In [ ]:
```
#import keras libraries
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Convolution2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
```

In [ ]:
```
#image preprocessing(or) image augmentation
from keras.preprocessing.image import ImageDataGenerator
```

In [ ]:
```
train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True
#rescale => rescaling pixel value from 0 to 255 to 0 to 1
#shear_range=> counter clock wise rotation(anti clock)
```

In [ ]:
```
test_datagen = ImageDataGenerator(rescale=1./255)
```

In [ ]:
```
x_train = train_datagen.flow_from_directory("/content/drive/MyDrive/ibm project/TRAIN_SET",target_siz
```

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

In [ ]:
```
x_test = test_datagen.flow_from_directory("/content/drive/MyDrive/ibm project/TEST_SET",target_size=(
```

```
Found 929 images belonging to 3 classes.
```

In [ ]:
```
x_train.class_indices
```

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

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

```
{'APPLES': 0, 'BANANA': 1, 'ORANGE': 2}
```

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

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

In [ ]:
```
#Initializing the model
model = Sequential()
```

In [ ]:
```
# add First convolution layer
```

In [ ]:
```
model.add(Convolution2D(32,(3,3),input_shape=(64,64,3),activation="relu"))
# 32 indicates => no of feature detectors
#(3,3)=> kernel size (feature detector size)
```

In [ ]:
```
# add Maxpooling layer
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
#Second convolution layer and pooling
model.add(Convolution2D(32,(3,3),activation='relu'))
```

In [ ]:
```
model.add(MaxPooling2D(pool_size=(2,2)))
```

In [ ]:
```
#Flattening the layers
model.add(Flatten())
```

In [ ]:
```
model.add(Dense(units=128,activation='relu'))
```

In [ ]:
```
model.add(Dense(units=5,activation='softmax'))
```

In [ ]:
```
# add flatten layer => input to your ANN
```

In [ ]:
```
model.add(Flatten())
```

In [ ]:
```
model.summary()
```

```
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

flatten_1 (Flatten)          (None, 5)                 0
```

```
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

flatten_1 (Flatten)          (None, 5)                 0

=================================================================
Total params: 813,733
Trainable params: 813,733
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
# adding dense layer
```

In [ ]:
```
#hidden layer
```

In [ ]:
```
model.add(Dense(units=300,kernel_initializer="random_uniform",activation="relu"))
```

In [ ]:
```
model.add(Dense(units=200,kernel_initializer="random_uniform",activation="relu"))
```

In [ ]:
```
#output layer
```

In [ ]:
```
model.add(Dense(units=4,kernel_initializer="random_uniform",activation="softmax"))
len(x_train)
```

Out[ ]:
```
129
```

In [ ]:
```
#Ann starts so need to add dense layers
```

In [ ]:
```
model.add(Dense(units=128,activation="relu",kernel_initializer="random_uniform"))
```

In [ ]:
```
model.add(Dense(units=1,activation="sigmoid",kernel_initializer="random_uniform"))
```

In [ ]:
```
#Compile the model
model.compile(loss="binary_crossentropy",optimizer="adam",metrics=['accuracy'])
```

In [ ]:
```
#Train the model
```

In [ ]:
```
model.fit_generator(x_train,steps_per_epoch=len(x_train), validation_data=x_test, validation_steps=le
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use `Model.fit`, which supports generator
s.
  """Entry point for launching an IPython kernel.
Epoch 1/20
129/129 [==============================] - 2459s 19s/step - loss: -0.0526 - accuracy: 0.3273 - val_lo
ss: 0.1126 - val_accuracy: 0.4467
Epoch 2/20
129/129 [==============================] - 36s 277ms/step - loss: -3.0746 - accuracy: 0.3288 - val_lo
ss: 0.2155 - val_accuracy: 0.4467
Epoch 3/20
129/129 [==============================] - 35s 268ms/step - loss: -8.7866 - accuracy: 0.3288 - val_lo
ss: 0.5095 - val_accuracy: 0.4467
Epoch 4/20
129/129 [==============================] - 36s 281ms/step - loss: -17.7107 - accuracy: 0.3288 - val_l
oss: 0.9337 - val_accuracy: 0.4467
Epoch 5/20
129/129 [==============================] - 36s 282ms/step - loss: -29.8704 - accuracy: 0.3288 - val_l
oss: 1.4811 - val_accuracy: 0.4467
Epoch 6/20
129/129 [==============================] - 36s 277ms/step - loss: -45.0273 - accuracy: 0.3288 - val_l
oss: 2.1422 - val_accuracy: 0.4467
Epoch 7/20
129/129 [==============================] - 35s 269ms/step - loss: -62.9152 - accuracy: 0.3288 - val_l
oss: 2.9106 - val_accuracy: 0.4467
Epoch 8/20
129/129 [==============================] - 40s 309ms/step - loss: -83.5868 - accuracy: 0.3288 - val_l
oss: 3.7855 - val_accuracy: 0.4467
Epoch 9/20
129/129 [==============================] - 36s 281ms/step - loss: -106.7443 - accuracy: 0.3288 - val_
loss: 4.7640 - val_accuracy: 0.4467
Epoch 10/20
129/129 [==============================] - 36s 278ms/step - loss: -132.3641 - accuracy: 0.3288 - val_
loss: 5.8398 - val_accuracy: 0.4467
Epoch 11/20
129/129 [==============================] - 35s 271ms/step - loss: -160.3758 - accuracy: 0.3288 - val_
loss: 7.0081 - val_accuracy: 0.4467
Epoch 12/20
129/129 [==============================] - 35s 269ms/step - loss: -190.6966 - accuracy: 0.3288 - val_
loss: 8.2454 - val_accuracy: 0.4467
Epoch 13/20
129/129 [==============================] - 36s 279ms/step - loss: -223.1146 - accuracy: 0.3288 - val_
loss: 9.6145 - val_accuracy: 0.4467
Epoch 14/20
129/129 [==============================] - 36s 280ms/step - loss: -257.9082 - accuracy: 0.3288 - val_
loss: 11.0088 - val_accuracy: 0.4467
Epoch 15/20
129/129 [==============================] - 37s 290ms/step - loss: -294.5687 - accuracy: 0.3288 - val_
loss: 12.5175 - val_accuracy: 0.4467
Epoch 16/20
129/129 [==============================] - 34s 266ms/step - loss: -333.2441 - accuracy: 0.3288 - val_
loss: 14.1130 - val_accuracy: 0.4467
Epoch 17/20
129/129 [==============================] - 36s 279ms/step - loss: -374.0325 - accuracy: 0.3288 - val_
loss: 15.7641 - val_accuracy: 0.4467
Epoch 18/20
129/129 [==============================] - 36s 278ms/step - loss: -416.7053 - accuracy: 0.3288 - val_
loss: 17.5287 - val_accuracy: 0.4467
Epoch 19/20
```

```
129/129 [==============================] - 35s 267ms/step - loss: -461.2285 - accuracy: 0.3288 - val_
loss: 19.3238 - val_accuracy: 0.4467
Epoch 20/20
129/129 [==============================] - 34s 265ms/step - loss: -507.5266 - accuracy: 0.3288 - val_
loss: 21.2192 - val_accuracy: 0.4467
```

Out[ ]:

In [ ]:
```
model.save("nutrition.h5")
```

In [ ]:
```
#Prediction the result
```

In [ ]:
```
from tensorflow.keras.models import load_model
from keras.preprocessing  import image
model =load_model("nutrition.h5")
```

In [ ]:
```
import numpy as np
```

In [ ]:
```
from tensorflow.keras.utils import load_img
from tensorflow.keras.utils import img_to_array
#loading of the image
img = load_img(r'/content/drive/MyDrive/ibm project/Sample_Images-20221102T071233Z-001/Sample_Images/
#image to array
x = img_to_array(img)
#changing the shape
x= np.expand_dims(x,axis = 0)
predict_x=model.predict(x)
classes_x=np.argmax(predict_x,axis = -1)
classes_x
```

```
1/1 [==============================] - 0s 166ms/step
```

Out[ ]:
```
array([0])
```

In [ ]:
```
index=['APPLES', 'BANANA', 'ORANGE', 'PINEAPPLE', 'WATERMELON']
result=str(index[classes_x[0]])
result
```

Out[ ]:
```
'APPLES'
```