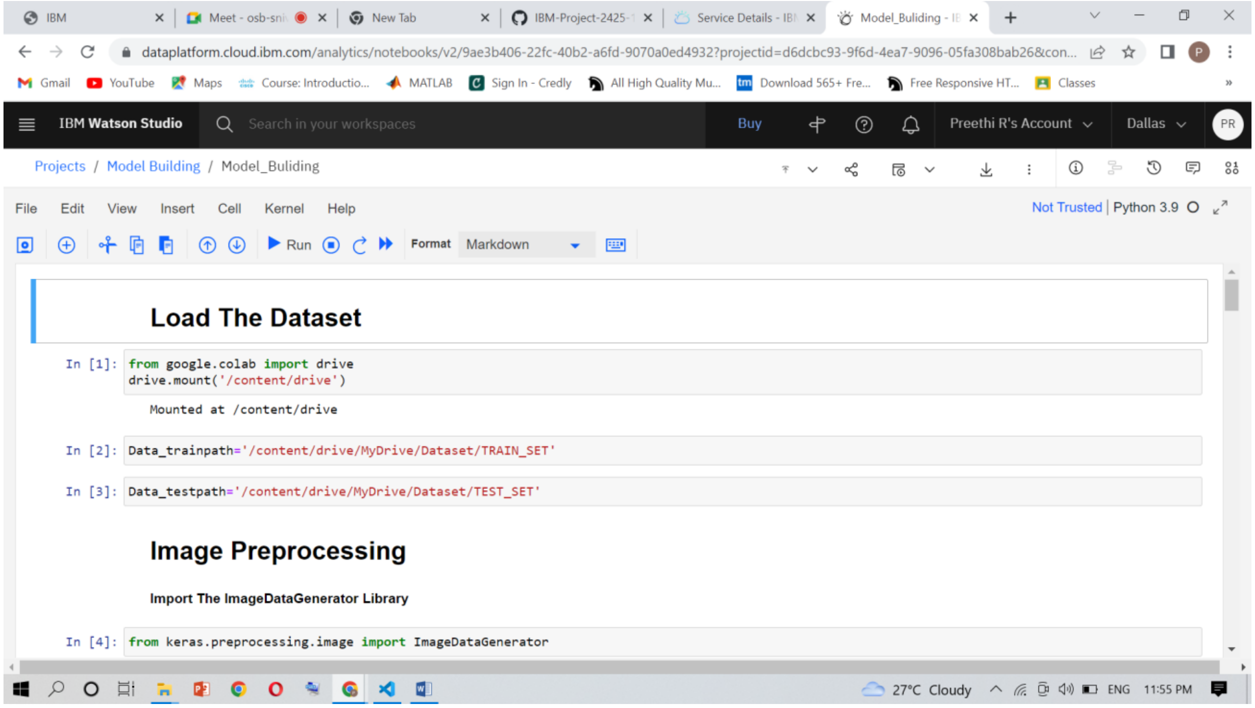
TRAIN MODEL ON IBM

DATE	22 November 2022
TEAM ID	PNT2022TMID14641
PROJECT NAME	AI-powered Nutrition Analyzer for Fitness Enthusiasts



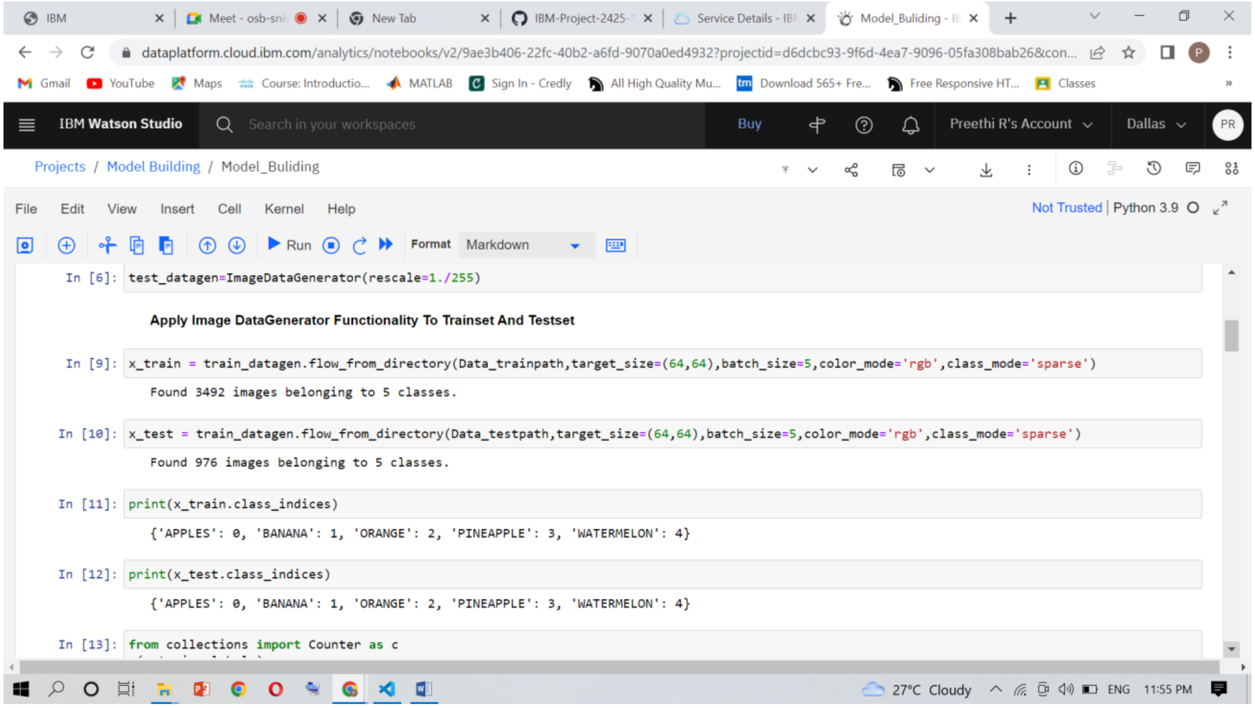
Load The Dataset

```
In [1]: from google.colab import drive
        drive.mount('/content/drive')

        Mounted at /content/drive

In [2]: Data_trainpath='/content/drive/MyDrive/Dataset/TRAIN_SET'

In [3]: Data_testpath='/content/drive/MyDrive/Dataset/TEST_SET'
```

Image Preprocessing

Import The ImageDataGenerator Library

```
In [4]: from keras.preprocessing.image import ImageDataGenerator
```



```
In [6]: test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply Image DataGenerator Functionality To Trainset And Testset

```
In [9]: x_train = train_datagen.flow_from_directory(Data_trainpath,target_size=(64,64),batch_size=5,color_mode='rgb',class_mode='sparse')

        Found 3492 images belonging to 5 classes.

In [10]: x_test = train_datagen.flow_from_directory(Data_testpath,target_size=(64,64),batch_size=5,color_mode='rgb',class_mode='sparse')

        Found 976 images belonging to 5 classes.

In [11]: print(x_train.class_indices)

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

In [12]: print(x_test.class_indices)

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

In [13]: from collections import Counter as c
```

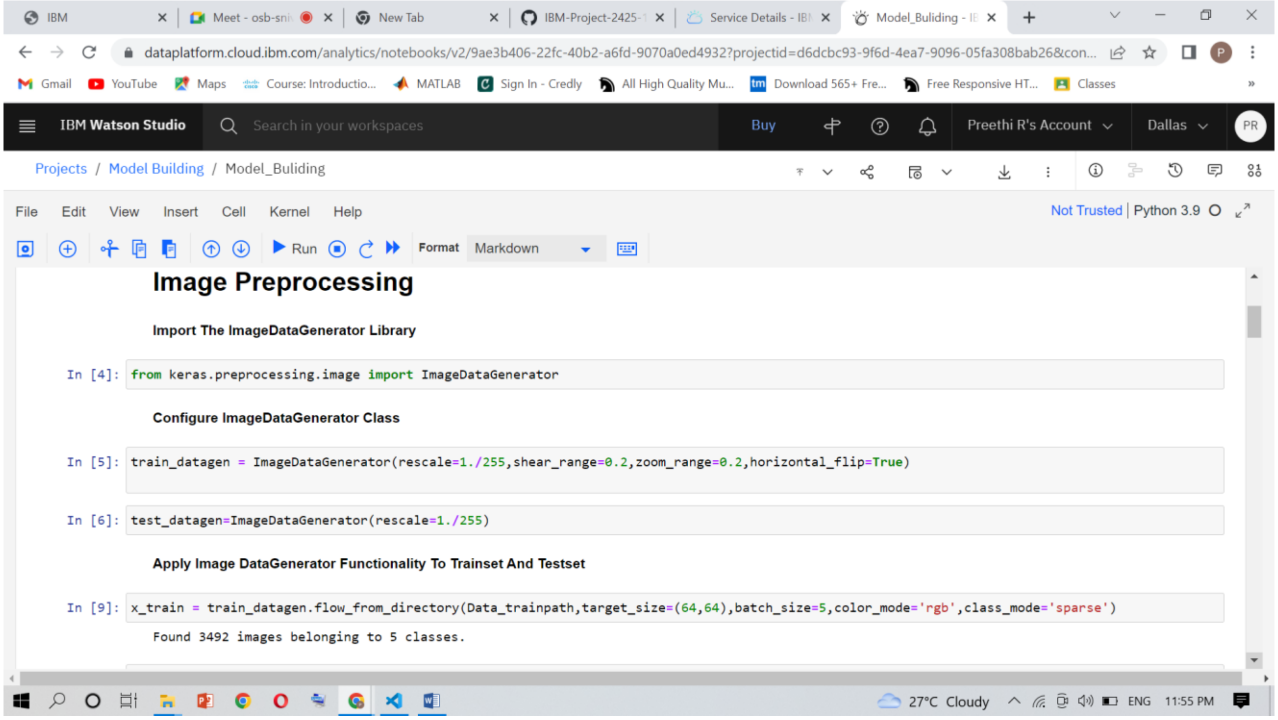


Image Preprocessing

Import The ImageDataGenerator Library

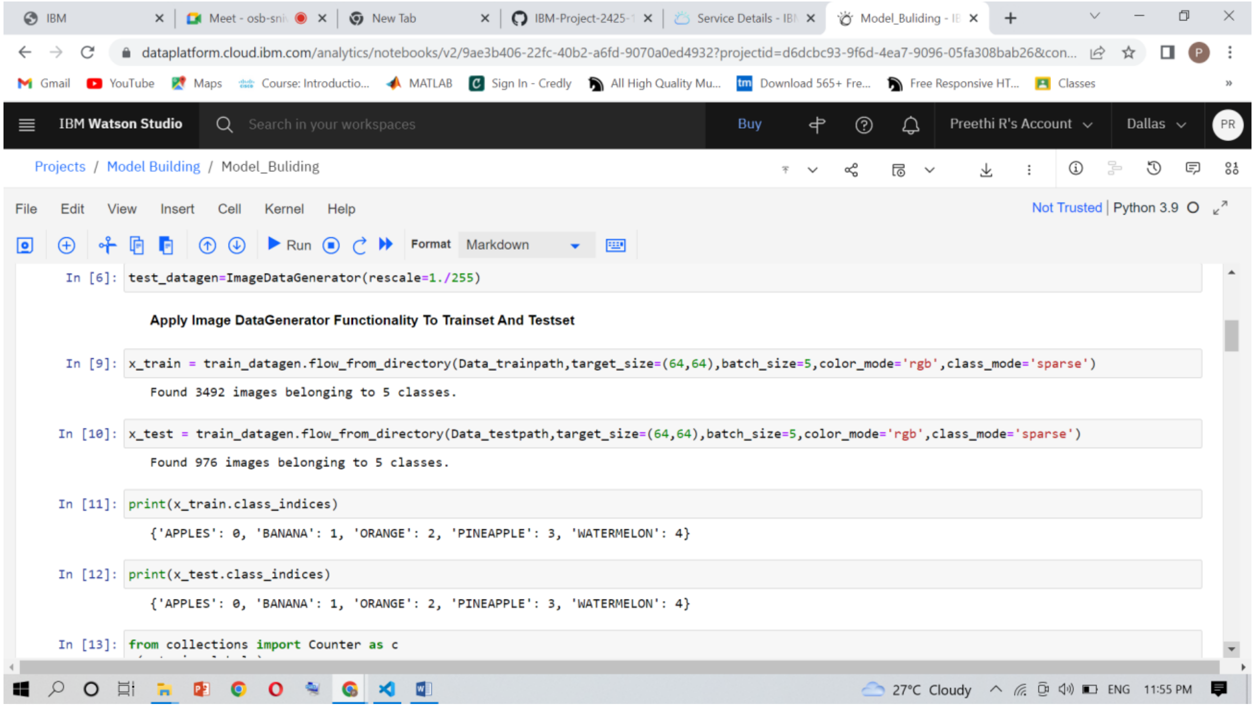
```
In [4]: from keras.preprocessing.image import ImageDataGenerator
```

Configure ImageDataGenerator Class

```
In [5]: train_datagen = ImageDataGenerator(rescale=1./255,shear_range=0.2,zoom_range=0.2,horizontal_flip=True)

In [6]: test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply Image DataGenerator Functionality To Trainset And Testset

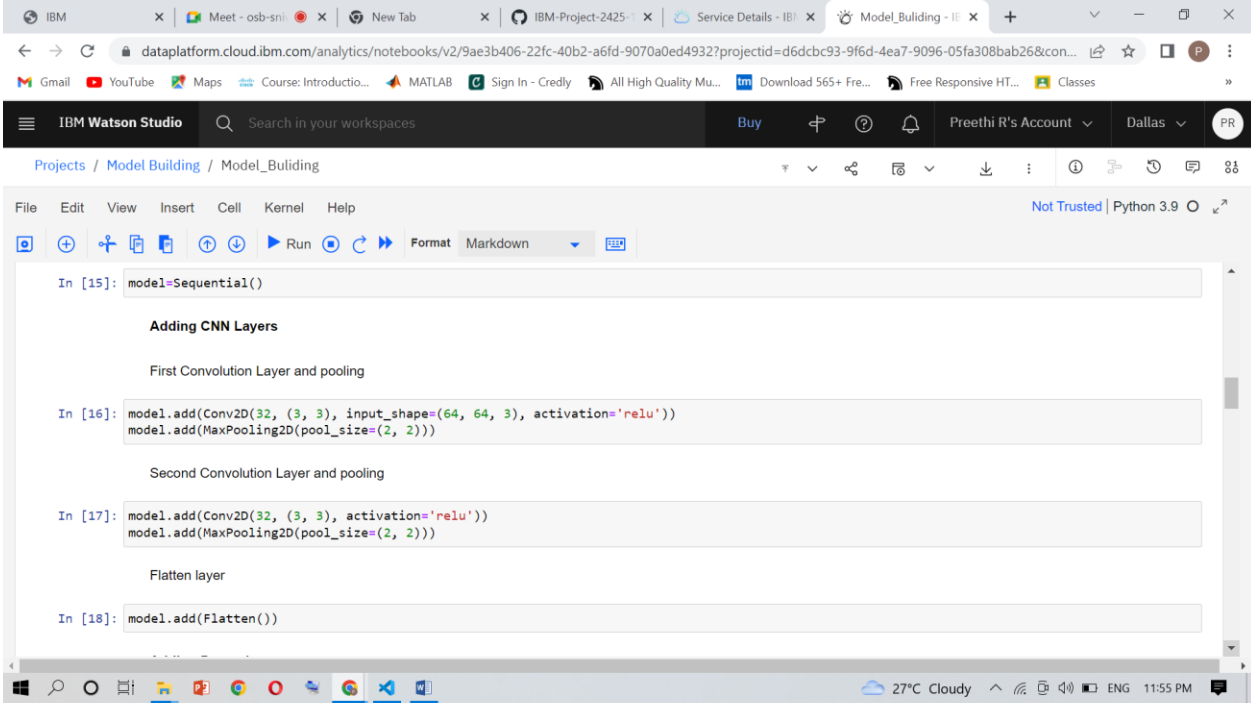
```
In [9]: x_train = train_datagen.flow_from_directory(Data_trainpath,target_size=(64,64),batch_size=5,color_mode='rgb',class_mode='sparse')

        Found 3492 images belonging to 5 classes.
```



```
In [6]: test_datagen=ImageDataGenerator(rescale=1./255)
```

Apply Image DataGenerator Functionality To Trainset And Testset

```
In [9]: x_train = train_datagen.flow_from_directory(Data_trainpath,target_size=(64,64),batch_size=5,color_mode='rgb',class_mode='sparse')

        Found 3492 images belonging to 5 classes.

In [10]: x_test = train_datagen.flow_from_directory(Data_testpath,target_size=(64,64),batch_size=5,color_mode='rgb',class_mode='sparse')

        Found 976 images belonging to 5 classes.

In [11]: print(x_train.class_indices)

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

In [12]: print(x_test.class_indices)

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

In [13]: from collections import Counter as c
```



```
In [15]: model=Sequential()
```

Adding CNN Layers

First Convolution Layer and pooling

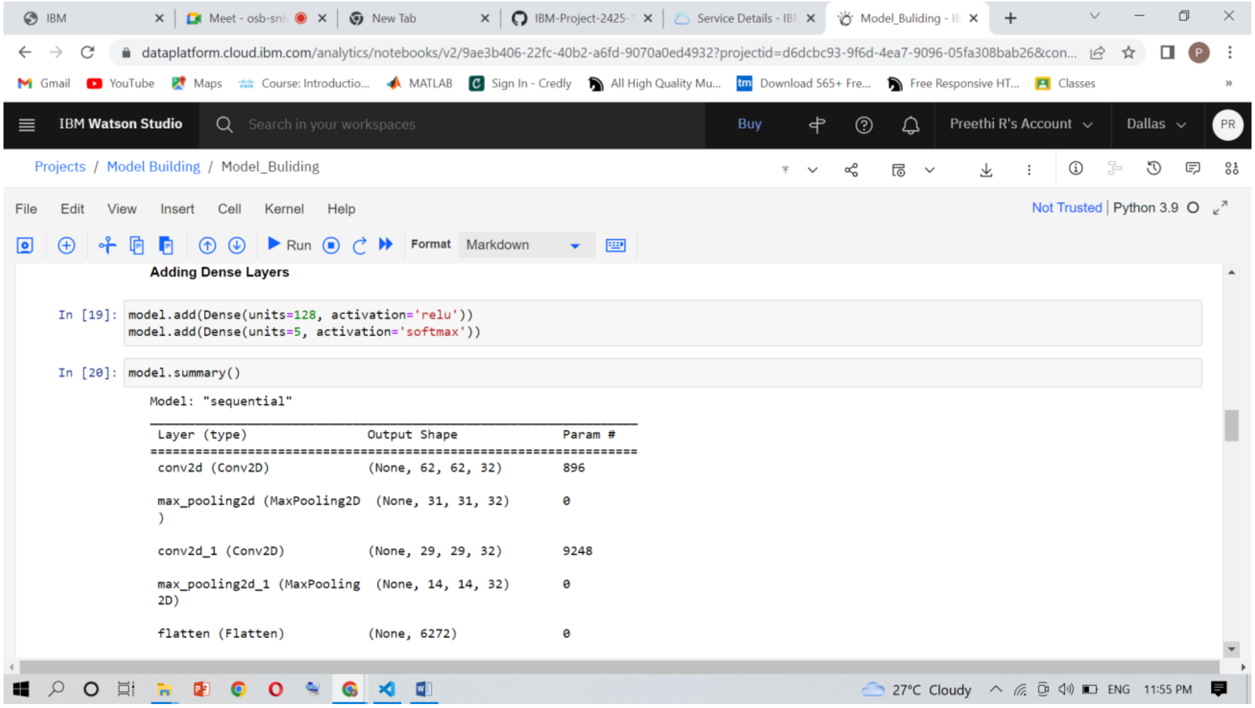
```
In [16]: model.add(Conv2D(32, (3, 3), input_shape=(64, 64, 3), activation='relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
```

Second Convolution Layer and pooling

```
In [17]: model.add(Conv2D(32, (3, 3), activation='relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
```

Flatten layer

```
In [18]: model.add(Flatten())
```

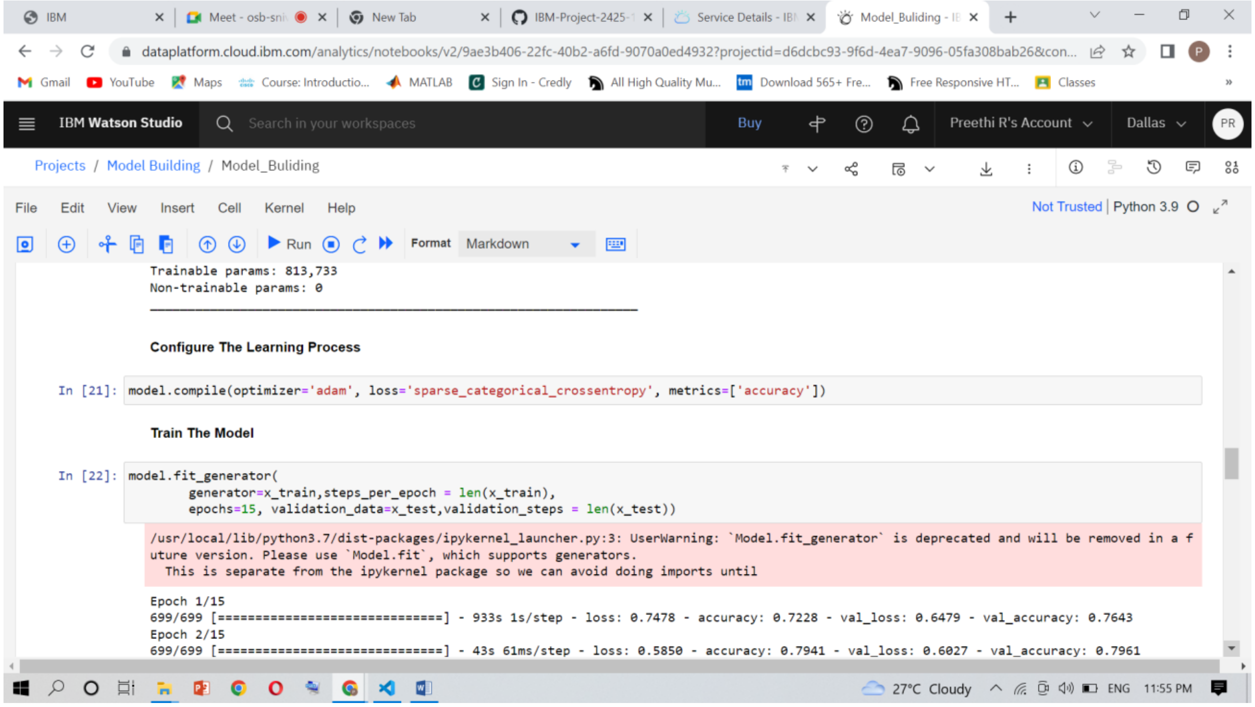


Adding Dense Layers

```
In [19]: model.add(Dense(units=128, activation='relu'))
         model.add(Dense(units=5, activation='softmax'))
```

```
In [20]: 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
```



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

Configure The Learning Process

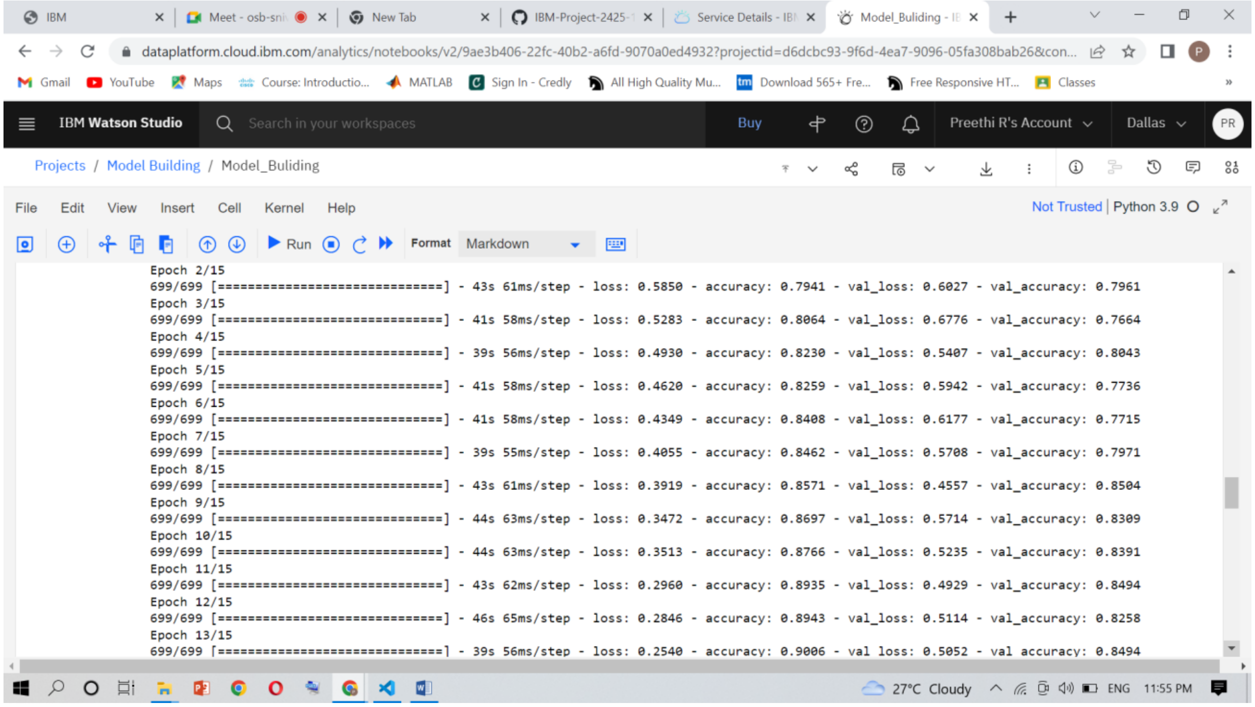
```
In [21]: model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Train The Model

```
In [22]: model.fit_generator(
                generator=x_train,steps_per_epoch = len(x_train),
                epochs=15, validation_data=x_test,validation_steps = len(x_test))
```

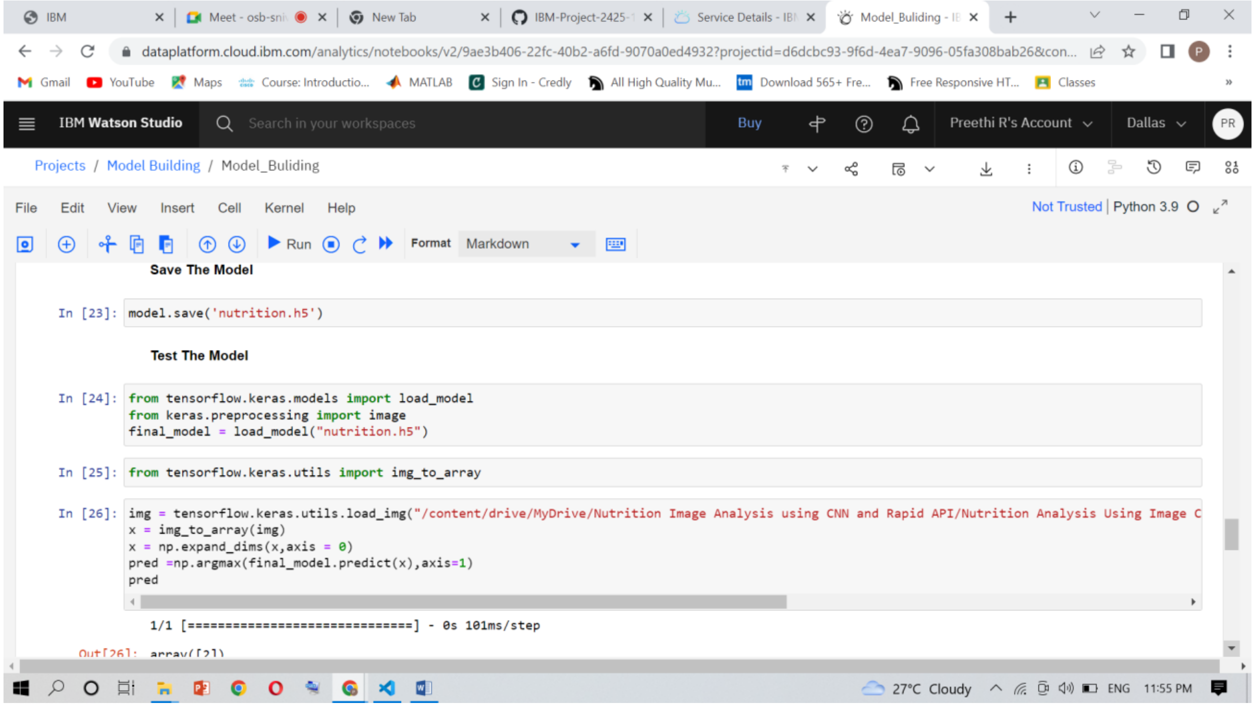
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: UserWarning: `Model.fit_generator` is deprecated and will be removed in a f
uture version. Please use `Model.fit`, which supports generators.
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
Epoch 1/15
699/699 [==============================] - 933s 1s/step - loss: 0.7478 - accuracy: 0.7228 - val_loss: 0.6479 - val_accuracy: 0.7643
Epoch 2/15
699/699 [==============================] - 43s 61ms/step - loss: 0.5850 - accuracy: 0.7941 - val_loss: 0.6027 - val_accuracy: 0.7961
```



IBM Watson Studio
Projects / Model Building / Model_Buliding
File Edit View Insert Cell Kernel Help
Not Trusted | Python 3.9
Run
Format Markdown

```
Epoch 2/15
699/699 [==============================] - 43s 61ms/step - loss: 0.5850 - accuracy: 0.7941 - val_loss: 0.6027 - val_accuracy: 0.7961
Epoch 3/15
699/699 [==============================] - 41s 58ms/step - loss: 0.5283 - accuracy: 0.8064 - val_loss: 0.6776 - val_accuracy: 0.7664
Epoch 4/15
699/699 [==============================] - 39s 56ms/step - loss: 0.4930 - accuracy: 0.8230 - val_loss: 0.5407 - val_accuracy: 0.8043
Epoch 5/15
699/699 [==============================] - 41s 58ms/step - loss: 0.4620 - accuracy: 0.8259 - val_loss: 0.5942 - val_accuracy: 0.7736
Epoch 6/15
699/699 [==============================] - 41s 58ms/step - loss: 0.4349 - accuracy: 0.8408 - val_loss: 0.6177 - val_accuracy: 0.7715
Epoch 7/15
699/699 [==============================] - 39s 55ms/step - loss: 0.4055 - accuracy: 0.8462 - val_loss: 0.5708 - val_accuracy: 0.7971
Epoch 8/15
699/699 [==============================] - 43s 61ms/step - loss: 0.3919 - accuracy: 0.8571 - val_loss: 0.4557 - val_accuracy: 0.8504
Epoch 9/15
699/699 [==============================] - 44s 63ms/step - loss: 0.3472 - accuracy: 0.8697 - val_loss: 0.5714 - val_accuracy: 0.8309
Epoch 10/15
699/699 [==============================] - 44s 63ms/step - loss: 0.3513 - accuracy: 0.8766 - val_loss: 0.5235 - val_accuracy: 0.8391
Epoch 11/15
699/699 [==============================] - 43s 62ms/step - loss: 0.2960 - accuracy: 0.8935 - val_loss: 0.4929 - val_accuracy: 0.8494
Epoch 12/15
699/699 [==============================] - 46s 65ms/step - loss: 0.2846 - accuracy: 0.8943 - val_loss: 0.5114 - val_accuracy: 0.8258
Epoch 13/15
699/699 [==============================] - 39s 56ms/step - loss: 0.2540 - accuracy: 0.9006 - val_loss: 0.5052 - val_accuracy: 0.8494
```



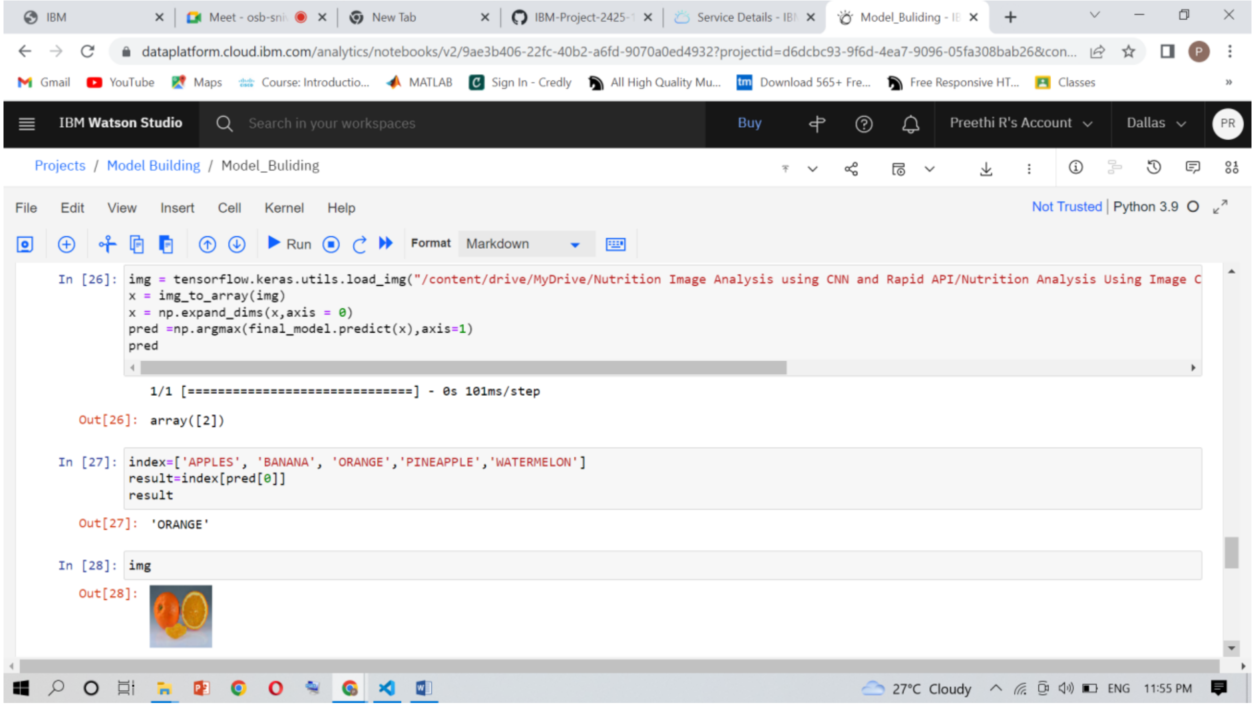
IBM Watson Studio
Projects / Model Building / Model_Buliding
File Edit View Insert Cell Kernel Help
Not Trusted | Python 3.9
Run
Format Markdown

Save The Model

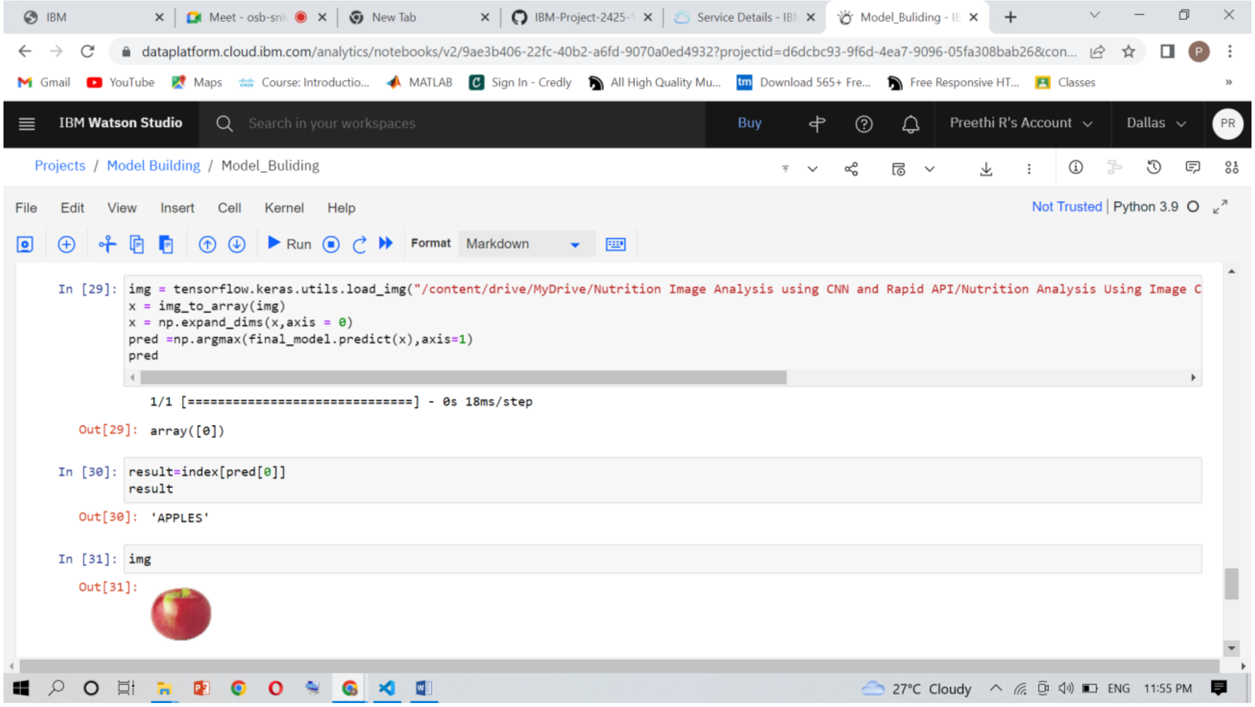
```
In [23]: model.save('nutrition.h5')
```

Test The Model

```
In [24]: from tensorflow.keras.models import load_model
         from keras.preprocessing import image
         final_model = load_model("nutrition.h5")

In [25]: from tensorflow.keras.utils import img_to_array

In [26]: img = tensorflow.keras.utils.load_img("/content/drive/MyDrive/Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image C
         x = img_to_array(img)
         x = np.expand_dims(x,axis = 0)
         pred =np.argmax(final_model.predict(x),axis=1)
         pred

         1/1 [==============================] - 0s 101ms/step

Out[26]: array([2])
```



```
In [26]: img = tensorflow.keras.utils.load_img("/content/drive/MyDrive/Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image C
         x = img_to_array(img)
         x = np.expand_dims(x,axis = 0)
         pred =np.argmax(final_model.predict(x),axis=1)
         pred

         1/1 [==============================] - 0s 101ms/step

Out[26]: array([2])

In [27]: index=['APPLES', 'BANANA', 'ORANGE','PINEAPPLE','WATERMELON']
         result=index[pred[0]]
         result

Out[27]: 'ORANGE'

In [28]: img

Out[28]:
```



```
In [29]: img = tensorflow.keras.utils.load_img("/content/drive/MyDrive/Nutrition Image Analysis using CNN and Rapid API/Nutrition Analysis Using Image C
         x = img_to_array(img)
         x = np.expand_dims(x,axis = 0)
         pred =np.argmax(final_model.predict(x),axis=1)
         pred

         1/1 [==============================] - 0s 18ms/step

Out[29]: array([0])

In [30]: result=index[pred[0]]
         result

Out[30]: 'APPLES'

In [31]: img

Out[31]:
```