FLIP ROBO

Image Scraping and Classification Project

Submitted by: Deepam Purkayastha

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 Mr. Shubham Yadav for his constant guidance and support.

Reference sources are :-

- Google
- Stackoverflow.com
- Notes and repository from DataTrained
- Krish Naik youtube videos

INTRODUCTION

Business Problem Framing

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. We are trying to give an exposure of how an end to end project is developed in this field.

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.

Motivation for the Problem Undertaken

In today world we are dealing with lots of data which is present in various formats like numbers ,categories, images etc. The data present in numbers or categorical form is easy to store and understand but image data is somehow different form these. In image data we have to use some different techniques to deal it and these new techniques ,algorithms to save and interpret and the way to deal image data motivates me to undertake this project and build a model for image classification.

Analytical Problem Framing

Modeling of the Problem

In this project we are dealing with image classification model that will classify between these 3 categories. This project is divided into 2 main phases.

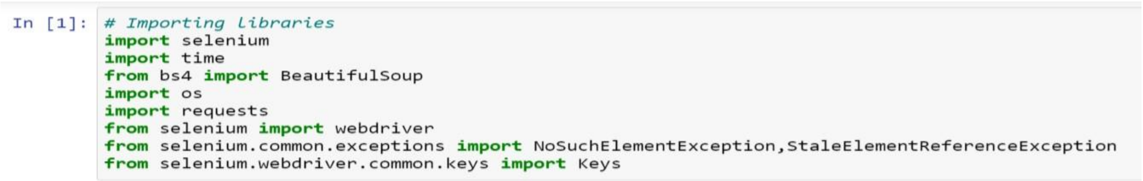
a) Data Collection Phase
b) Model Building Phase

In data collection phase we have to scrape and collect data and after collection data we have to build convolution neural network model.

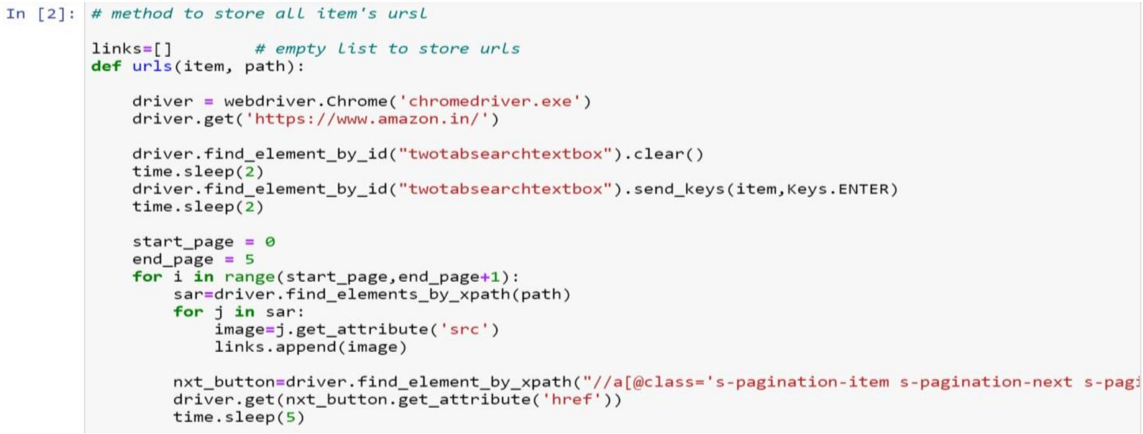
Before the model building we scraped the images of sarees, jeans ,trousers from e-commerce website Amazon.

Data Collection

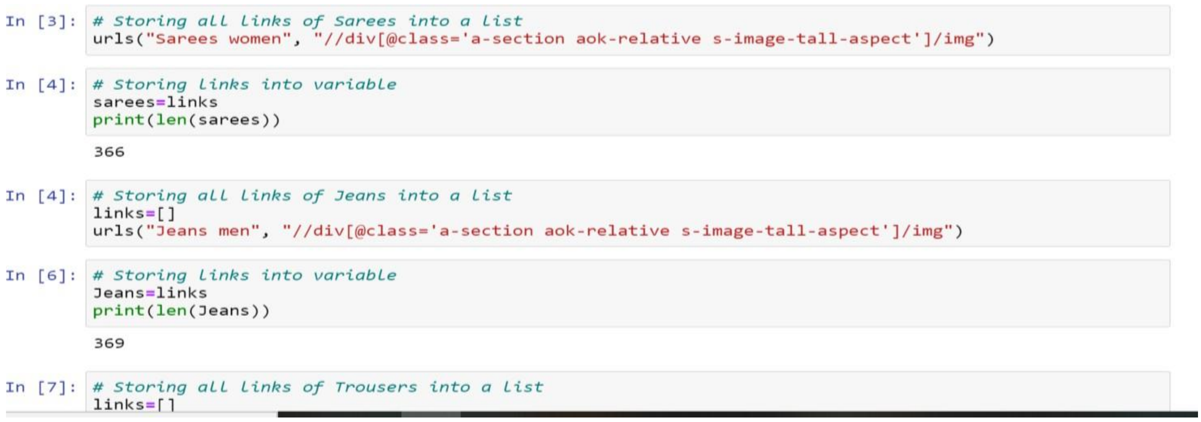
We gathered our data from online e-commerce website Amazon. As this is image classification project of 3 categories. So we collected the images of only 3 categories i.e Sarees, Jeans and Trousers. We save these images into our local system and then using Keras and Tensorflow modules we build an image classification model.



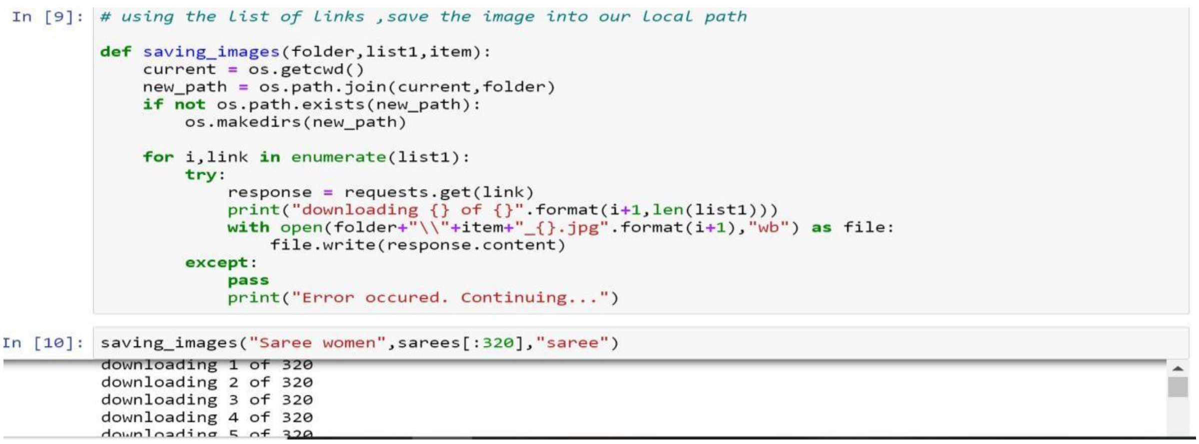
```
In [1]: # Importing Libraries
        import selenium
        import time
        from bs4 import BeautifulSoup
        import os
        import requests
        from selenium import webdriver
        from selenium.common.exceptions import NoSuchElementException,StaleElementReferenceException
        from selenium.webdriver.common.keys import Keys
```



```
In [2]: # method to store all item's ursl

        links=[]         # empty list to store urls
        def urls(item, path):

            driver = webdriver.Chrome('chromedriver.exe')
            driver.get('https://www.amazon.in/')

            driver.find_element_by_id("twotabsearchtextbox").clear()
            time.sleep(2)
            driver.find_element_by_id("twotabsearchtextbox").send_keys(item,Keys.ENTER)
            time.sleep(2)

            start_page = 0
            end_page = 5
            for i in range(start_page,end_page+1):
                sar=driver.find_elements_by_xpath(path)
                for j in sar:
                    image=j.get_attribute('src')
                    links.append(image)

                nxt_button=driver.find_element_by_xpath("//a[@class='s-pagination-item s-pagination-next s-pagi
                driver.get(nxt_button.get_attribute('href'))
                time.sleep(5)
```



```
In [3]: # Storing all links of Sarees into a list
        urls("Sarees women", "//div[@class='a-section aok-relative s-image-tall-aspect']/img")

In [4]: # Storing links into variable
        sarees=links
        print(len(sarees))

        366

In [4]: # Storing all links of Jeans into a list
        links=[]
        urls("Jeans men", "//div[@class='a-section aok-relative s-image-tall-aspect']/img")

In [6]: # Storing links into variable
        Jeans=links
        print(len(Jeans))

        369

In [7]: # Storing all links of Trousers into a list
        links=[]
```



```
In [9]: # using the list of links ,save the image into our local path

        def saving_images(folder,list1,item):
            current = os.getcwd()
            new_path = os.path.join(current,folder)
            if not os.path.exists(new_path):
                os.makedirs(new_path)

            for i,link in enumerate(list1):
                try:
                    response = requests.get(link)
                    print("downloading {} of {}".format(i+1,len(list1)))
                    with open(folder+"\\"+item+"_{}.jpg".format(i+1),"wb") as file:
                        file.write(response.content)
                except:
                    pass
                    print("Error occured. Continuing...")

In [10]: saving_images("Saree women",sarees[:320],"saree")
        downloading 1 of 320
        downloading 2 of 320
        downloading 3 of 320
        downloading 4 of 320
        downloading 5 of 320
```

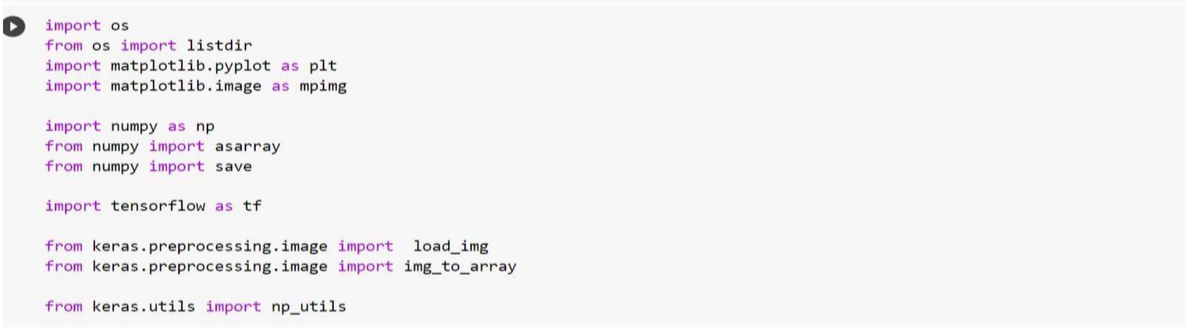
So in these we steps we collected the urls links for each image into a separate list and then using these lists we save/downloaded the image into our system.

- ## Data Preprocessing -

In data preprocessing and model building we are using google colab for our further project needs. After downloading the images we uploaded them into drive and then using the google mount option we were able to use them in our model building phase.

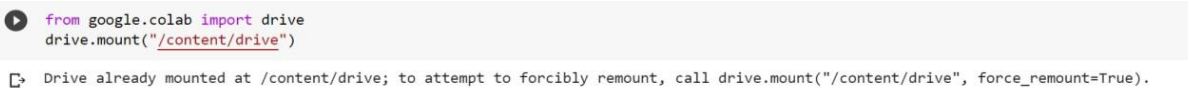
Libraries required for our model building phase are :-

1. Os
2. Numpy
3. Matplotlib
4. Keras
5. Tensorflow



```
import os
from os import listdir
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

import numpy as np
from numpy import asarray
from numpy import save

import tensorflow as tf

from keras.preprocessing.image import  load_img
from keras.preprocessing.image import img_to_array

from keras.utils import np_utils
```

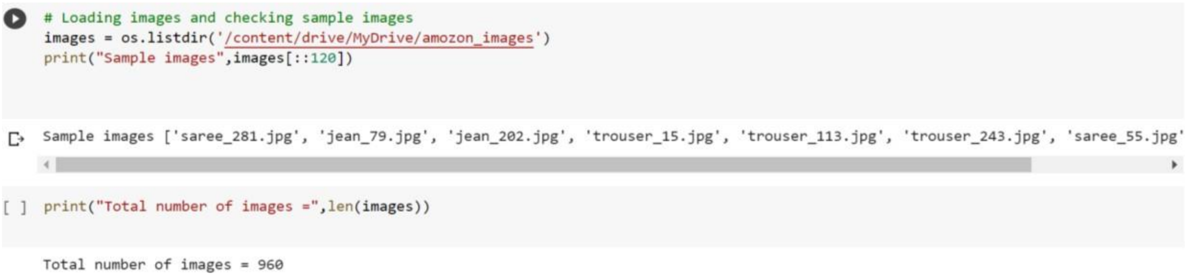
Now after loading the required libraries we mount our google drive into goole colab notebook



```
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).
```

After mounting the drive we created a variable “Image” to store all the images from our data collection process.



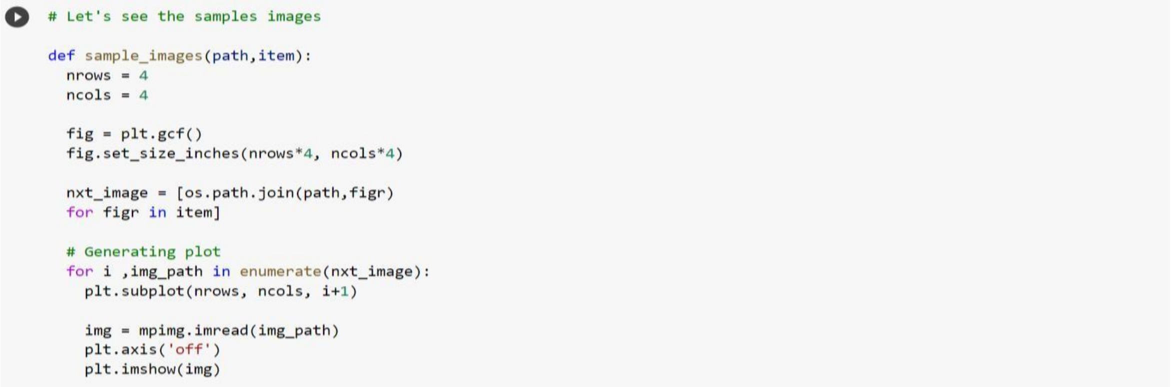
```
# Loading images and checking sample images
images = os.listdir('/content/drive/MyDrive/amozon_images')
print("Sample images",images[::120])
```

```
Sample images ['saree_281.jpg', 'jean_79.jpg', 'jean_202.jpg', 'trouser_15.jpg', 'trouser_113.jpg', 'trouser_243.jpg', 'saree_55.jpg'
```

```
print("Total number of images =",len(images))
```

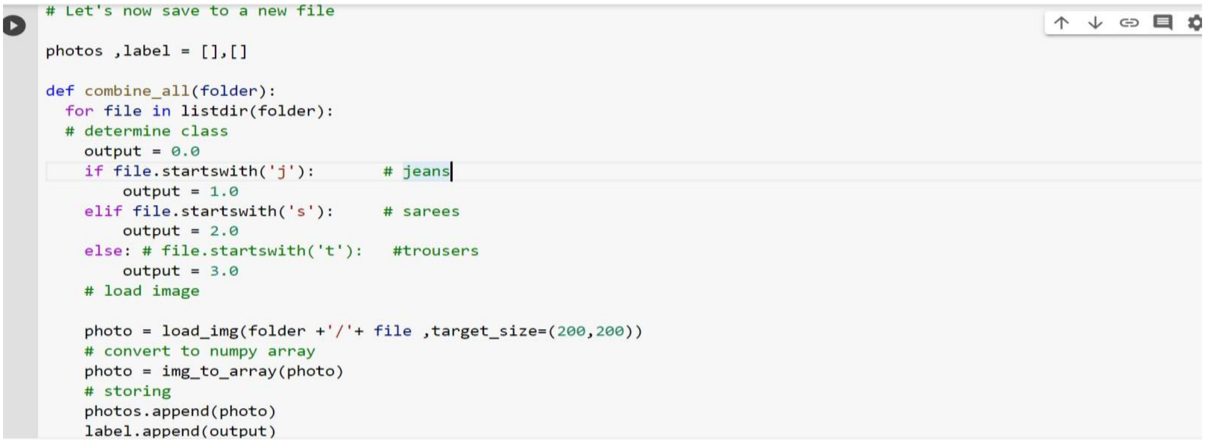
```
Total number of images = 960
```

Now we created a method to see the sample images into picture form.



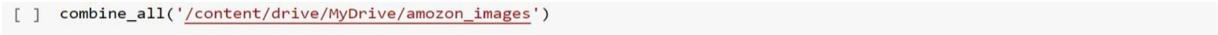
```
# Let's see the samples images

def sample_images(path,item):
  nrows = 4
  ncols = 4

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

  nxt_image = [os.path.join(path,figr)
  for figr in item]

  # Generating plot
  for i ,img_path in enumerate(nxt_image):
    plt.subplot(nrows, ncols, i+1)

    img = mpimg.imread(img_path)
    plt.axis('off')
    plt.imshow(img)
```

So after checking the images we are not in position to convert them into caler form as our system could not interpret the image. So to converting the images we again created another method. This method will find the image tag name and based on the tag name it will convert the image into an numeric form –

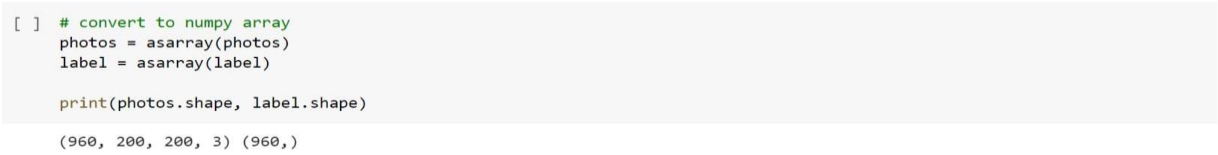


```
# Let's now save to a new file

photos ,label = [],[]

def combine_all(folder):
  for file in listdir(folder):
  # determine class
    output = 0.0
    if file.startswith('j'):        # jeans
        output = 1.0
    elif file.startswith('s'):      # sarees
        output = 2.0
    else: # file.startswith('t'):   #trousers
        output = 3.0
    # load image

    photo = load_img(folder +'/'+ file ,target_size=(200,200))
    # convert to numpy array
    photo = img_to_array(photo)
    # storing
    photos.append(photo)
    label.append(output)
```

Here we provided the input shape(200,200) to our image

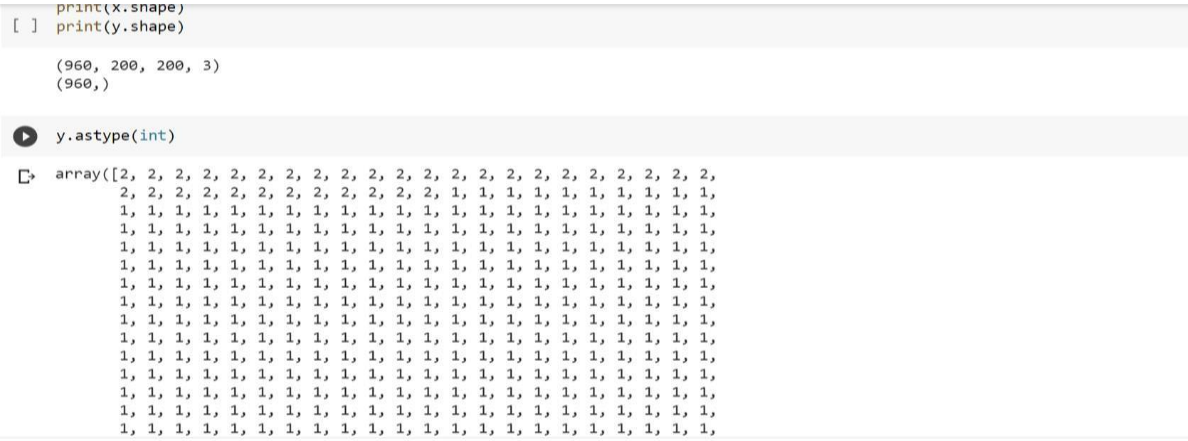


```
[ ]  combine_all('/content/drive/MyDrive/amozon_images')
```



```
[ ]  # convert to numpy array
     photos = asarray(photos)
     label = asarray(label)

     print(photos.shape, label.shape)

     (960, 200, 200, 3) (960,)
```

So after converting into array form we get the data into above format.



```
print(x.shape)
print(y.shape)

(960, 200, 200, 3)
(960,)
```

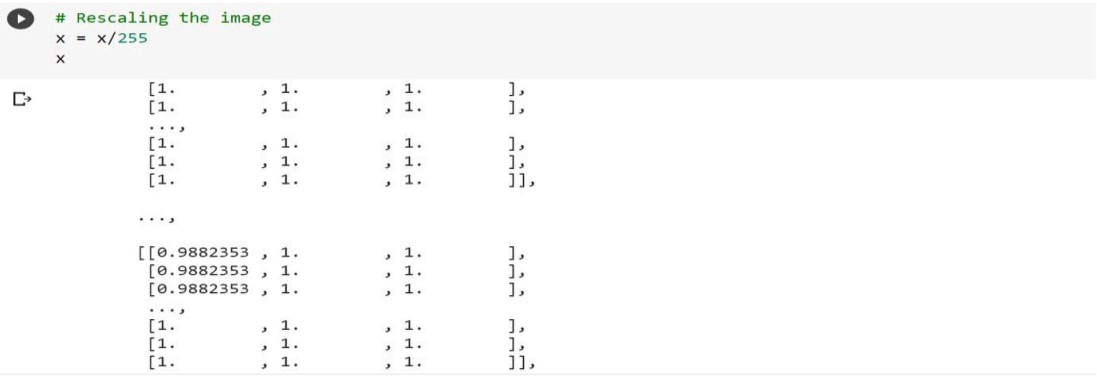
```
y.astype(int)
```

```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Now we assign independent and target variable i.e x and y respectively.

The y variable defines the category of image which is Saree, Jeans or Trouser.

After this the most important step is to rescale our independent data. We are rescaling the data into 255 scale.

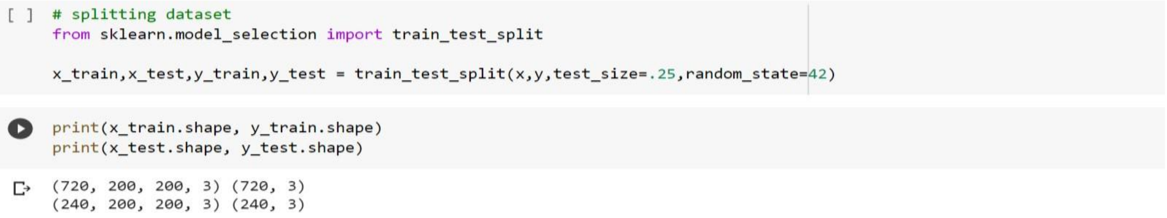


```
# Rescaling the image
x = x/255
x
```

```
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ],
          ...,
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ]],

         ...,

         [[0.9882353 , 1.        , 1.        ],
          [0.9882353 , 1.        , 1.        ],
          [0.9882353 , 1.        , 1.        ],
          ...,
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ],
          [1.        , 1.        , 1.        ]],
```

- ## Model Building Phase

So in model building phase first of all we splitted our dataset into train and test data using the train_test_split model selection method.

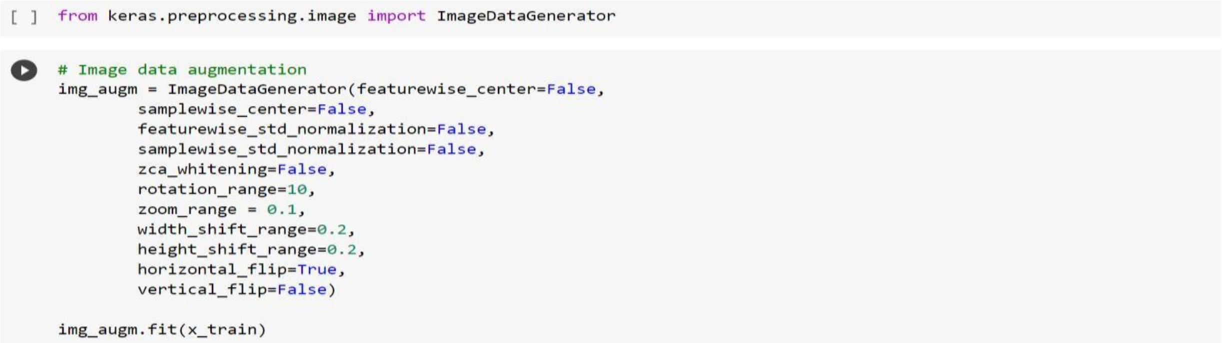


```
# splitting dataset
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.25,random_state=42)
```

```
print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
(720, 200, 200, 3) (720, 3)
(240, 200, 200, 3) (240, 3)
```

After splitting are using ImageDataGenerator technique. This technique is used for data argumentation which means to create more data from the present data. This technique will produce more data using the zoom,rotation etc methods.



```
[ ] from keras.preprocessing.image import ImageDataGenerator
```

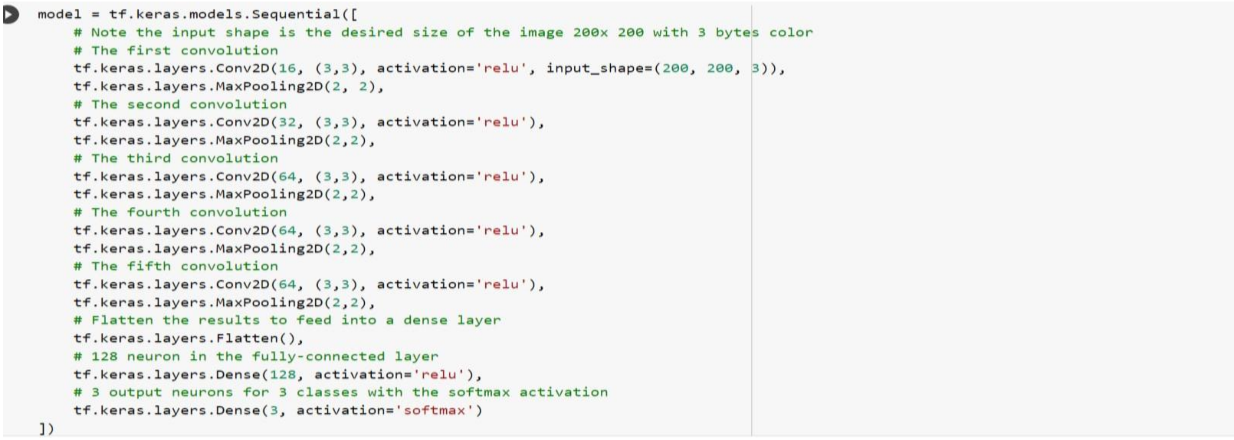
```
# Image data augmentation
img_augm = ImageDataGenerator(featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        rotation_range=10,
        zoom_range = 0.1,
        width_shift_range=0.2,
        height_shift_range=0.2,
        horizontal_flip=True,
        vertical_flip=False)

img_augm.fit(x_train)
```

So this is CNN(Convolution Neural Network) project, we are using the Sequential method to build our model which is present in tensorflow module. In this Sequential model we are using 6 step Convolution2d and MaxPooling layers.

In 1st layer of Convolution2d layer we are using 16 filters with a kernel size of (3,3) and the activation method used is “Relu activation”. After this Maxpooling layers follows up.

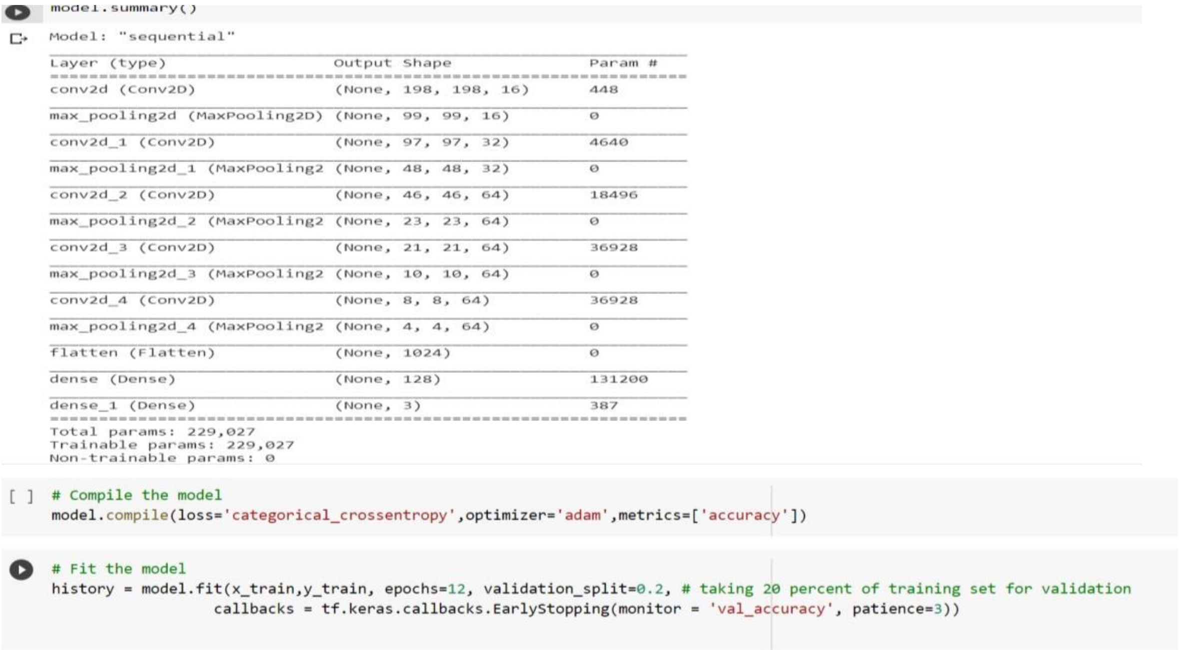
In 2nd layer of Convolution2d we increase the filter size by 2 and so on in other layers. After convolutional and maxplooing another layer present is flatteing layer-

Flattening a tensor means to remove all of the dimensions except for one. This is exactly what the Flatten layer does.



```
model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 200x 200 with 3 bytes color
    # The first convolution
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200, 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')
])
```

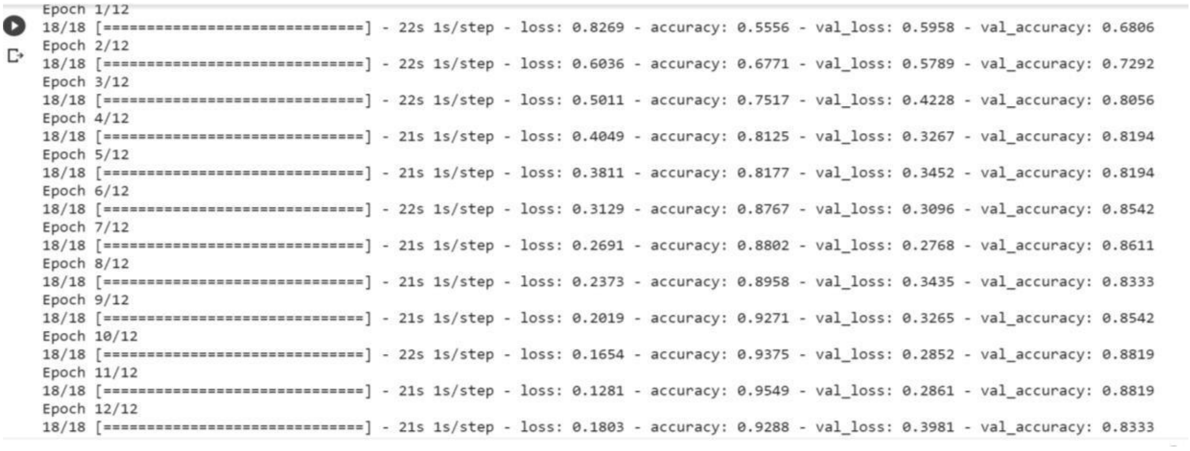
Model Summary :



```
model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 198, 198, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 99, 99, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 97, 97, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 48, 48, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 46, 46, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 23, 23, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 10, 10, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 128)               131200
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 387
=================================================================
Total params: 229,027
Trainable params: 229,027
Non-trainable params: 0
```

```
[ ] # Compile the model
    model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

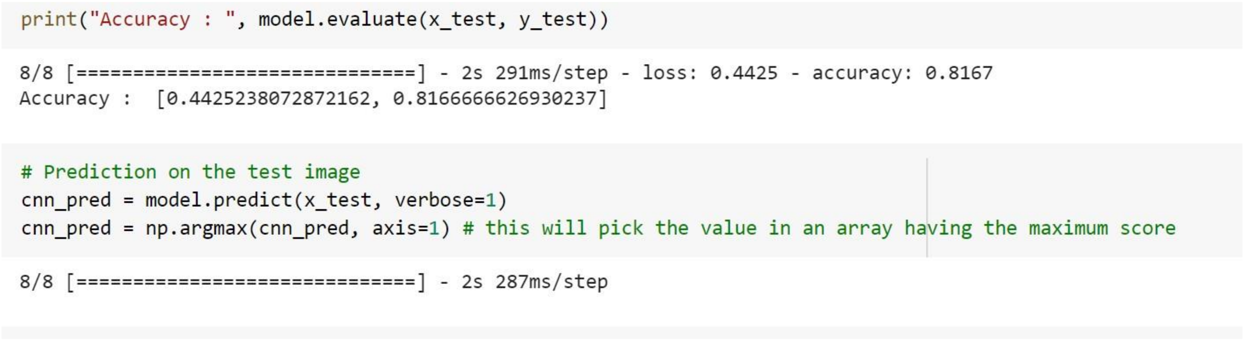
```
# Fit the model
history = model.fit(x_train,y_train, epochs=12, validation_split=0.2, # taking 20 percent of training set for validation
                    callbacks = tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy', patience=3))
```



```
Epoch 1/12
18/18 [==============================] - 22s 1s/step - loss: 0.8269 - accuracy: 0.5556 - val_loss: 0.5958 - val_accuracy: 0.6806
Epoch 2/12
18/18 [==============================] - 22s 1s/step - loss: 0.6036 - accuracy: 0.6771 - val_loss: 0.5789 - val_accuracy: 0.7292
Epoch 3/12
18/18 [==============================] - 22s 1s/step - loss: 0.5011 - accuracy: 0.7517 - val_loss: 0.4228 - val_accuracy: 0.8056
Epoch 4/12
18/18 [==============================] - 21s 1s/step - loss: 0.4049 - accuracy: 0.8125 - val_loss: 0.3267 - val_accuracy: 0.8194
Epoch 5/12
18/18 [==============================] - 21s 1s/step - loss: 0.3811 - accuracy: 0.8177 - val_loss: 0.3452 - val_accuracy: 0.8194
Epoch 6/12
18/18 [==============================] - 22s 1s/step - loss: 0.3129 - accuracy: 0.8767 - val_loss: 0.3096 - val_accuracy: 0.8542
Epoch 7/12
18/18 [==============================] - 21s 1s/step - loss: 0.2691 - accuracy: 0.8802 - val_loss: 0.2768 - val_accuracy: 0.8611
Epoch 8/12
18/18 [==============================] - 21s 1s/step - loss: 0.2373 - accuracy: 0.8958 - val_loss: 0.3435 - val_accuracy: 0.8333
Epoch 9/12
18/18 [==============================] - 21s 1s/step - loss: 0.2019 - accuracy: 0.9271 - val_loss: 0.3265 - val_accuracy: 0.8542
Epoch 10/12
18/18 [==============================] - 22s 1s/step - loss: 0.1654 - accuracy: 0.9375 - val_loss: 0.2852 - val_accuracy: 0.8819
Epoch 11/12
18/18 [==============================] - 21s 1s/step - loss: 0.1281 - accuracy: 0.9549 - val_loss: 0.2861 - val_accuracy: 0.8819
Epoch 12/12
18/18 [==============================] - 21s 1s/step - loss: 0.1803 - accuracy: 0.9288 - val_loss: 0.3981 - val_accuracy: 0.8333
```

Model Evaluations :

Accuracy



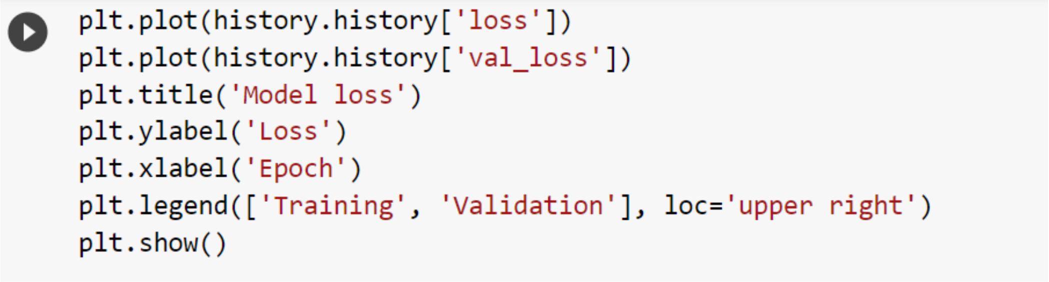
```
print("Accuracy : ", model.evaluate(x_test, y_test))
```

```
8/8 [==============================] - 2s 291ms/step - loss: 0.4425 - accuracy: 0.8167
Accuracy :  [0.4425238072872162, 0.8166666626930237]
```

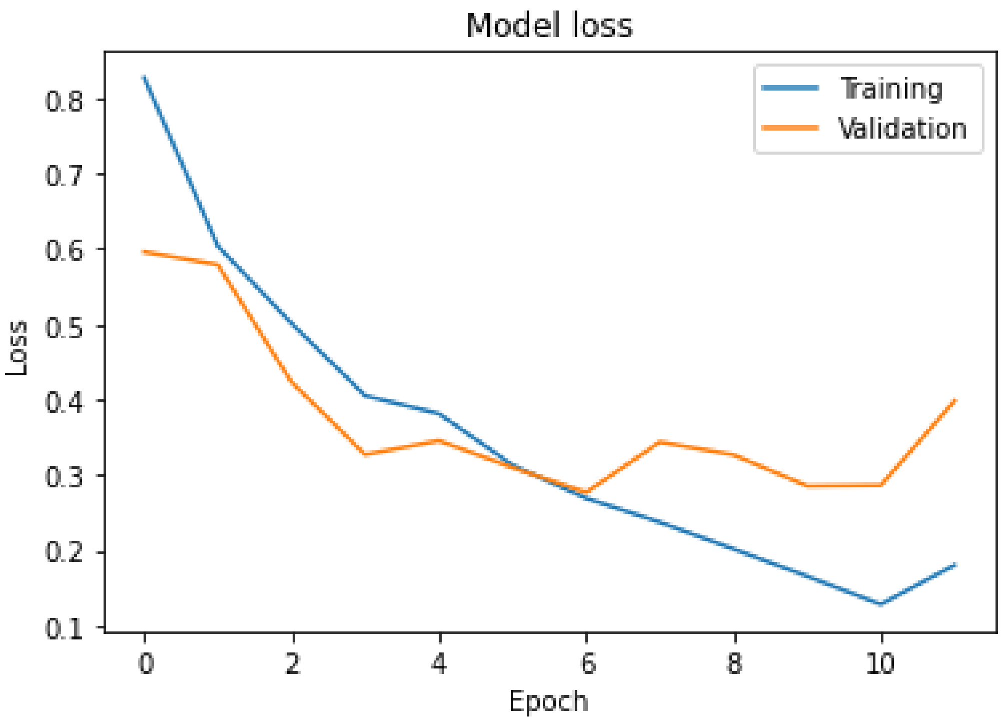
```
# Prediction on the test image
cnn_pred = model.predict(x_test, verbose=1)
cnn_pred = np.argmax(cnn_pred, axis=1) # this will pick the value in an array having the maximum score
```

```
8/8 [==============================] - 2s 287ms/step
```

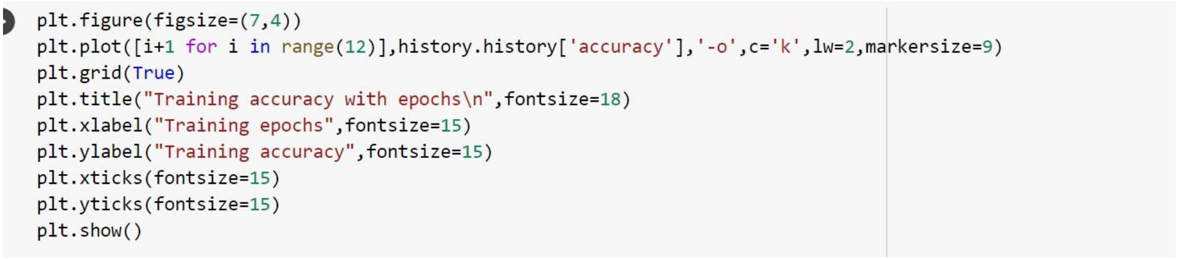
Using the line plots we are checking the training and validation loss-



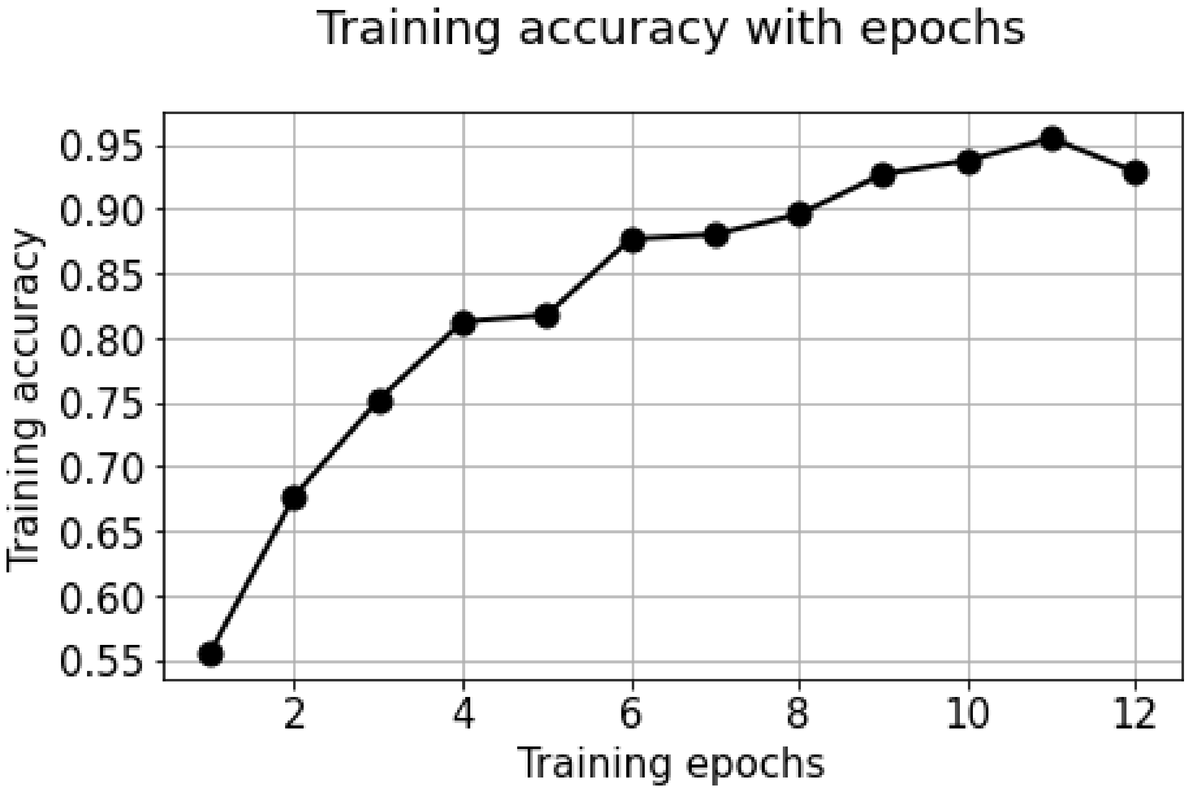
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.show()
```



From the above graph we find that as the epochs increases loss values also decrease and training and validation loss are almost equal at 5 epoches. After that again the validation loss starts to increase.

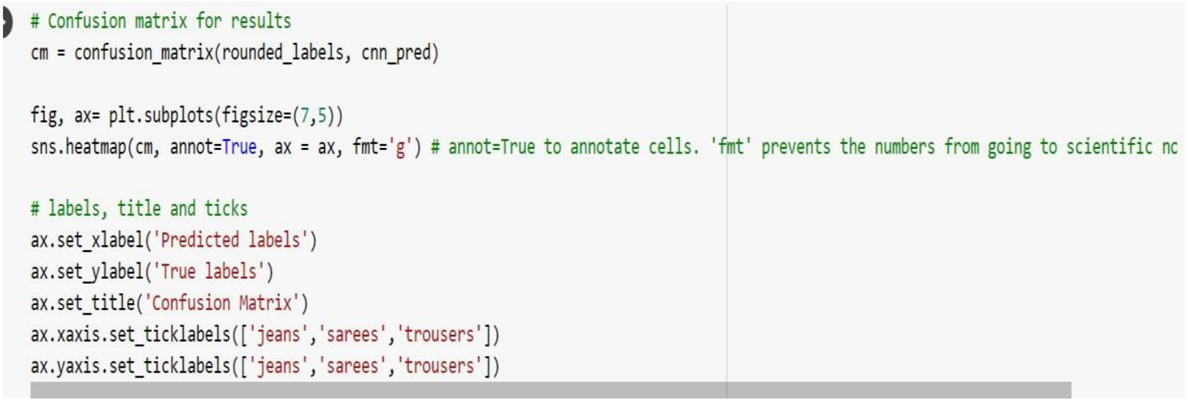


```
plt.figure(figsize=(7,4))
plt.plot([i+1 for i in range(12)],history.history['accuracy'],'-o',c='k',lw=2,markersize=9)
plt.grid(True)
plt.title("Training accuracy with epochs\n",fontsize=18)
plt.xlabel("Training epochs",fontsize=15)
plt.ylabel("Training accuracy",fontsize=15)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.show()
```

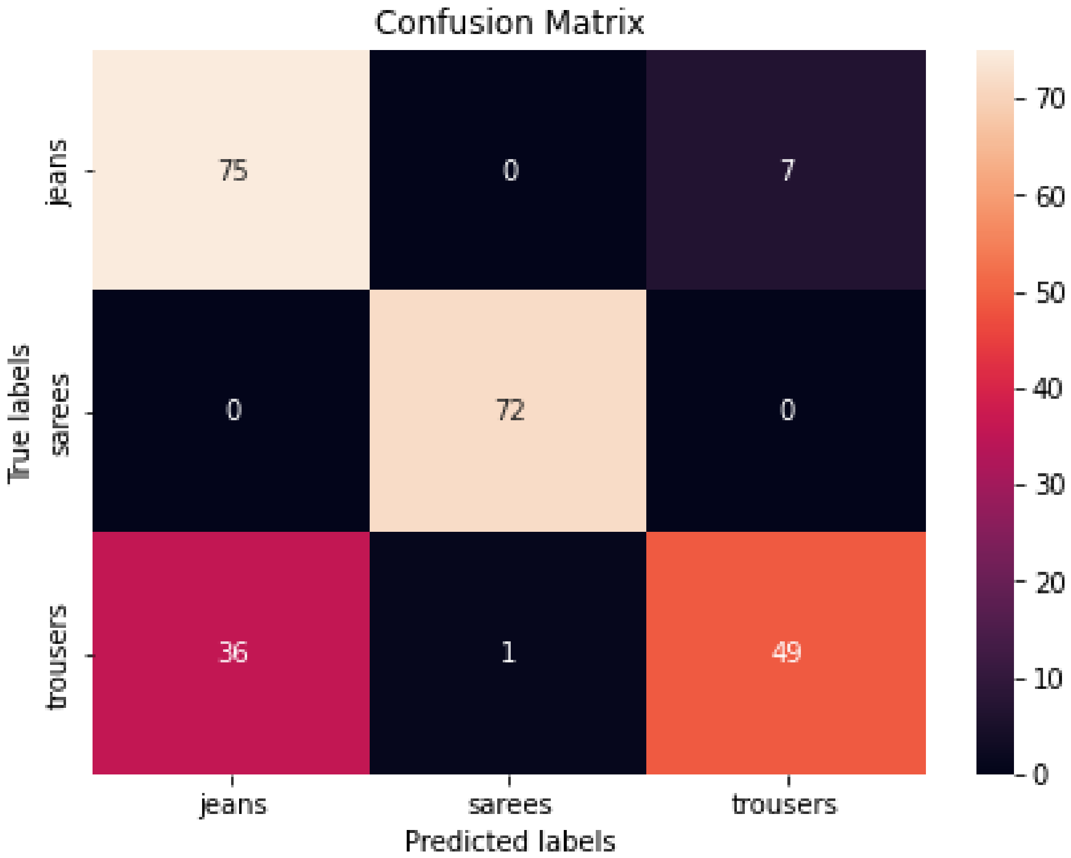


As clear from the above curve that there is increase in accuracy of model with increase of epochs. But 11 epochs are giving the best accuracy from model.

Confusion Matrix :-



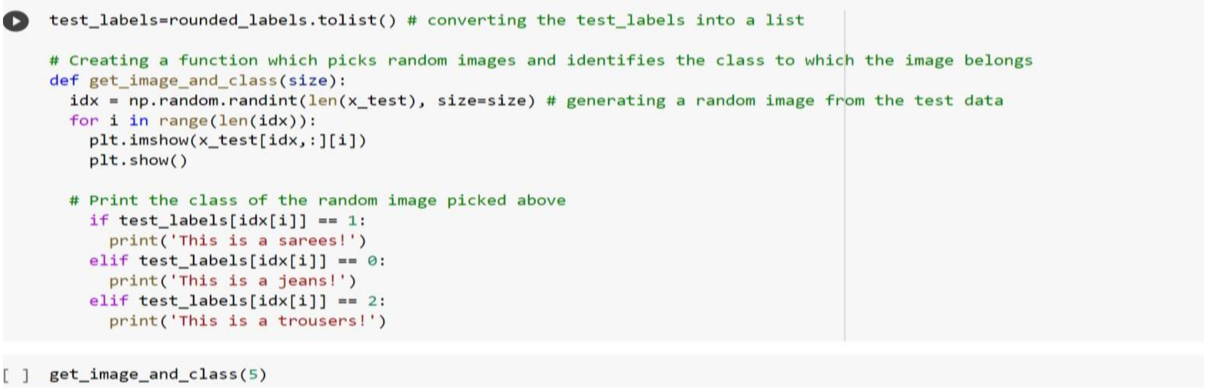
```
# Confusion matrix for results
cm = confusion_matrix(rounded_labels, cnn_pred)

fig, ax= plt.subplots(figsize=(7,5))
sns.heatmap(cm, annot=True, ax = ax, fmt='g') # annot=True to annotate cells. 'fmt' prevents the numbers from going to scientific nc

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(['jeans','sarees','trousers'])
ax.yaxis.set_ticklabels(['jeans','sarees','trousers'])
```



	jeans	sarees	trousers
jeans	75	0	7
sarees	0	72	0
trousers	36	1	49

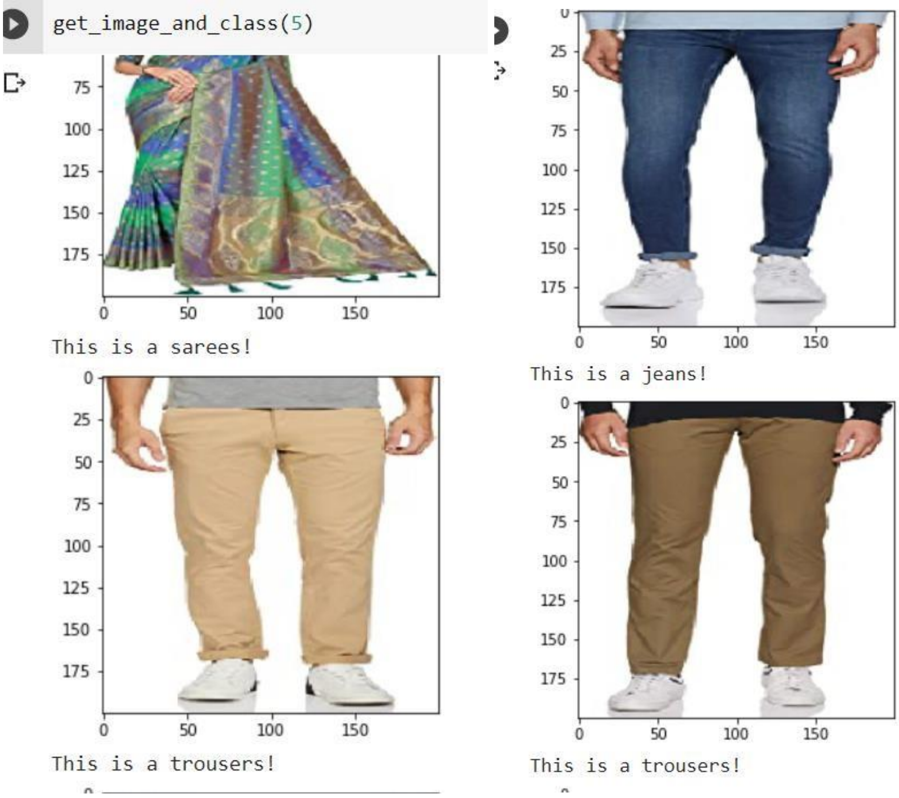
In Confusion matrix, here we are getting good result in classification of sarres but our model is somehow lacking in accurate classification between Jeans and Trousers.

Predictions:



```
test_labels=rounded_labels.tolist() # converting the test_labels into a list

# Creating a function which picks random images and identifies the class to which the image belongs
def get_image_and_class(size):
  idx = np.random.randint(len(x_test), size=size) # generating a random image from the test data
  for i in range(len(idx)):
    plt.imshow(x_test[idx,:][i])
    plt.show()

  # Print the class of the random image picked above
    if test_labels[idx[i]] == 1:
      print('This is a sarees!')
    elif test_labels[idx[i]] == 0:
      print('This is a jeans!')
    elif test_labels[idx[i]] == 2:
      print('This is a trousers!')
```

```
[ ]  get_image_and_class(5)
```



```
get_image_and_class(5)
```

```
This is a sarees!
```

```
This is a trousers!
```

```
This is a jeans!
```

```
This is a trousers!
```

CONCLUSION

The key finding and the conclusion of the study :-

1. The image data was collected using Webscrapping from Amazon for Jeans ,Sarees and Trousers.
2. We have used Convolutional Neural Network for the project and giving us the accuracy of 81% with 0.4 log loss.
3. The model is working fine except in case of Jeans and Trousers.

Thankyou