FLIP ROBO

IMAGE SCRAPPING AND CLASSIFICATION PROJECT

Submitted by:
NIPAM GOGOI

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies for providing me with the opportunity to work on this project from which I have learned a lot. I am also grateful to Miss Khushboo Garg for her constant guidance and support.

INTRODUCTION

Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details

The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portal. This is done to make the model more and more robust.

BUSINESS PROBLEM FRAMING

This task is divided into two phases: Data Collection and Mode Building.

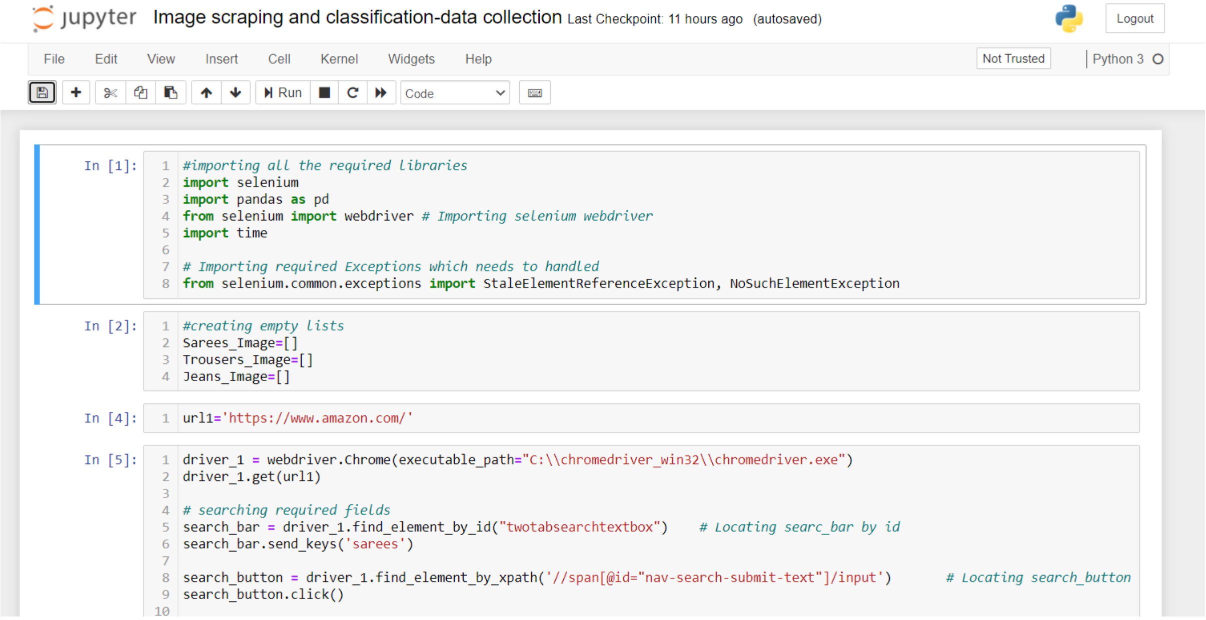
We need to scrape images from e-commerce portal, Amazon.com. The clothing categories used for scraping will be:

- Sarees (women)
- Trousers (men)
- Jeans (men)

You need to scrape images of these 3 categories and build your data from it. That data will be provided as an input to your deep learning problem. You need to scrape minimum 200 images of each categories. There is no maximum limit to the data collection. You are free to apply image augmentation techniques to increase the size of your data but make sure the quality of data is not compromised.

ANALYTICAL PROBLEM FRAMING

DATA COLLECTION - WEBSCRAPPING

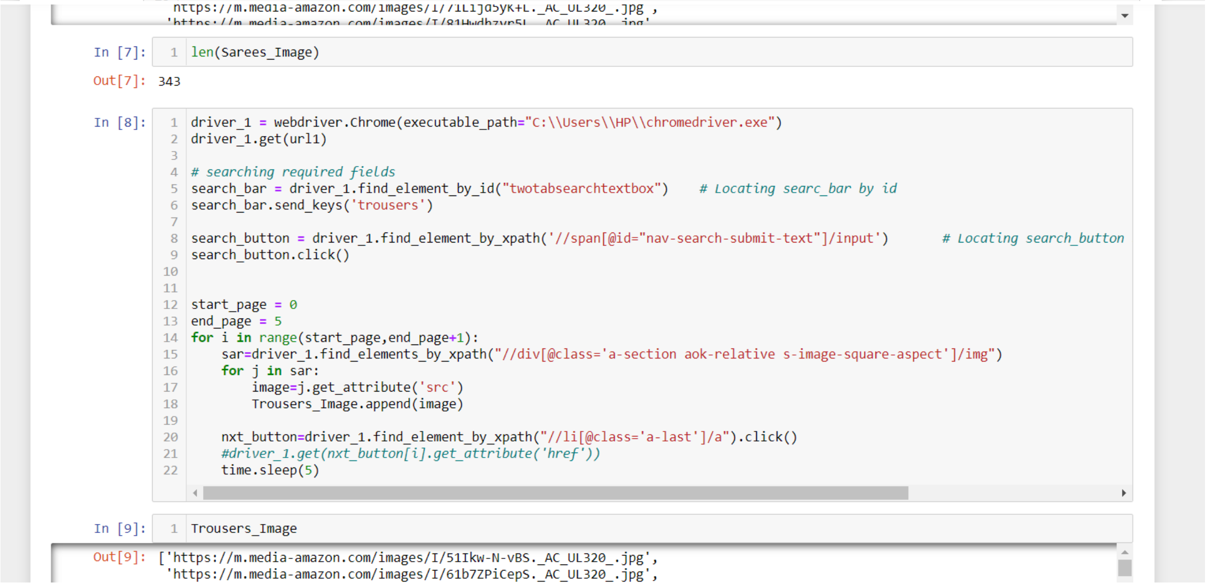

jupyter
Image scraping and classification-data collection Last Checkpoint: 11 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Not Trusted
Python 3
Run
Code

In [1]:
```
#importing all the required libraries
import selenium
import pandas as pd
from selenium import webdriver # Importing selenium webdriver
import time

# Importing required Exceptions which needs to handled
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
```

In [2]:
```
#creating empty lists
Sarees_Image=[]
Trousers_Image=[]
Jeans_Image=[]
```

In [4]:
```
url1='https://www.amazon.com/'
```

In [5]:
```
driver_1 = webdriver.Chrome(executable_path="C:\\chromedriver_win32\\chromedriver.exe")
driver_1.get(url1)

# searching required fields
search_bar = driver_1.find_element_by_id("twotabsearchtextbox")    # Locating searc_bar by id
search_bar.send_keys('sarees')

search_button = driver_1.find_element_by_xpath('//span[@id="nav-search-submit-text"]/input')       # Locating search_button
search_button.click()

```



```
 https://m.media-amazon.com/images/I/71L1jd5yK+L._AC_UL320_.jpg',
 'https://m.media-amazon.com/images/I/81Hwdbzvr5L._AC_UL320_.jpg',
```

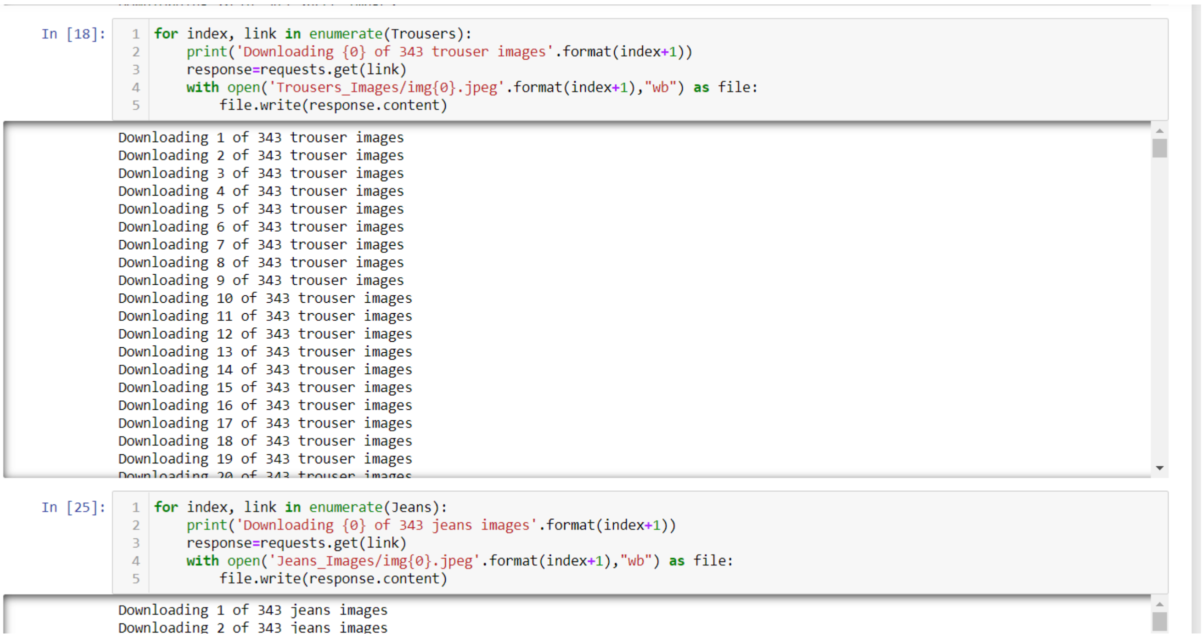
In [7]:
```
len(Sarees_Image)
```

Out[7]:
```
343
```

In [8]:
```
driver_1 = webdriver.Chrome(executable_path="C:\\Users\\HP\\chromedriver.exe")
driver_1.get(url1)

# searching required fields
search_bar = driver_1.find_element_by_id("twotabsearchtextbox")    # Locating searc_bar by id
search_bar.send_keys('trousers')

search_button = driver_1.find_element_by_xpath('//span[@id="nav-search-submit-text"]/input')       # Locating search_button
search_button.click()


start_page = 0
end_page = 5
for i in range(start_page,end_page+1):
    sar=driver_1.find_elements_by_xpath("//div[@class='a-section aok-relative s-image-square-aspect']/img")
    for j in sar:
        image=j.get_attribute('src')
        Trousers_Image.append(image)

    nxt_button=driver_1.find_element_by_xpath("//li[@class='a-last']/a").click()
    #driver_1.get(nxt_button[i].get_attribute('href'))
    time.sleep(5)
```

In [9]:
```
Trousers_Image
```

Out[9]:
```
['https://m.media-amazon.com/images/I/51Ikw-N-vBS._AC_UL320_.jpg',
 'https://m.media-amazon.com/images/I/61b7ZPiCepS._AC_UL320_.jpg',
```



In [18]:

```
for index, link in enumerate(Trousers):
    print('Downloading {0} of 343 trouser images'.format(index+1))
    response=requests.get(link)
    with open('Trousers_Images/img{0}.jpeg'.format(index+1),"wb") as file:
        file.write(response.content)
```

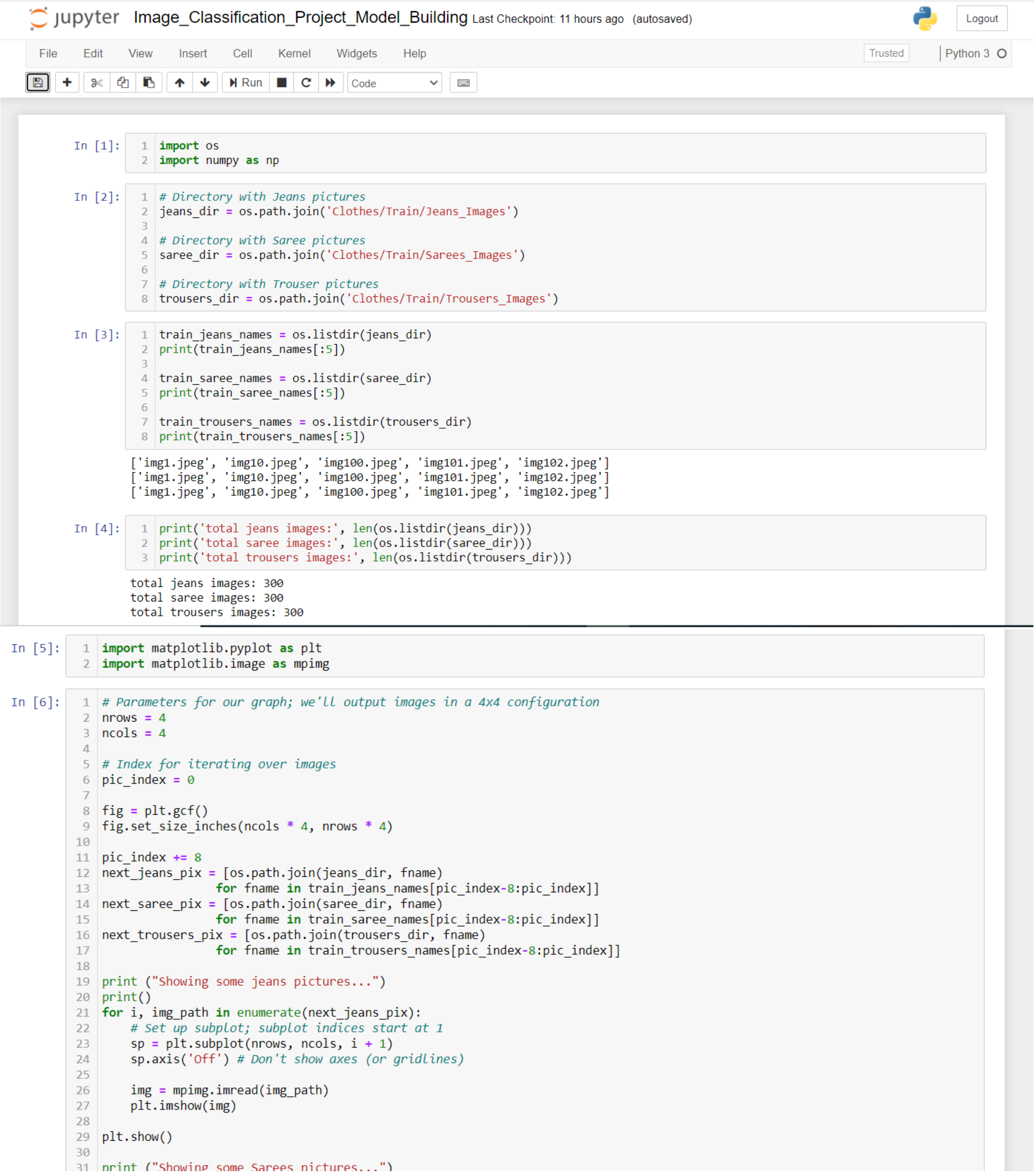
```
Downloading 1 of 343 trouser images
Downloading 2 of 343 trouser images
Downloading 3 of 343 trouser images
Downloading 4 of 343 trouser images
Downloading 5 of 343 trouser images
Downloading 6 of 343 trouser images
Downloading 7 of 343 trouser images
Downloading 8 of 343 trouser images
Downloading 9 of 343 trouser images
Downloading 10 of 343 trouser images
Downloading 11 of 343 trouser images
Downloading 12 of 343 trouser images
Downloading 13 of 343 trouser images
Downloading 14 of 343 trouser images
Downloading 15 of 343 trouser images
Downloading 16 of 343 trouser images
Downloading 17 of 343 trouser images
Downloading 18 of 343 trouser images
Downloading 19 of 343 trouser images
Downloading 20 of 343 trouser images
```

In [25]:

```
for index, link in enumerate(Jeans):
    print('Downloading {0} of 343 jeans images'.format(index+1))
    response=requests.get(link)
    with open('Jeans_Images/img{0}.jpeg'.format(index+1),"wb") as file:
        file.write(response.content)
```

```
Downloading 1 of 343 jeans images
Downloading 2 of 343 jeans images
```

MODEL BUILDING


jupyter
Image_Classification_Project_Model_Building Last Checkpoint: 11 hours ago (autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

In [1]:

```
import os
import numpy as np
```

In [2]:

```
# Directory with Jeans pictures
jeans_dir = os.path.join('Clothes/Train/Jeans_Images')

# Directory with Saree pictures
saree_dir = os.path.join('Clothes/Train/Sarees_Images')

# Directory with Trouser pictures
trousers_dir = os.path.join('Clothes/Train/Trousers_Images')
```

In [3]:

```
train_jeans_names = os.listdir(jeans_dir)
print(train_jeans_names[:5])

train_saree_names = os.listdir(saree_dir)
print(train_saree_names[:5])

train_trousers_names = os.listdir(trousers_dir)
print(train_trousers_names[:5])
```

```
['img1.jpeg', 'img10.jpeg', 'img100.jpeg', 'img101.jpeg', 'img102.jpeg']
['img1.jpeg', 'img10.jpeg', 'img100.jpeg', 'img101.jpeg', 'img102.jpeg']
['img1.jpeg', 'img10.jpeg', 'img100.jpeg', 'img101.jpeg', 'img102.jpeg']
```

In [4]:

```
print('total jeans images:', len(os.listdir(jeans_dir)))
print('total saree images:', len(os.listdir(saree_dir)))
print('total trousers images:', len(os.listdir(trousers_dir)))
```

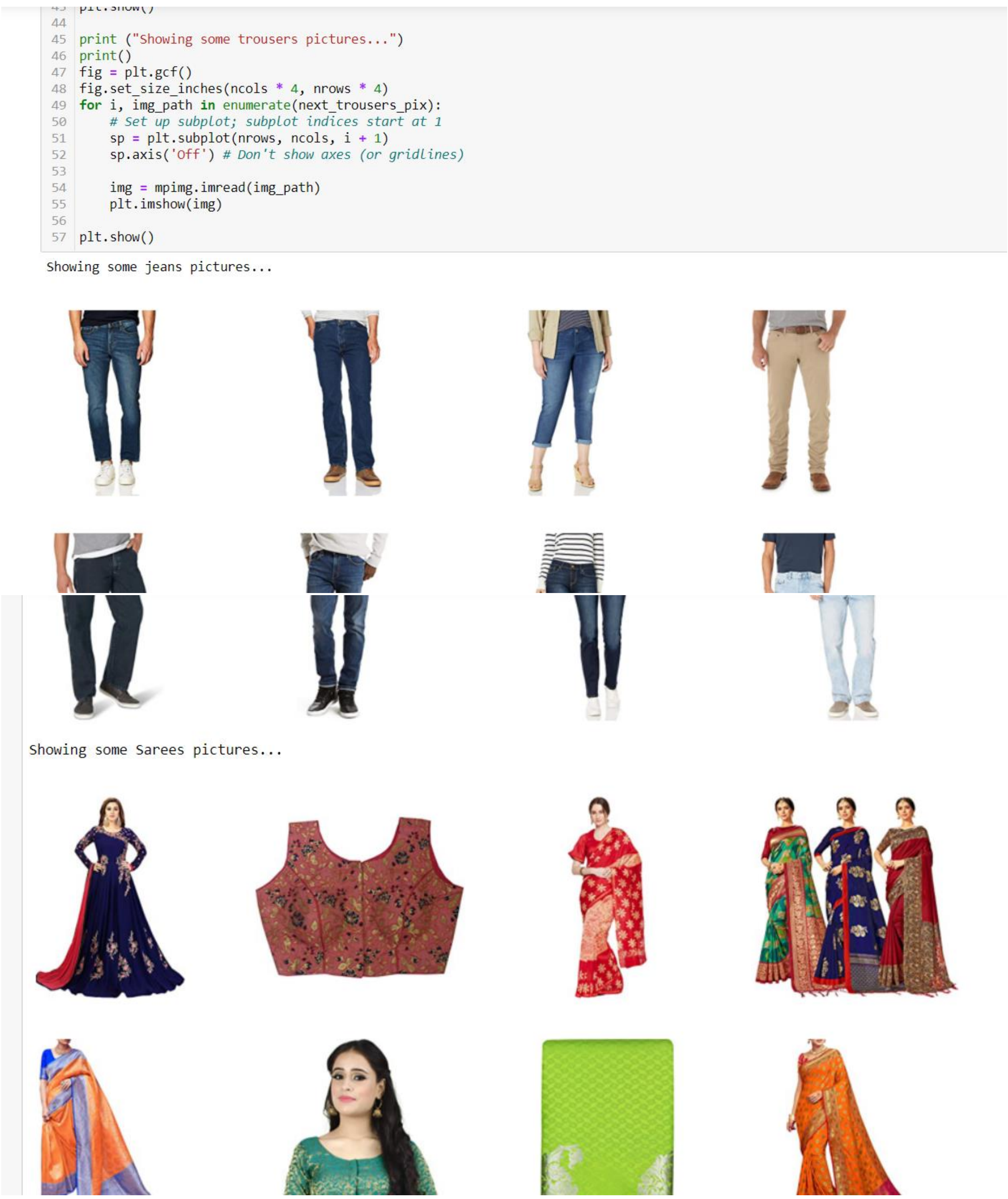
```
total jeans images: 300
total saree images: 300
total trousers images: 300
```

In [5]:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

In [6]:

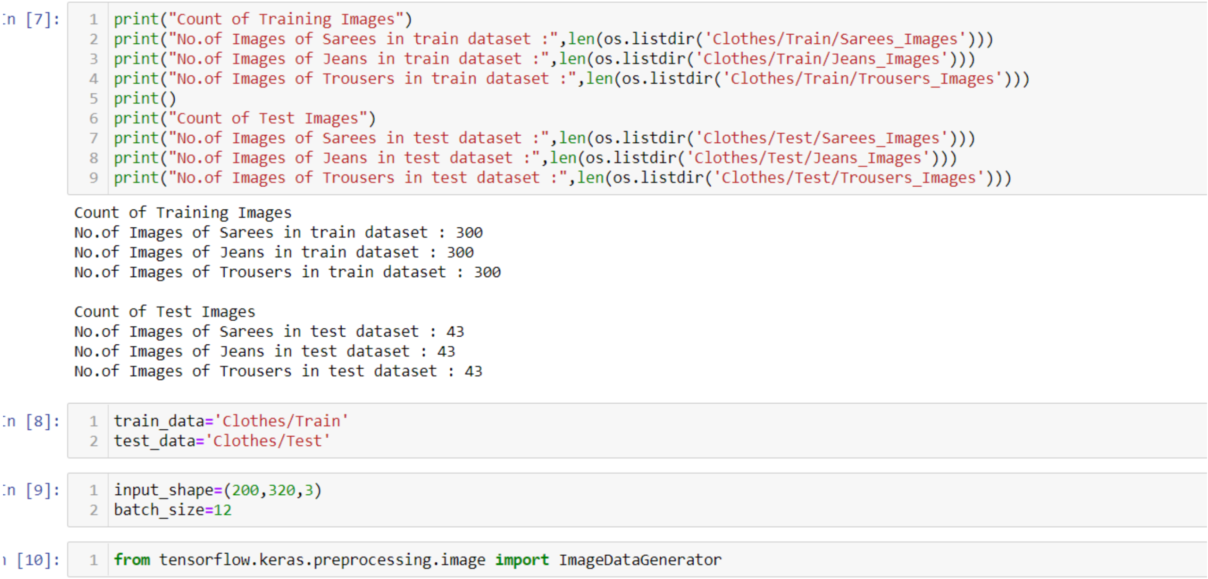
```
# Parameters for our graph; we'll output images in a 4x4 configuration
nrows = 4
ncols = 4

# Index for iterating over images
pic_index = 0

fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)

pic_index += 8
next_jeans_pix = [os.path.join(jeans_dir, fname)
                for fname in train_jeans_names[pic_index-8:pic_index]]
next_saree_pix = [os.path.join(saree_dir, fname)
                for fname in train_saree_names[pic_index-8:pic_index]]
next_trousers_pix = [os.path.join(trousers_dir, fname)
                for fname in train_trousers_names[pic_index-8:pic_index]]

print ("Showing some jeans pictures...")
print()
for i, img_path in enumerate(next_jeans_pix):
    # Set up subplot; subplot indices start at 1
    sp = plt.subplot(nrows, ncols, i + 1)
    sp.axis('Off') # Don't show axes (or gridlines)

    img = mpimg.imread(img_path)
    plt.imshow(img)

plt.show()

print ("Showing some Sarees pictures...")
```



```
plt.show()

print ("Showing some trousers pictures...")
print()
fig = plt.gcf()
fig.set_size_inches(ncols * 4, nrows * 4)
for i, img_path in enumerate(next_trousers_pix):
    # Set up subplot; subplot indices start at 1
    sp = plt.subplot(nrows, ncols, i + 1)
    sp.axis('Off') # Don't show axes (or gridlines)

    img = mpimg.imread(img_path)
    plt.imshow(img)

plt.show()
```

```
Showing some jeans pictures...
```

```
Showing some Sarees pictures...
```



:n [7]:

```
print("Count of Training Images")
print("No.of Images of Sarees in train dataset :",len(os.listdir('Clothes/Train/Sarees_Images')))
print("No.of Images of Jeans in train dataset :",len(os.listdir('Clothes/Train/Jeans_Images')))
print("No.of Images of Trousers in train dataset :",len(os.listdir('Clothes/Train/Trousers_Images')))
print()
print("Count of Test Images")
print("No.of Images of Sarees in test dataset :",len(os.listdir('Clothes/Test/Sarees_Images')))
print("No.of Images of Jeans in test dataset :",len(os.listdir('Clothes/Test/Jeans_Images')))
print("No.of Images of Trousers in test dataset :",len(os.listdir('Clothes/Test/Trousers_Images')))
```

```
Count of Training Images
No.of Images of Sarees in train dataset : 300
No.of Images of Jeans in train dataset : 300
No.of Images of Trousers in train dataset : 300

Count of Test Images
No.of Images of Sarees in test dataset : 43
No.of Images of Jeans in test dataset : 43
No.of Images of Trousers in test dataset : 43
```

:n [8]:

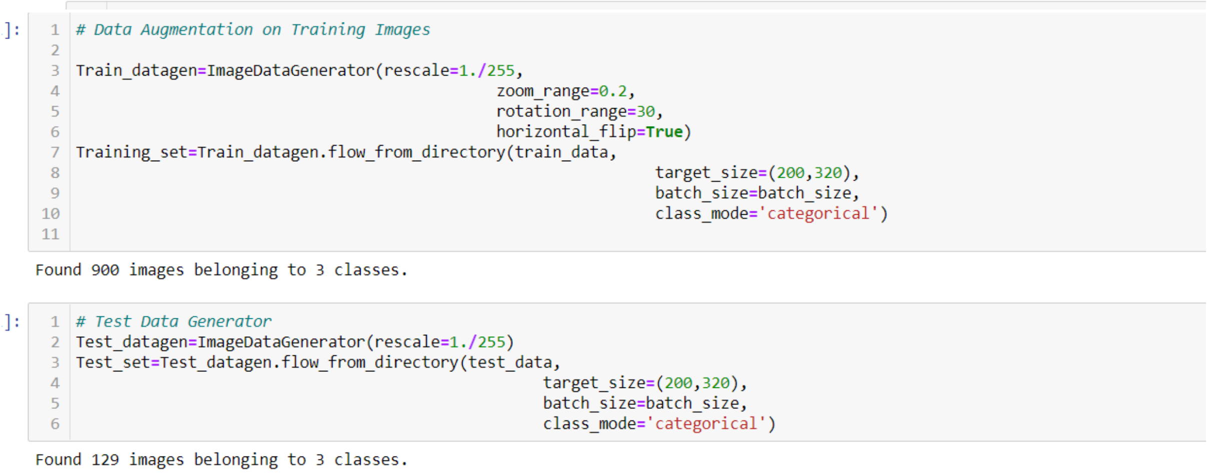
```
train_data='Clothes/Train'
test_data='Clothes/Test'
```

:n [9]:

```
input_shape=(200,320,3)
batch_size=12
```

ı [10]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```



]:

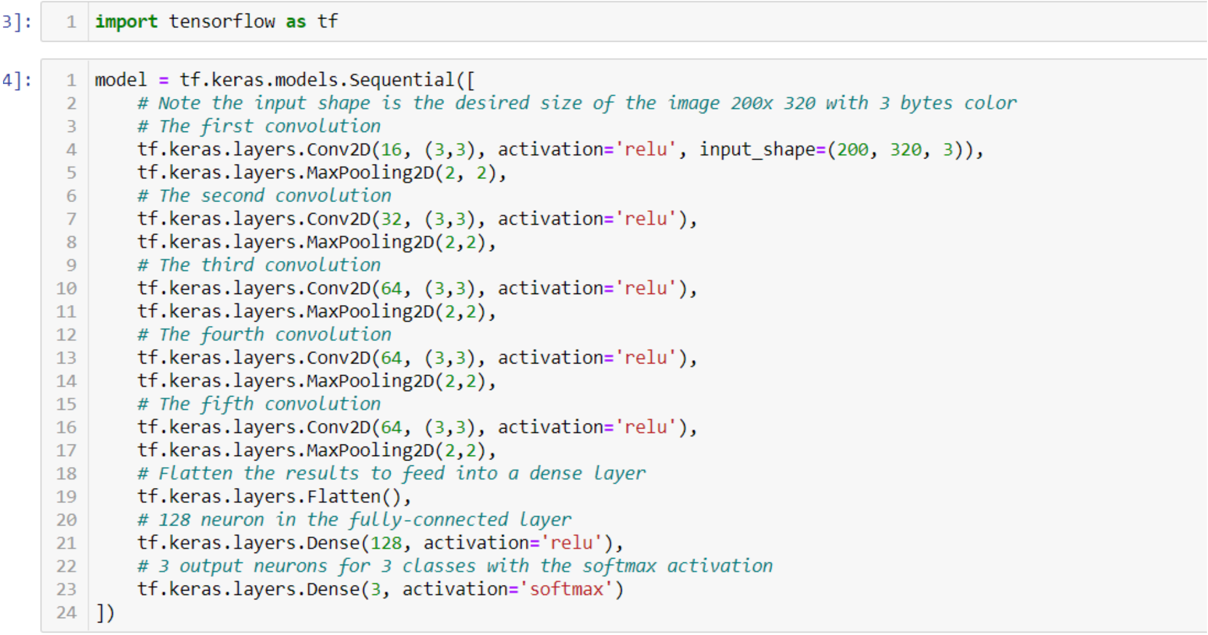
```
# Data Augmentation on Training Images

Train_datagen=ImageDataGenerator(rescale=1./255,
                                              zoom_range=0.2,
                                              rotation_range=30,
                                              horizontal_flip=True)
Training_set=Train_datagen.flow_from_directory(train_data,
                                                               target_size=(200,320),
                                                               batch_size=batch_size,
                                                               class_mode='categorical')

```

```
Found 900 images belonging to 3 classes.
```

]:

```
# Test Data Generator
Test_datagen=ImageDataGenerator(rescale=1./255)
Test_set=Test_datagen.flow_from_directory(test_data,
                                                   target_size=(200,320),
                                                   batch_size=batch_size,
                                                   class_mode='categorical')
```

```
Found 129 images belonging to 3 classes.
```

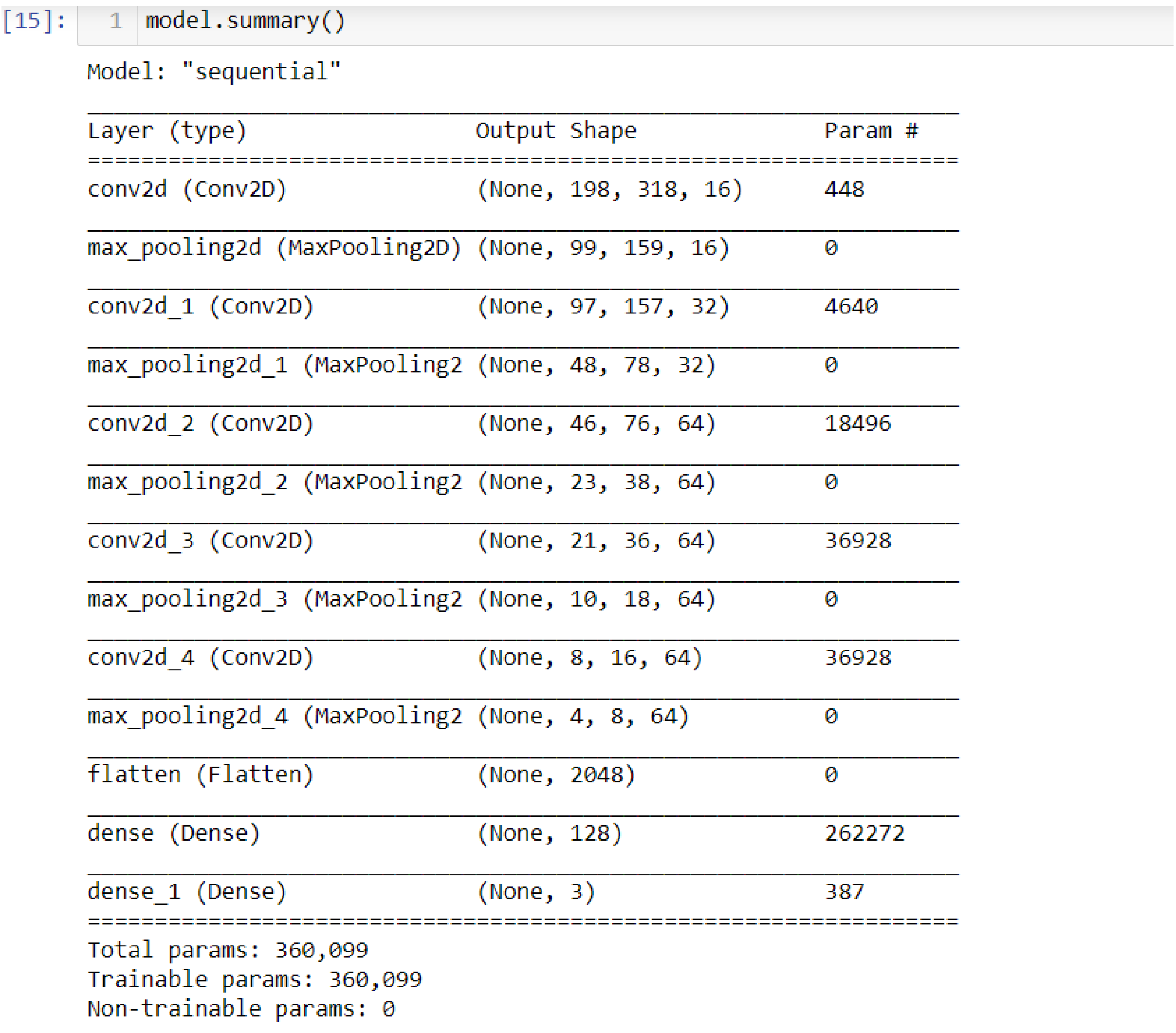


3]:

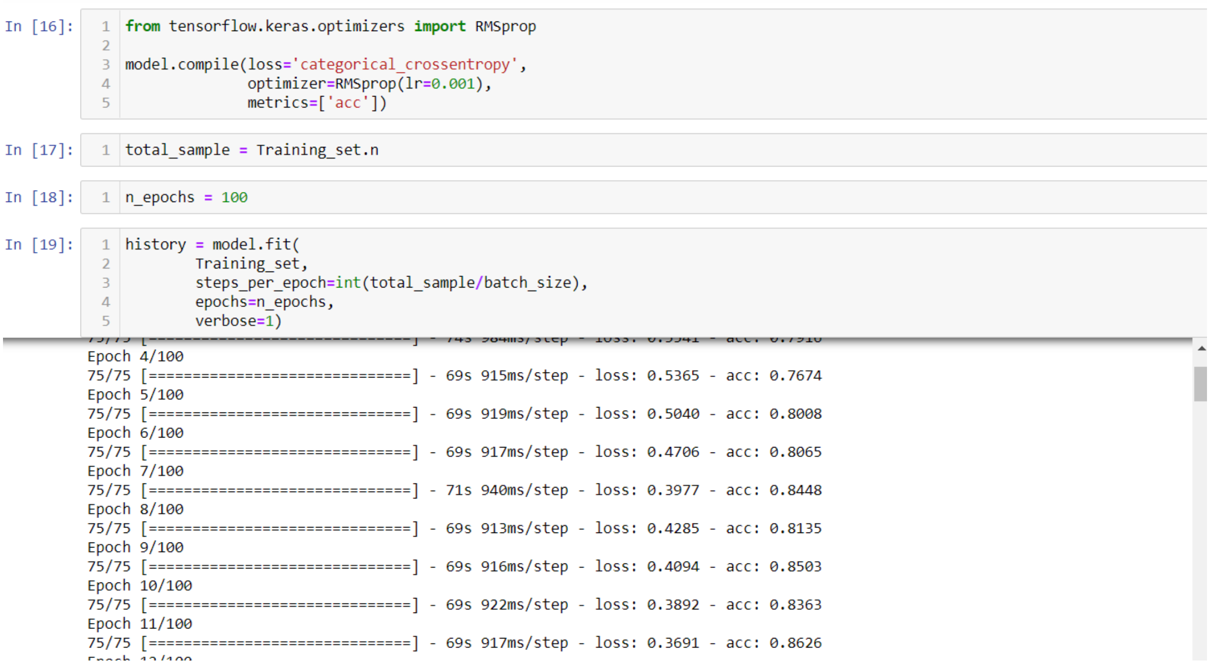
```
import tensorflow as tf
```

4]:

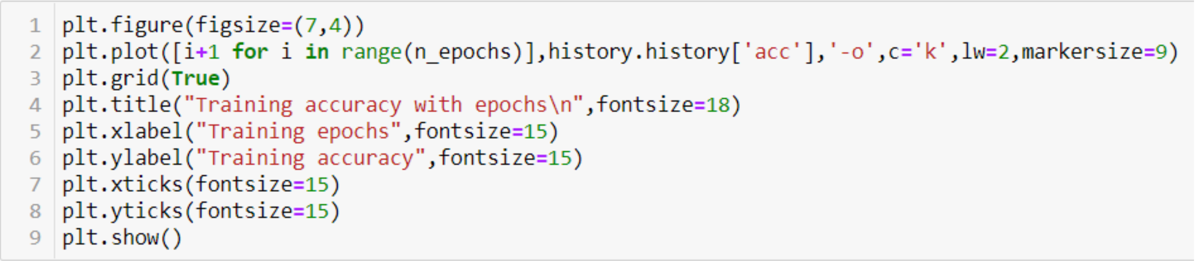
```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 320 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 320, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fifth convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a dense layer
    tf.keras.layers.Flatten(),
    # 128 neuron in the fully-connected layer
    tf.keras.layers.Dense(128, activation='relu'),
    # 3 output neurons for 3 classes with the softmax activation
    tf.keras.layers.Dense(3, activation='softmax')
])
```



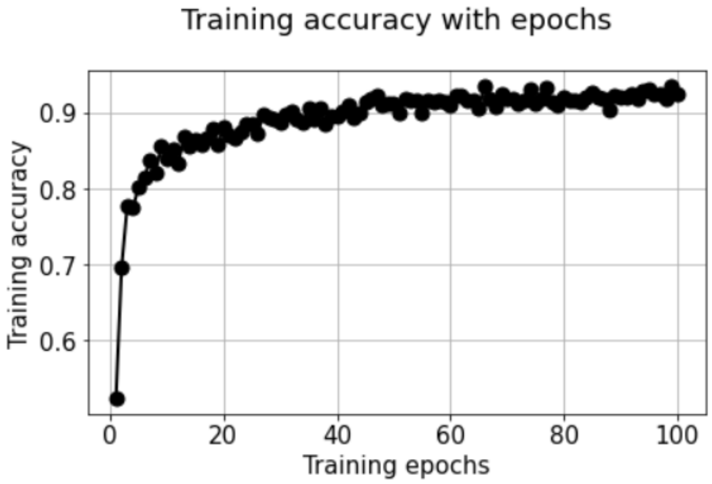
```
[15]:  model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 318, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 99, 159, 16)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 97, 157, 32)       4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 78, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 76, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 38, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 36, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 18, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 16, 64)         36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 8, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 2048)              0
_________________________________________________________________
dense (Dense)                (None, 128)               262272
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 360,099
Trainable params: 360,099
Non-trainable params: 0
```

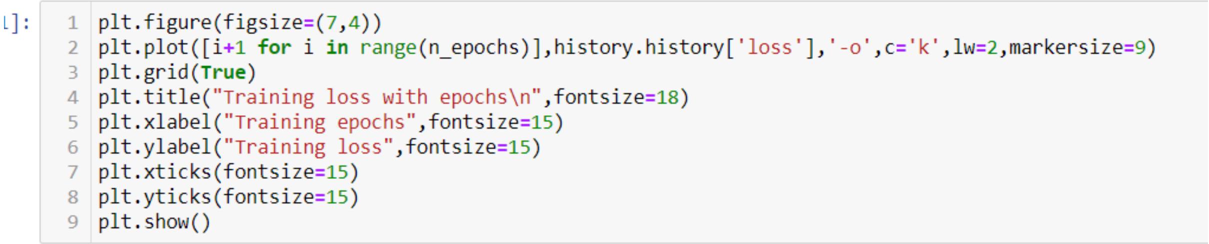


```
In [16]:  from tensorflow.keras.optimizers import RMSprop

          model.compile(loss='categorical_crossentropy',
                        optimizer=RMSprop(lr=0.001),
                        metrics=['acc'])

In [17]:  total_sample = Training_set.n

In [18]:  n_epochs = 100

In [19]:  history = model.fit(
                  Training_set,
                  steps_per_epoch=int(total_sample/batch_size),
                  epochs=n_epochs,
                  verbose=1)
75/75 [==============================] - 74s 984ms/step - loss: 0.5541 - acc: 0.7910
Epoch 4/100
75/75 [==============================] - 69s 915ms/step - loss: 0.5365 - acc: 0.7674
Epoch 5/100
75/75 [==============================] - 69s 919ms/step - loss: 0.5040 - acc: 0.8008
Epoch 6/100
75/75 [==============================] - 69s 917ms/step - loss: 0.4706 - acc: 0.8065
Epoch 7/100
75/75 [==============================] - 71s 940ms/step - loss: 0.3977 - acc: 0.8448
Epoch 8/100
75/75 [==============================] - 69s 913ms/step - loss: 0.4285 - acc: 0.8135
Epoch 9/100
75/75 [==============================] - 69s 916ms/step - loss: 0.4094 - acc: 0.8503
Epoch 10/100
75/75 [==============================] - 69s 922ms/step - loss: 0.3892 - acc: 0.8363
Epoch 11/100
75/75 [==============================] - 69s 917ms/step - loss: 0.3691 - acc: 0.8626
Epoch 12/100
```

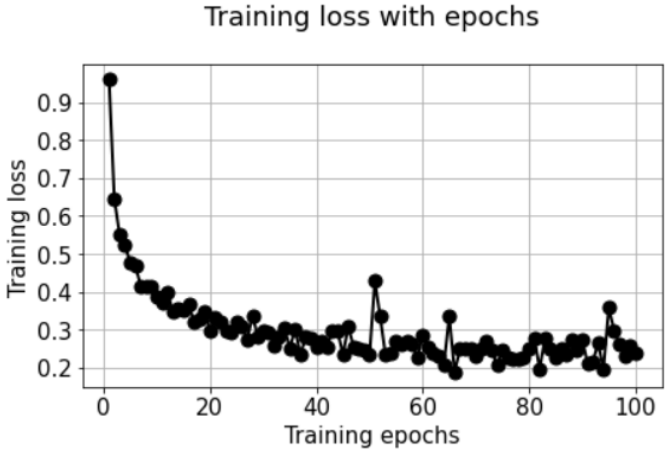


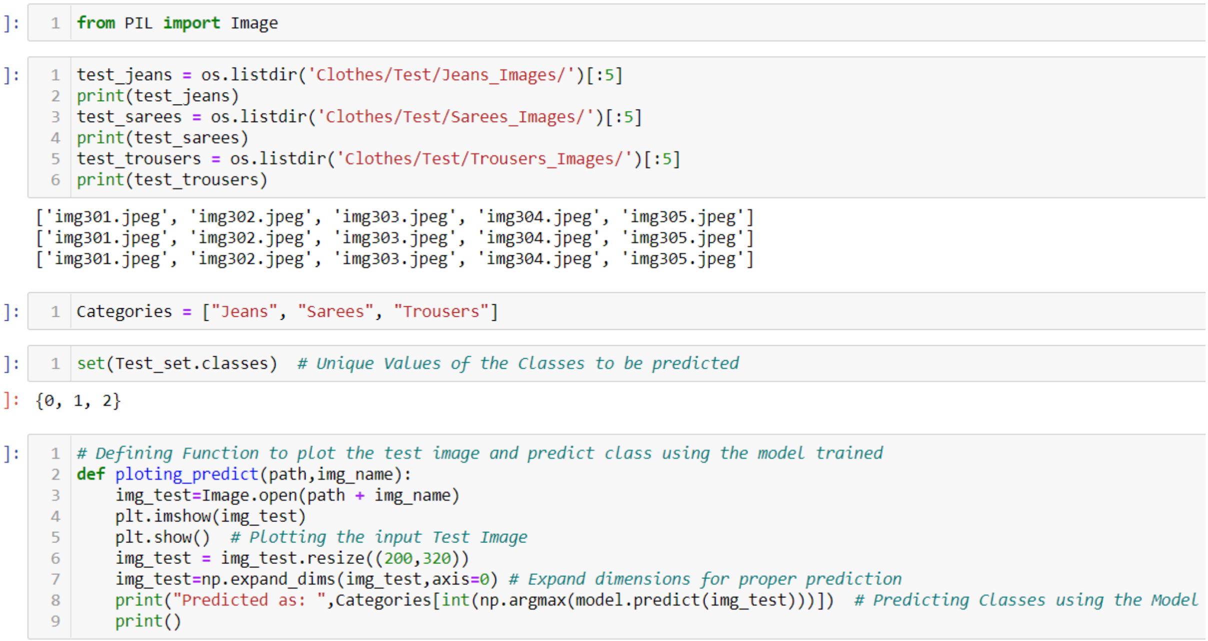
```python
plt.figure(figsize=(7,4))
plt.plot([i+1 for i in range(n_epochs)],history.history['acc'],'-o',c='k',lw=2,markersize=9)
plt.grid(True)
plt.title("Training accuracy with epochs\n",fontsize=18)
plt.xlabel("Training epochs",fontsize=15)
plt.ylabel("Training accuracy",fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```





```python
plt.figure(figsize=(7,4))
plt.plot([i+1 for i in range(n_epochs)],history.history['loss'],'-o',c='k',lw=2,markersize=9)
plt.grid(True)
plt.title("Training loss with epochs\n",fontsize=18)
plt.xlabel("Training epochs",fontsize=15)
plt.ylabel("Training loss",fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```





]:

```
from PIL import Image
```

]:

```
test_jeans = os.listdir('Clothes/Test/Jeans_Images/')[:5]
print(test_jeans)
test_sarees = os.listdir('Clothes/Test/Sarees_Images/')[:5]
print(test_sarees)
test_trousers = os.listdir('Clothes/Test/Trousers_Images/')[:5]
print(test_trousers)
```

```
['img301.jpeg', 'img302.jpeg', 'img303.jpeg', 'img304.jpeg', 'img305.jpeg']
['img301.jpeg', 'img302.jpeg', 'img303.jpeg', 'img304.jpeg', 'img305.jpeg']
['img301.jpeg', 'img302.jpeg', 'img303.jpeg', 'img304.jpeg', 'img305.jpeg']
```

]:

```
Categories = ["Jeans", "Sarees", "Trousers"]
```

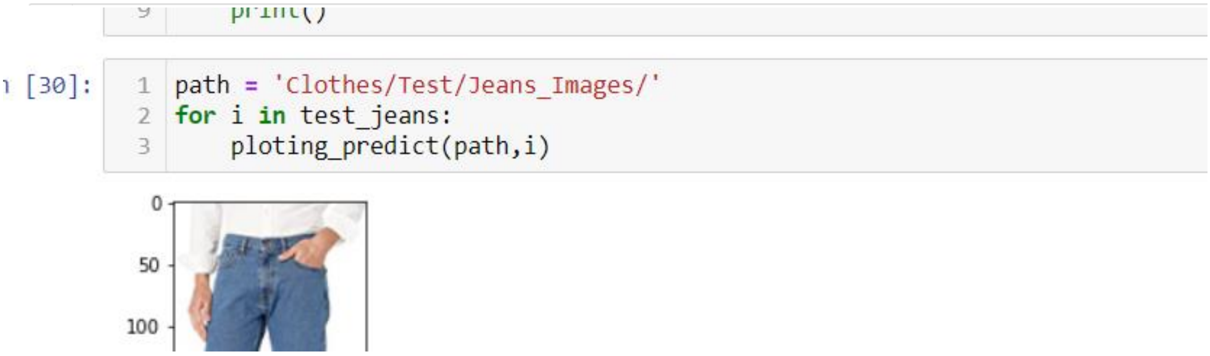
]:

```
set(Test_set.classes)  # Unique Values of the Classes to be predicted
```

]: {0, 1, 2}

]:

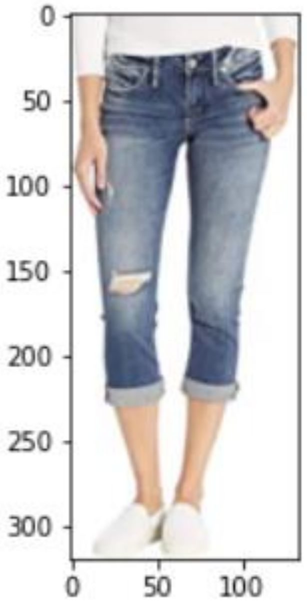
```
# Defining Function to plot the test image and predict class using the model trained
def ploting_predict(path,img_name):
    img_test=Image.open(path + img_name)
    plt.imshow(img_test)
    plt.show()  # Plotting the input Test Image
    img_test = img_test.resize((200,320))
    img_test=np.expand_dims(img_test,axis=0) # Expand dimensions for proper prediction
    print("Predicted as: ",Categories[int(np.argmax(model.predict(img_test)))])  # Predicting Classes using the Model
    print()
```



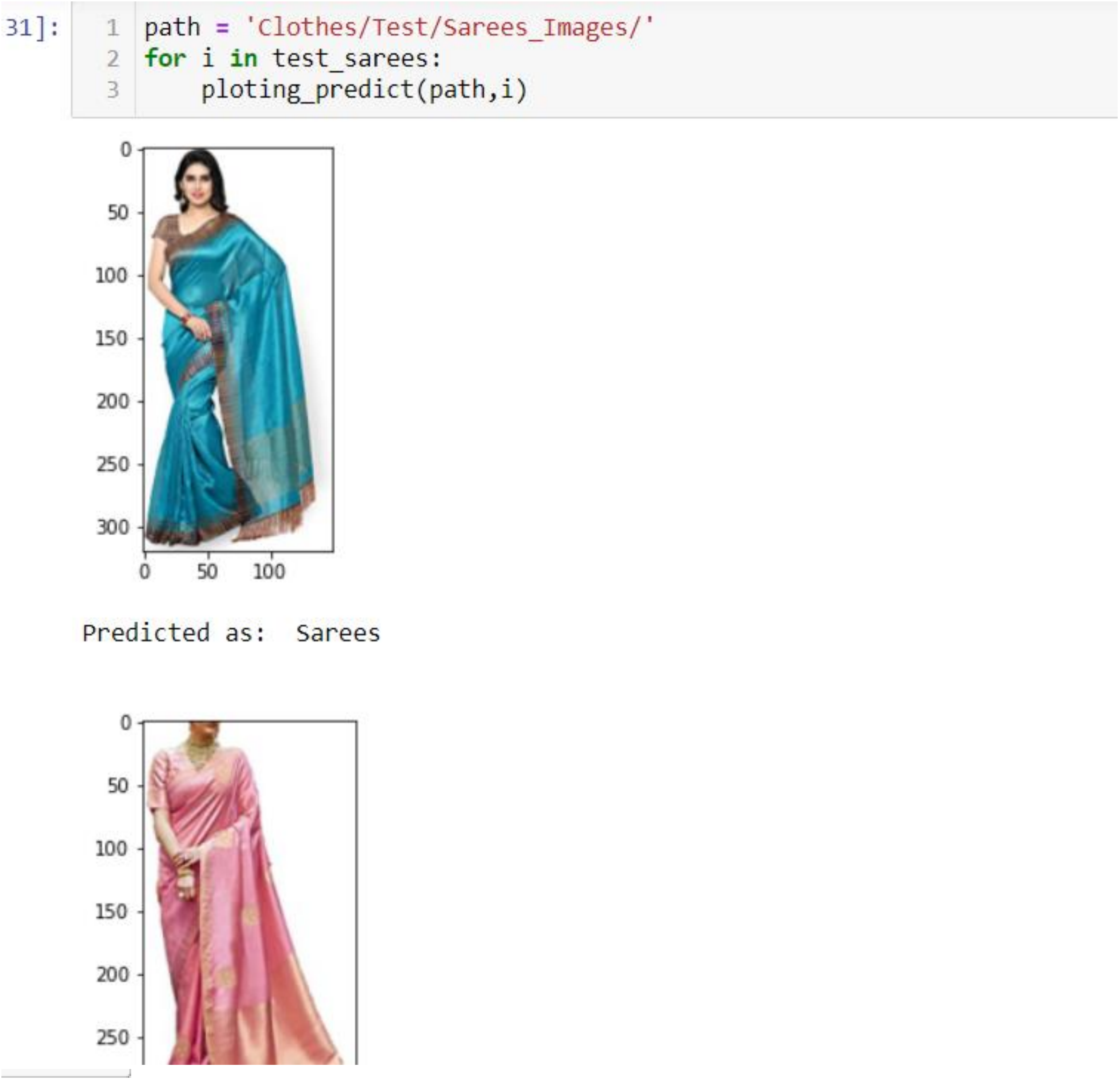
```
    print()
```

ı [30]:

```
path = 'Clothes/Test/Jeans_Images/'
for i in test_jeans:
    ploting_predict(path,i)
```

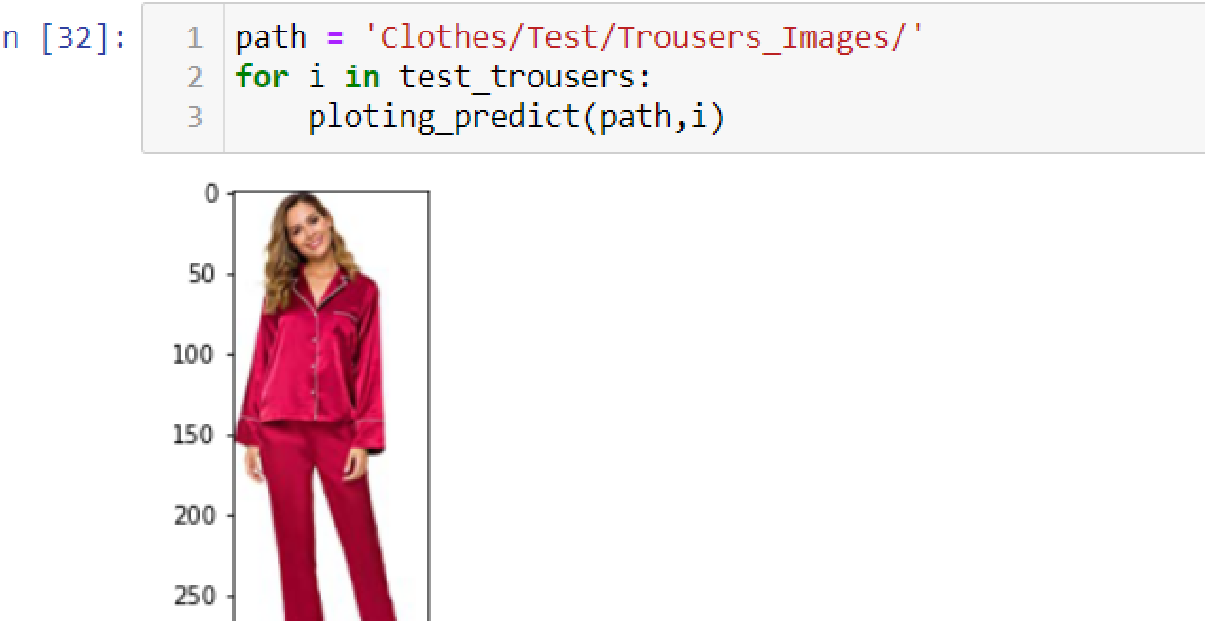


```
Predicted as:  Jeans
```



```
path = 'Clothes/Test/Sarees_Images/'
for i in test_sarees:
    ploting_predict(path,i)
```

```
Predicted as:  Sarees
```



```
path = 'Clothes/Test/Trousers_Images/'
for i in test_trousers:
    ploting_predict(path,i)
```

CONCLUSION

- The Image Data was collected using webscrapping from Amazon for Jeans, Sarees and Trousers.
- We have used Deep Learning model – Convolutional Neural Network for the project
- The model is working well and was able to classify the three clothing items properly with 92% overall accuracy.
- We can improve the classification of Jeans and Trousers by increasing the training dataset. For Sarees it was accurately predicted in test data.
- Since in all three categories there were some extra/unnecessary items other than the main items hence, it could have been removed and we could have got better result. Moreover, training data could have been increased.