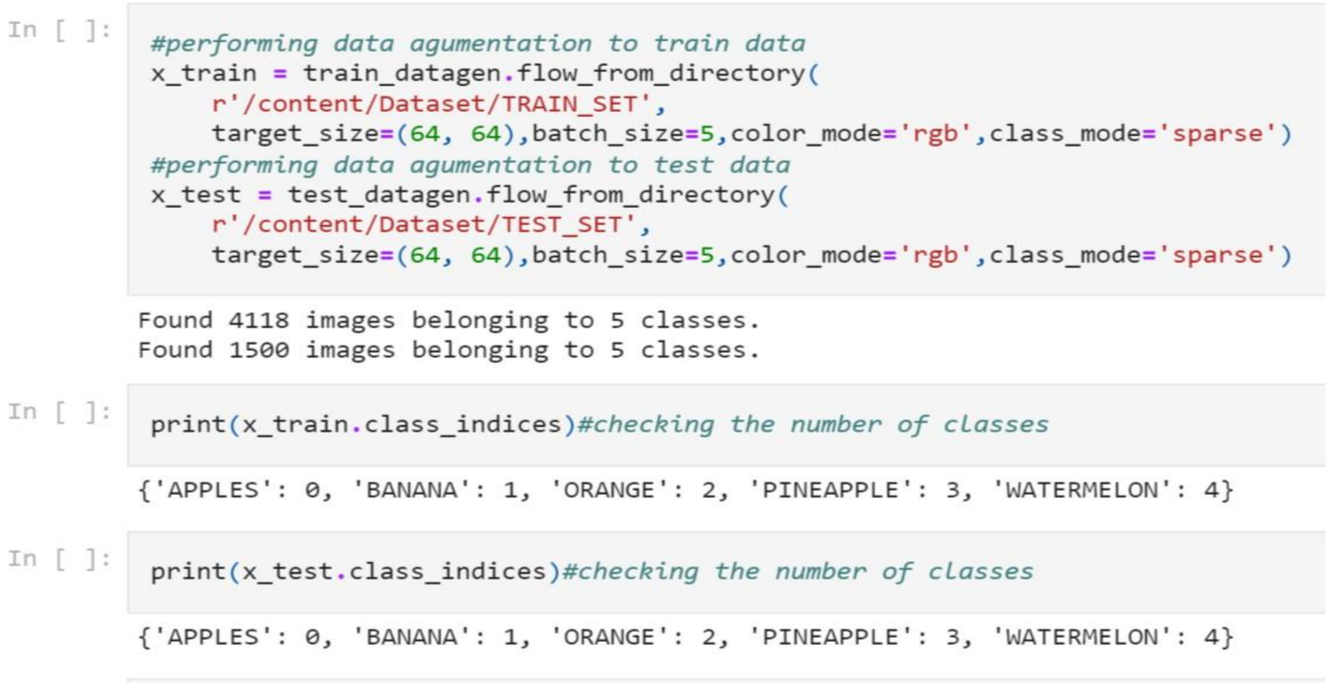
```
In [ ]: import numpy as np#used for numerical analysis
        import tensorflow #open source used for both ML and DL for computation
        from tensorflow.keras.models import Sequential #it is a plain stack of layers
        from tensorflow.keras import layers #A layer consists of a tensor-in tensor-out computation function
        #Dense layer is the regular deeply connected neural network layer
        from tensorflow.keras.layers import Dense,Flatten
        #Faltten-used fot flattening the input or change the dimension
        from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dropout #Convolutional layer
        #MaxPooling2D-for downsampling the image
        from keras.preprocessing.image import ImageDataGenerator
```

Image Data Augmentation



```
In [ ]: #setting parameter for Image Data agumentation to the training data
        train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
        #Image Data agumentation to the testing data
        test_datagen=ImageDataGenerator(rescale=1./255)
```

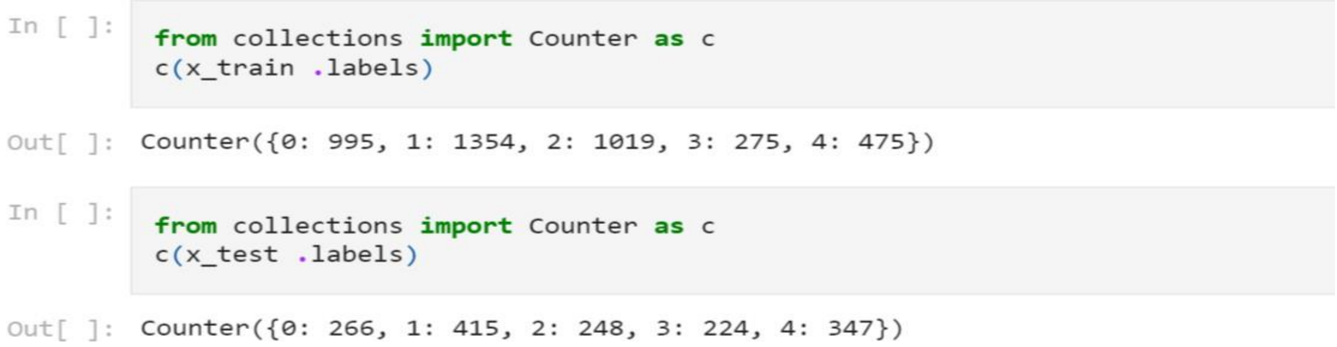
Loading our data and performing data augmentation



```
In [ ]:  #performing data agumentation to train data
         x_train = train_datagen.flow_from_directory(
             r'/content/Dataset/TRAIN_SET',
             target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
         #performing data agumentation to test data
         x_test = test_datagen.flow_from_directory(
             r'/content/Dataset/TEST_SET',
             target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')

         Found 4118 images belonging to 5 classes.
         Found 1500 images belonging to 5 classes.
```

```
In [ ]:  print(x_train.class_indices)#checking the number of classes

         {'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}
```

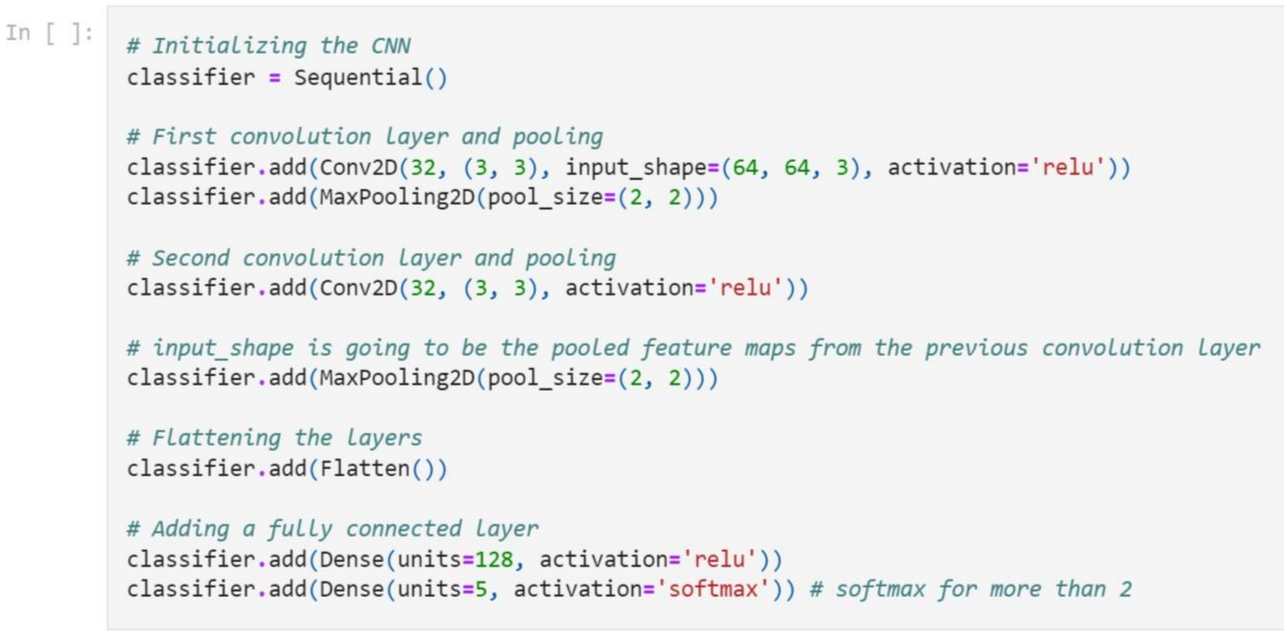
```
In [ ]:  print(x_test.class_indices)#checking the number of classes

         {'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}
```



```
In [ ]:  from collections import Counter as c
         c(x_train .labels)

Out[ ]:  Counter({0: 995, 1: 1354, 2: 1019, 3: 275, 4: 475})
```

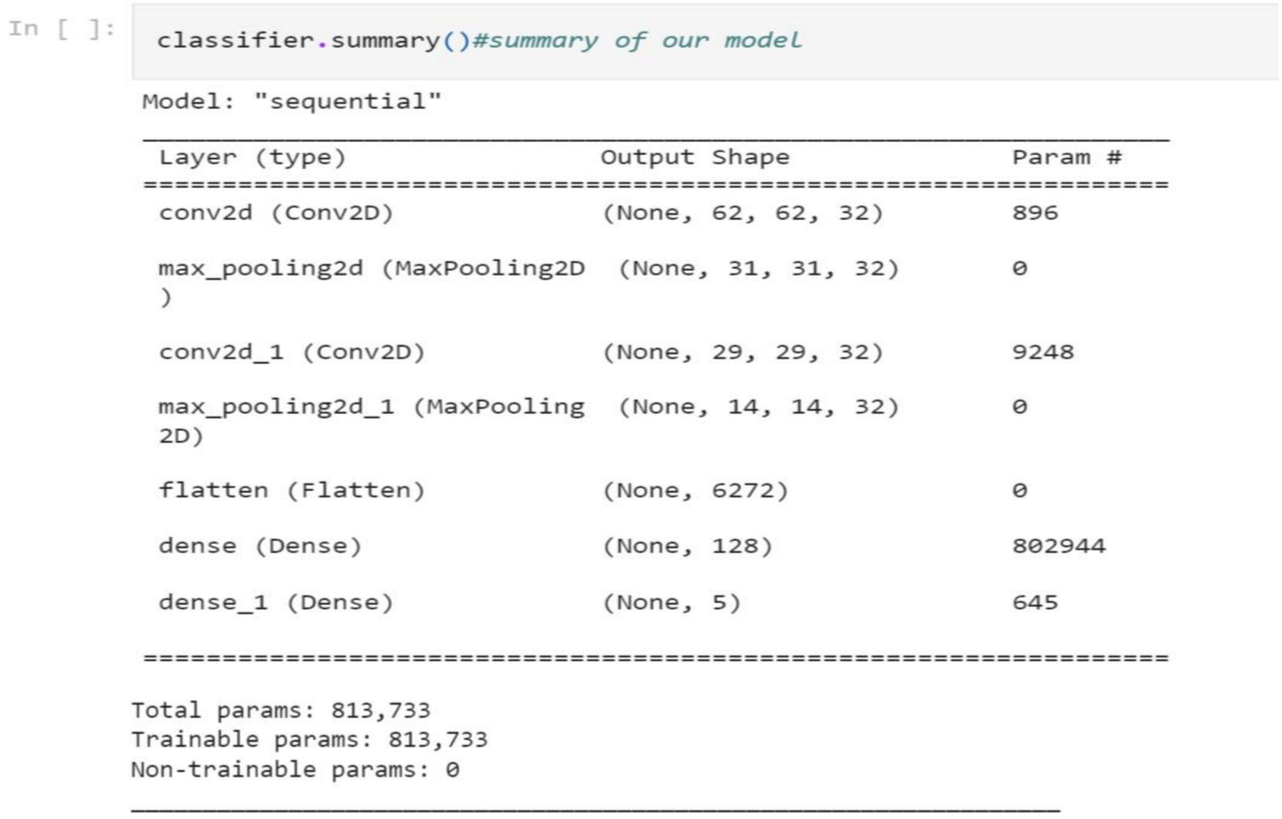
```
In [ ]:  from collections import Counter as c
         c(x_test .labels)

Out[ ]:  Counter({0: 266, 1: 415, 2: 248, 3: 224, 4: 347})
```

Creating the Model



In [ ]:

```
# Initializing the CNN
classifier = Sequential()

# First convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Second convolution layer and pooling
classifier.add(Conv2D(32, (3, 3), activation='relu'))

# input_shape is going to be the pooled feature maps from the previous convolution layer
classifier.add(MaxPooling2D(pool_size=(2, 2)))

# Flattening the layers
classifier.add(Flatten())

# Adding a fully connected layer
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax')) # softmax for more than 2
```

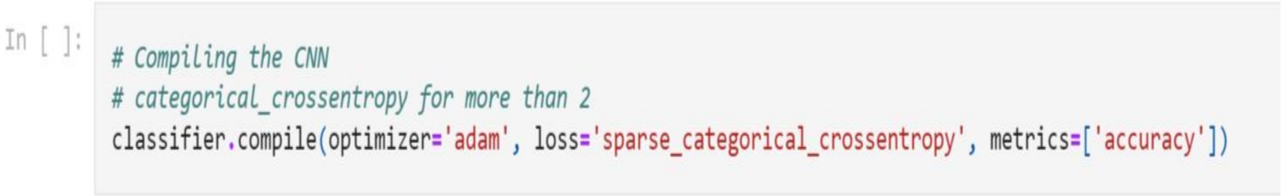


In [ ]:

```
classifier.summary()#summary of our model
```

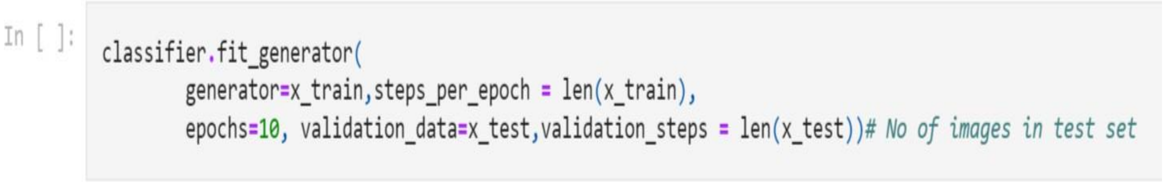
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 6272)              0         
                                                                 
 dense (Dense)               (None, 128)               802944    
                                                                 
 dense_1 (Dense)             (None, 5)                 645       
                                                                 
=================================================================
Total params: 813,733
Trainable params: 813,733
Non-trainable params: 0
_________________________________________________________________
```

Compiling The Model

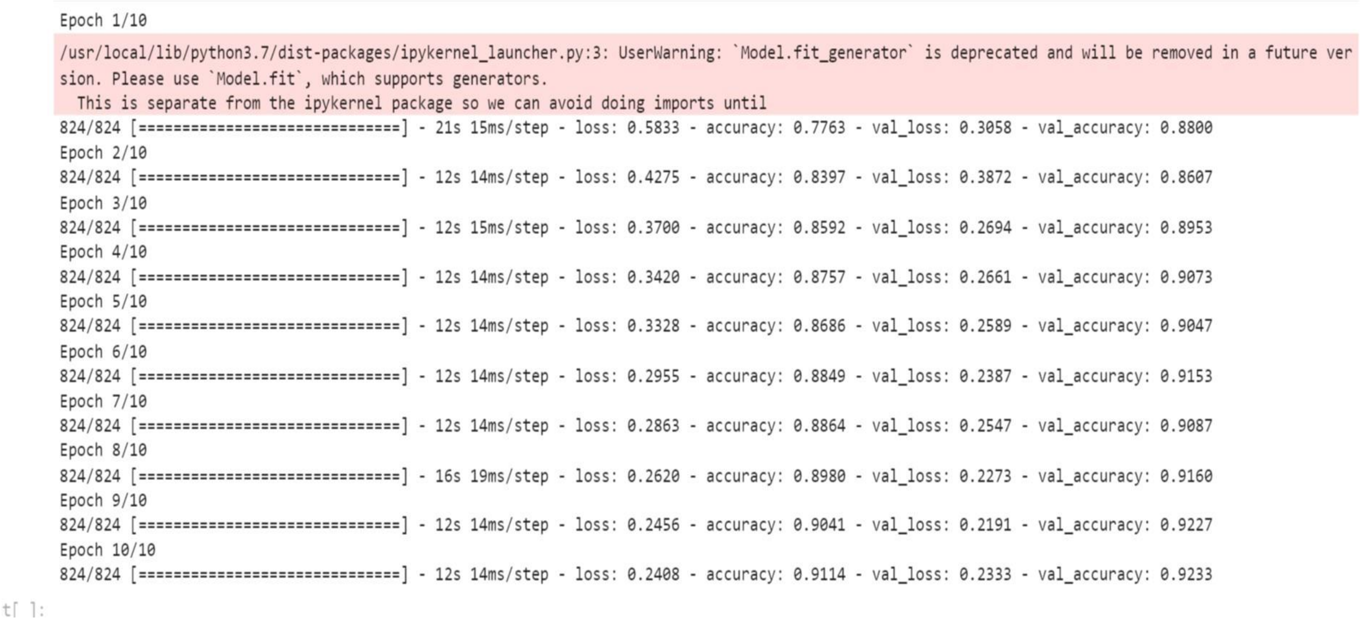


```
In [ ]:  # Compiling the CNN
         # categorical_crossentropy for more than 2
         classifier.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Fitting the Model

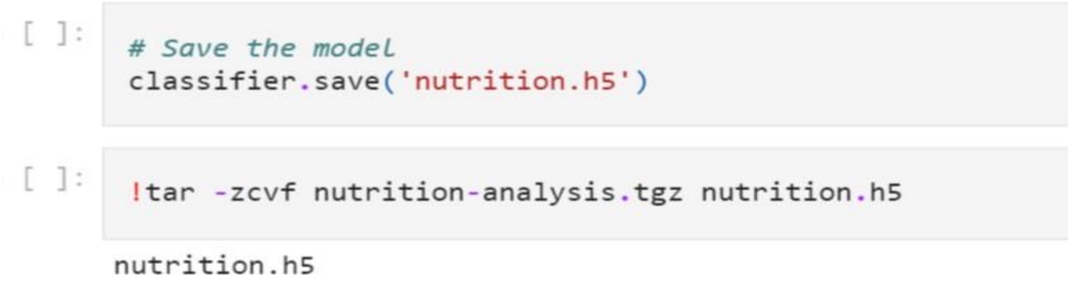


```
In [ ]:  classifier.fit_generator(
                 generator=x_train,steps_per_epoch = len(x_train),
                 epochs=10, validation_data=x_test,validation_steps = len(x_test))# No of images in test set
```



```
Epoch 1/10
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future ver
sion. Please use `Model.fit`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports until
824/824 [==============================] - 21s 15ms/step - loss: 0.5833 - accuracy: 0.7763 - val_loss: 0.3058 - val_accuracy: 0.8800
Epoch 2/10
824/824 [==============================] - 12s 14ms/step - loss: 0.4275 - accuracy: 0.8397 - val_loss: 0.3872 - val_accuracy: 0.8607
Epoch 3/10
824/824 [==============================] - 12s 15ms/step - loss: 0.3700 - accuracy: 0.8592 - val_loss: 0.2694 - val_accuracy: 0.8953
Epoch 4/10
824/824 [==============================] - 12s 14ms/step - loss: 0.3420 - accuracy: 0.8757 - val_loss: 0.2661 - val_accuracy: 0.9073
Epoch 5/10
824/824 [==============================] - 12s 14ms/step - loss: 0.3328 - accuracy: 0.8686 - val_loss: 0.2589 - val_accuracy: 0.9047
Epoch 6/10
824/824 [==============================] - 12s 14ms/step - loss: 0.2955 - accuracy: 0.8849 - val_loss: 0.2387 - val_accuracy: 0.9153
Epoch 7/10
824/824 [==============================] - 12s 14ms/step - loss: 0.2863 - accuracy: 0.8864 - val_loss: 0.2547 - val_accuracy: 0.9087
Epoch 8/10
824/824 [==============================] - 16s 19ms/step - loss: 0.2620 - accuracy: 0.8980 - val_loss: 0.2273 - val_accuracy: 0.9160
Epoch 9/10
824/824 [==============================] - 12s 14ms/step - loss: 0.2456 - accuracy: 0.9041 - val_loss: 0.2191 - val_accuracy: 0.9227
Epoch 10/10
824/824 [==============================] - 12s 14ms/step - loss: 0.2408 - accuracy: 0.9114 - val_loss: 0.2333 - val_accuracy: 0.9233
t[ ]:
```

Saving Our Model



```
[ ]: # Save the model
     classifier.save('nutrition.h5')
```

```
[ ]: !tar -zcvf nutrition-analysis.tgz nutrition.h5
```

```
nutrition.h5
```