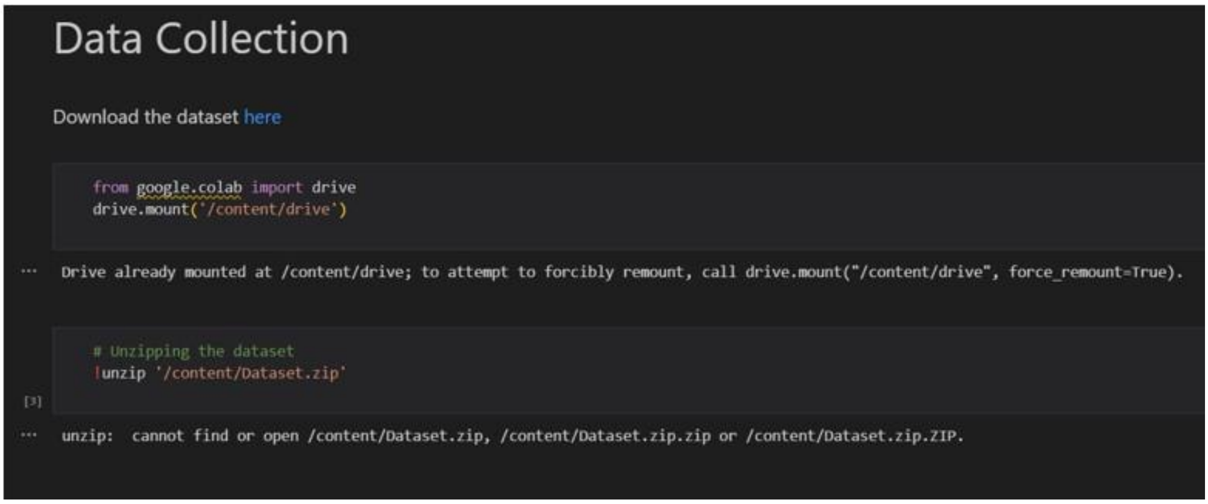
Coding and Solution



Data Collection

Download the dataset here

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Unzipping the dataset
!unzip '/content/Dataset.zip'
```

```
unzip:  cannot find or open /content/Dataset.zip, /content/Dataset.zip.zip or /content/Dataset.zip.ZIP.
```

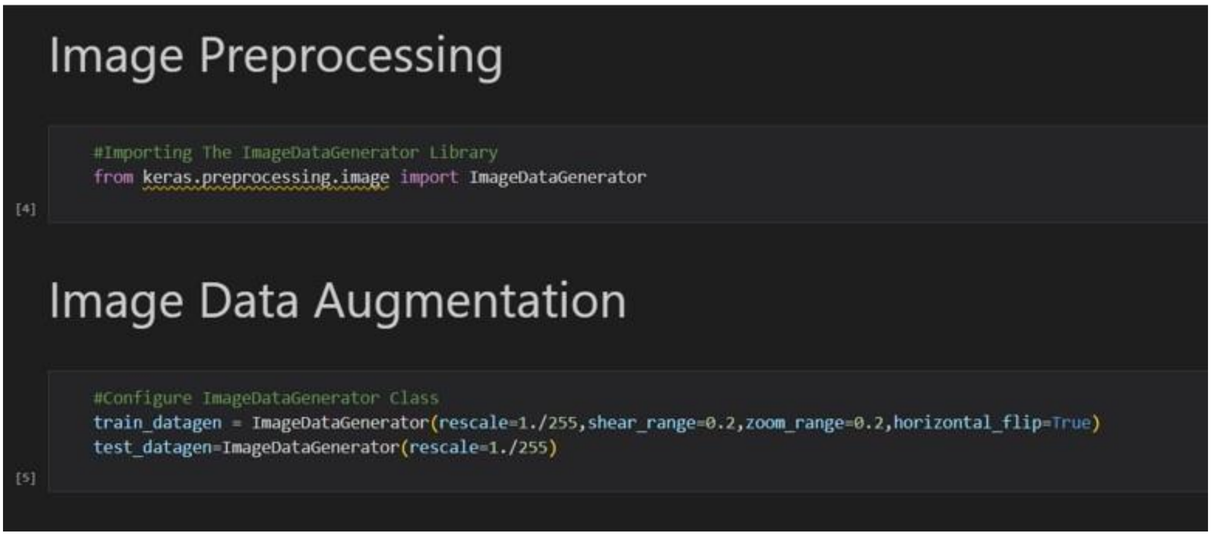
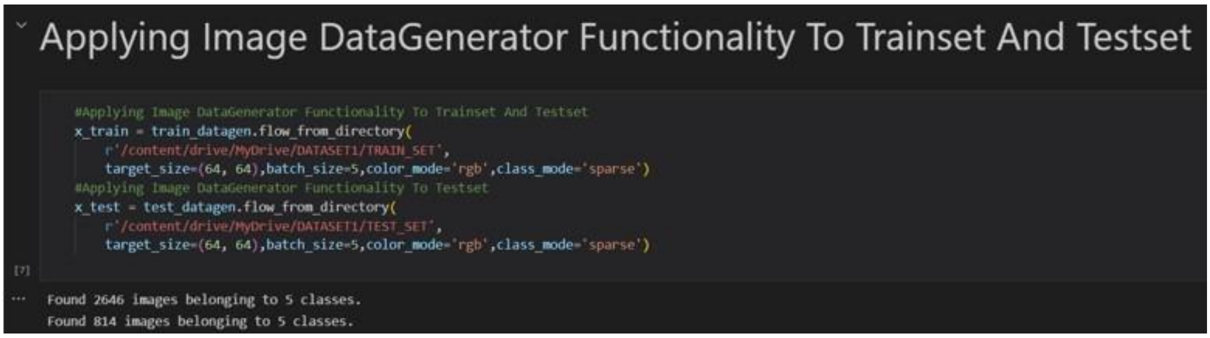


Image Preprocessing

```
#Importing The ImageDataGenerator Library
from keras.preprocessing.image import ImageDataGenerator
```

Image Data Augmentation

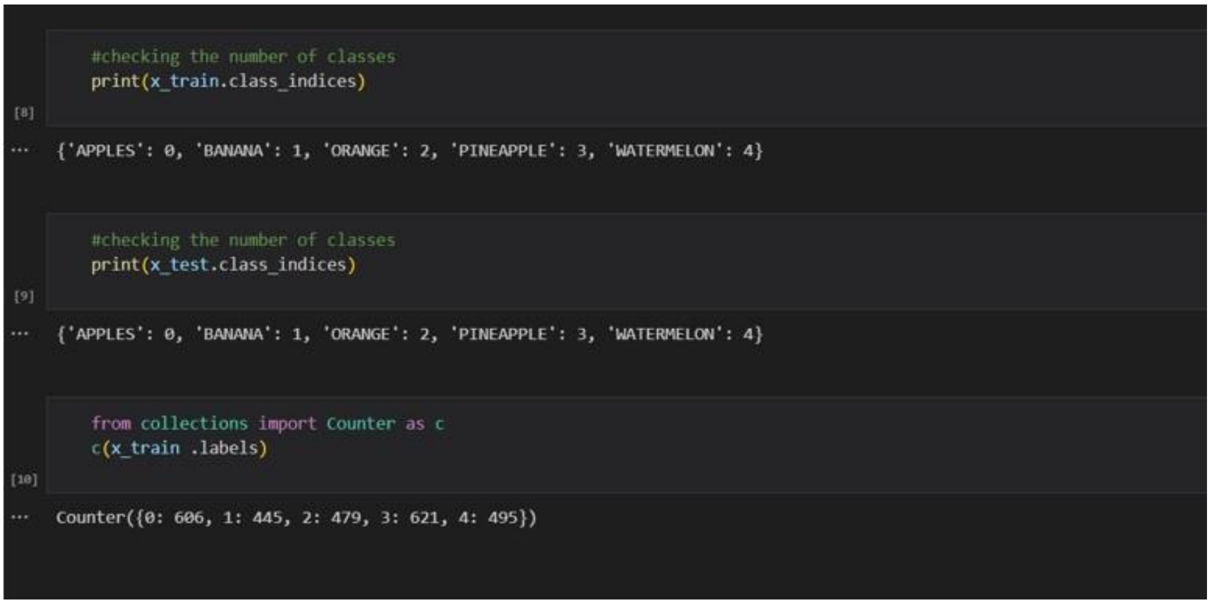
```
#Configure ImageDataGenerator Class
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)
```



Applying Image DataGenerator Functionality To Trainset And Testset

```
#Applying Image DataGenerator Functionality To Trainset And Testset
x_train = train_datagen.flow_from_directory(
    r'/content/drive/MyDrive/DATASET1/TRAIN_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
#Applying Image DataGenerator Functionality To Testset
x_test = test_datagen.flow_from_directory(
    r'/content/drive/MyDrive/DATASET1/TEST_SET',
    target_size=(64, 64),batch_size=5,color_mode='rgb',class_mode='sparse')
```

```
Found 2646 images belonging to 5 classes.
Found 814 images belonging to 5 classes.
```



```
#checking the number of classes
print(x_train.class_indices)
```
[8]

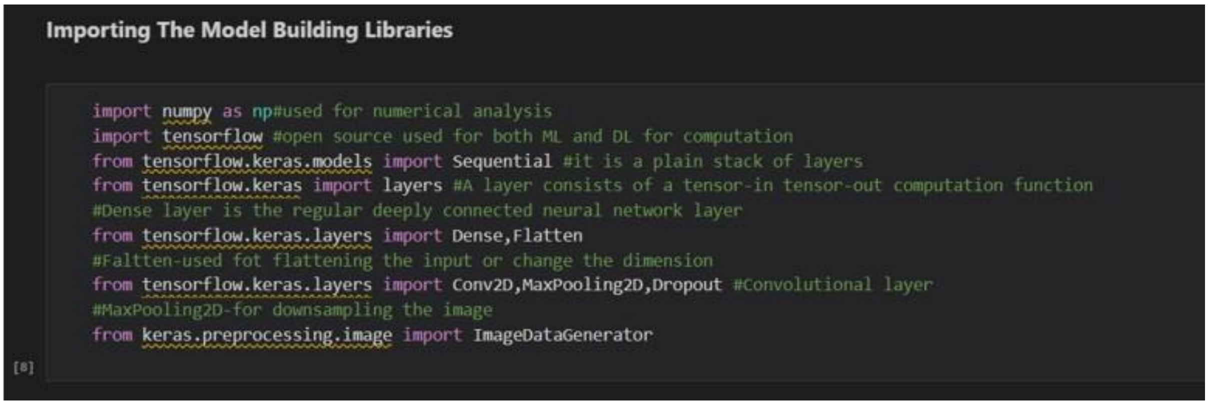
```
{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2, 'PINEAPPLE': 3, 'WATERMELON': 4}
```

```
#checking the number of classes
print(x_test.class_indices)
```
[9]

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

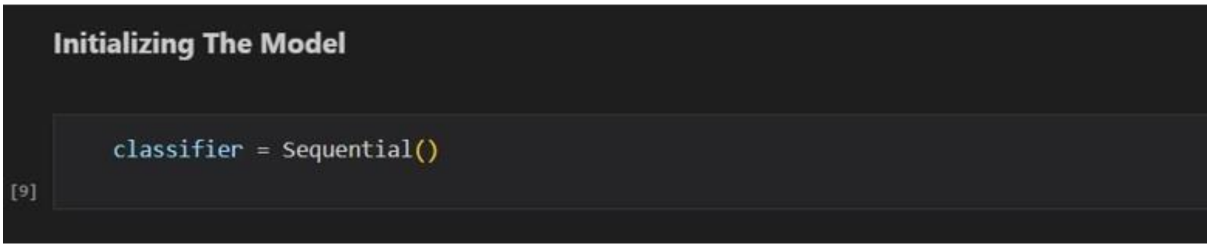
```
from collections import Counter as c
c(x_train .labels)
```
[10]

```
Counter({0: 606, 1: 445, 2: 479, 3: 621, 4: 495})
```



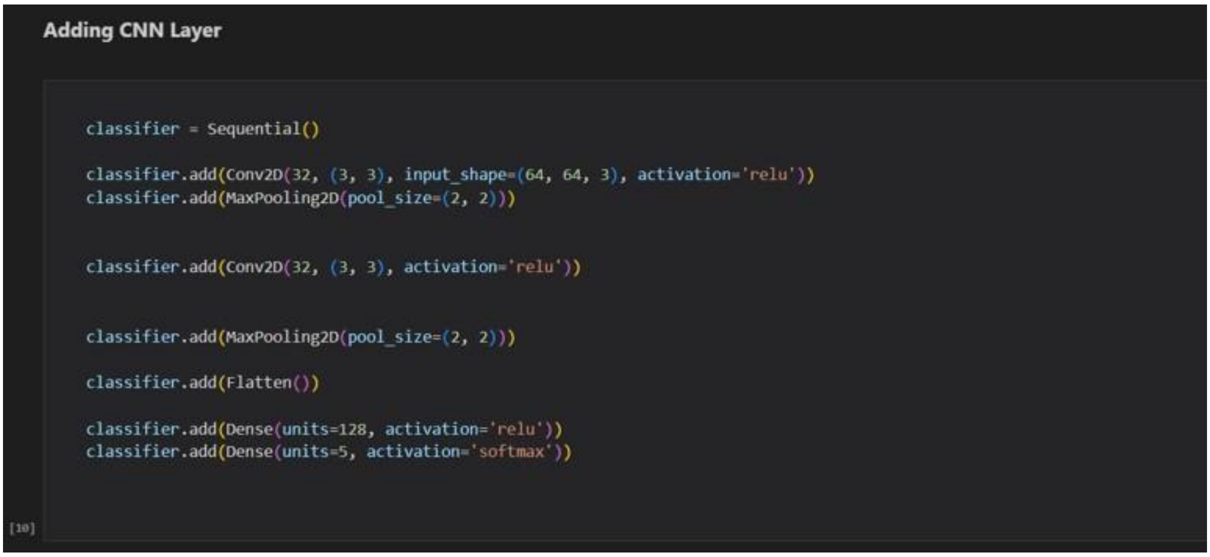
Importing The Model Building Libraries

```
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
```
[8]



Initializing The Model

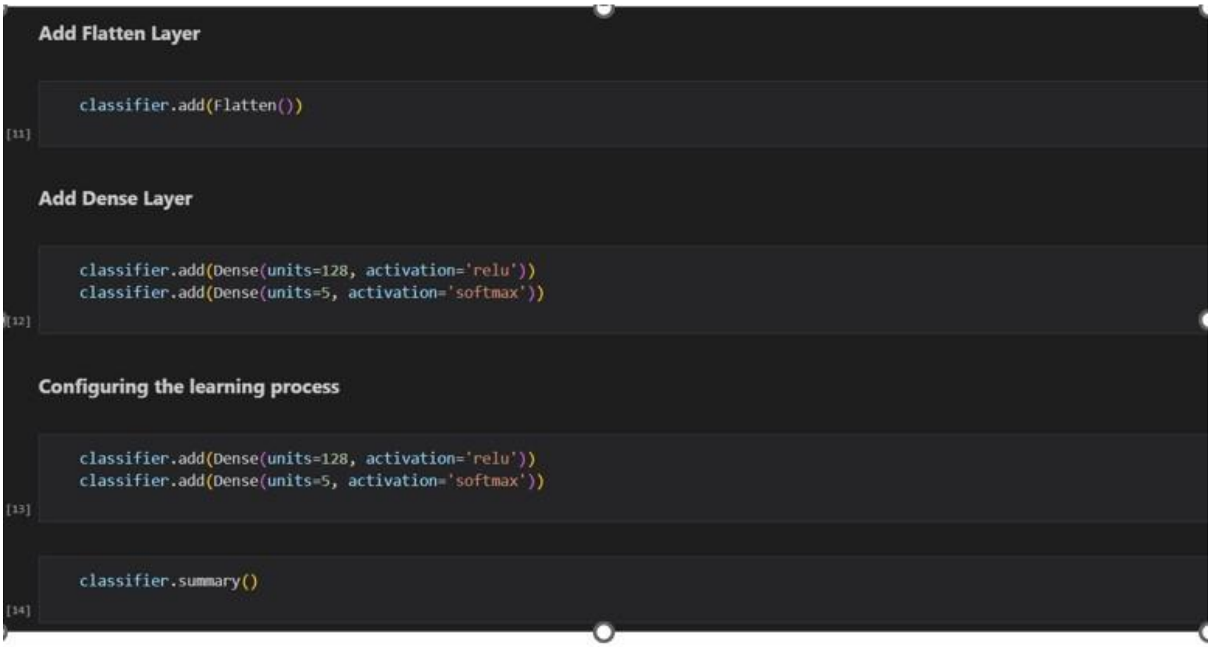
```
classifier = Sequential()
```
[9]



Adding CNN Layer

```
classifier = Sequential()

classifier.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
classifier.add(MaxPooling2D(pool_size=(2, 2)))


classifier.add(Conv2D(32, (3, 3), activation='relu'))


classifier.add(MaxPooling2D(pool_size=(2, 2)))

classifier.add(Flatten())

classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax'))
```

[10]



Add Flatten Layer

```
classifier.add(Flatten())
```

[11]

Add Dense Layer

```
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax'))
```

[12]

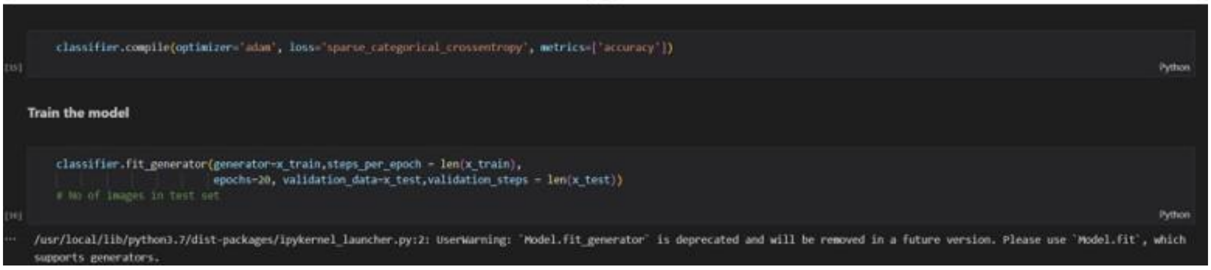
Configuring the learning process

```
classifier.add(Dense(units=128, activation='relu'))
classifier.add(Dense(units=5, activation='softmax'))
```

[13]

```
classifier.summary()
```

[14]



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

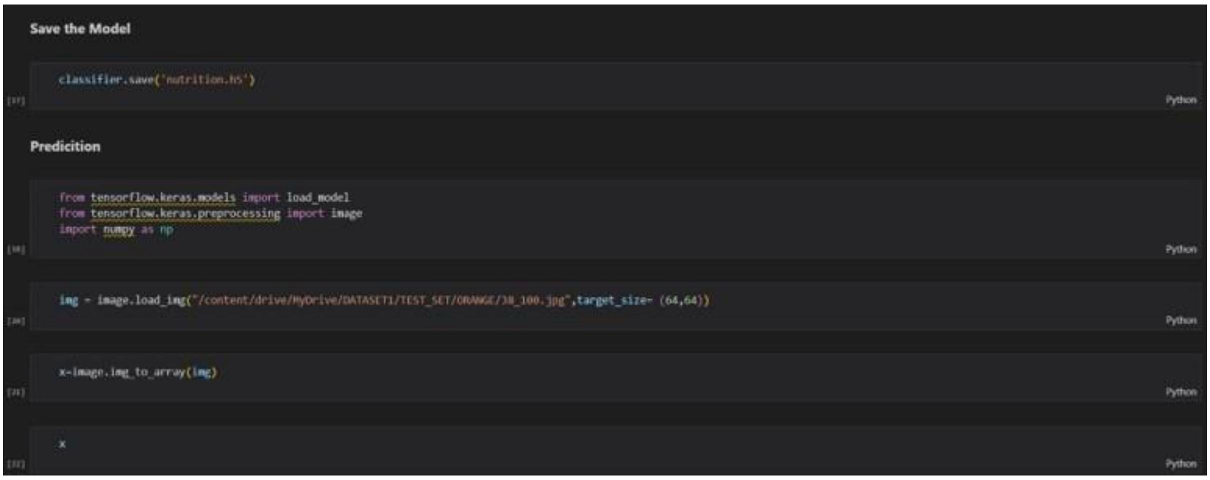
[15] Python

Train the model

```
classifier.fit_generator(generator=x_train,steps_per_epoch = len(x_train),
                         epochs=20, validation_data=x_test,validation_steps = len(x_test))
# No of images in test set
```

[16] Python

```
... /usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future version. Please use `Model.fit`, which supports generators.
```



Save the Model

```
classifier.save('nutrition.h5')
```

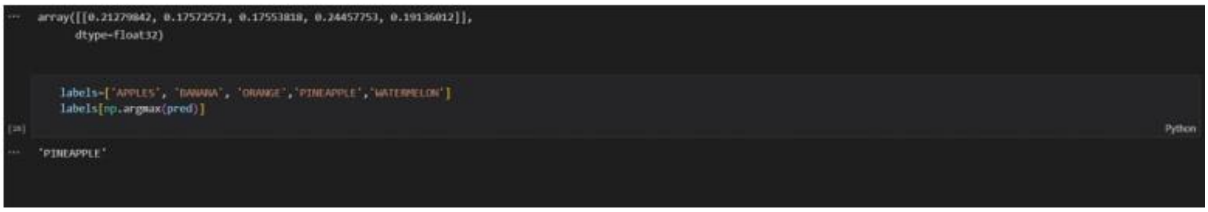
Predicition

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
img = image.load_img("/content/drive/MyDrive/DATASET1/TEST_SET/ORANGE/38_100.jpg",target_size= (64,64))
```

```
x=image.img_to_array(img)
```

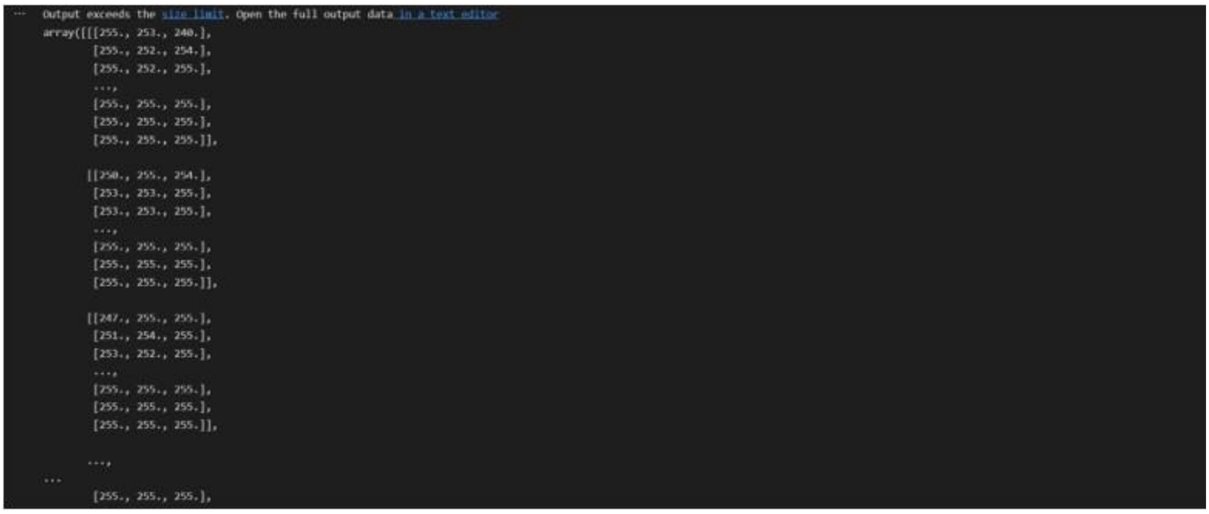
```
x
```



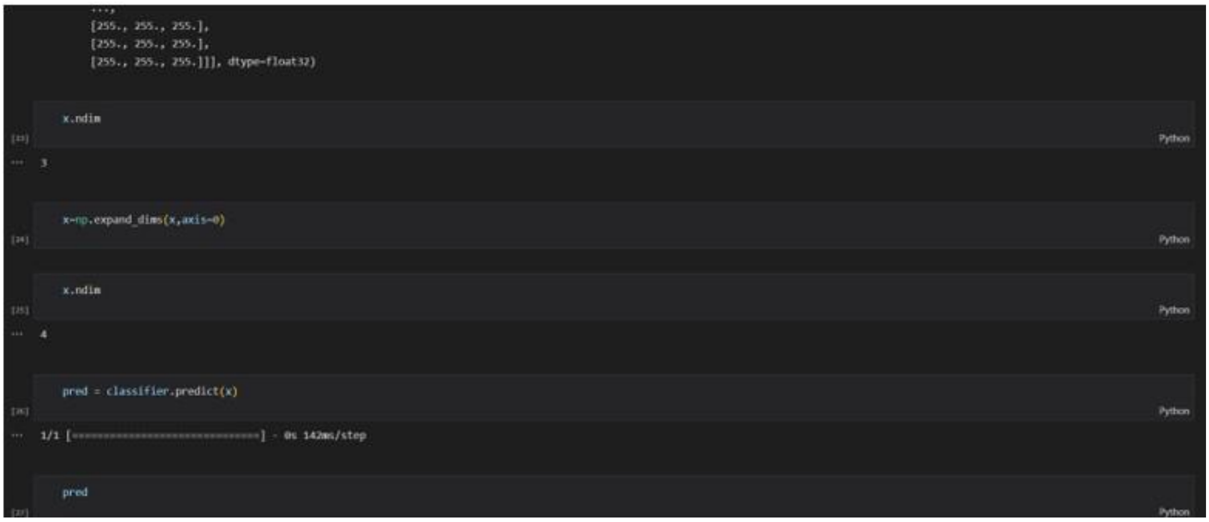
```
array([[0.21279842, 0.17572571, 0.17553818, 0.24457753, 0.19136012]],
      dtype=float32)
```

```
labels=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
labels[np.argmax(pred)]
```

```
'PINEAPPLE'
```



```
Output exceeds the size limit. Open the full output data in a text editor
array([[[[255., 253., 240.],
         [255., 252., 254.],
         [255., 252., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[250., 255., 254.],
         [253., 253., 255.],
         [253., 253., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        [[247., 255., 255.],
         [251., 254., 255.],
         [253., 252., 255.],
         ...,
         [255., 255., 255.],
         [255., 255., 255.],
         [255., 255., 255.]],

        ...,
...
         [255., 255., 255.],
```



```
        ...,
        [255., 255., 255.],
        [255., 255., 255.],
        [255., 255., 255.]]], dtype=float32)
```

```
x.ndim
```

```
3
```

```
x=np.expand_dims(x,axis=0)
```

```
x.ndim
```

```
4
```

```
pred = classifier.predict(x)
```

```
1/1 [==============================] - 0s 142ms/step
```

```
pred
```