FLIP ROBO

IMAGE CLASSIFICATION MODEL

Submitted by:

Bhushan Kumar Sharma,

Intern Data Scientist

ACKNOWLEDGMENT

*****

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with this dataset and it has helped me to improve my model building skills.

A huge thanks to my academic team “Datatrained” who is the reason behind what I am today. Last but not least my parents who have been my backbone in every step of my life. And also thank to many other persons who has helped me directly or indirectly to complete the project.

Following are the external references which I used:

www.geeksforgeeks.org

www.stackoverflow.com

www.w3school.com

www.google.com

Datatrained Lectures

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. I have tried to take an exposure of how an end to end project is developed in this field.

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

Data Collection Phase: In this section, I have scraped images from e-commerce portal, Amazon.com. The clothing categories used for scraping is:

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

I have scraped images of these 3 categories and build our data from it. That data is provided as an input to your deep learning problem.

In the image scraping, I have scraped 303 image for saree, 305 images for Trouser and 314 images for Jeans and also insured the good quality of data.

Model Building Phase: After the completed of data collection and preparation, I have built an image classification model that will classify between these 3 categories mentioned above.

➢ Conceptual Background of the Domain Problem

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

➢ Review of Literature

- Recently, image classification is growing and becoming a trend among technology developers especially with the growth of data in different parts of industry such as e-commerce, automotive, healthcare, and gaming.

➢ Motivation for the Problem Undertaken

- The most obvious example of this technology is applied to Facebook. Facebook now can detect up to 98% accuracy to identify your face with only a few tagged images and classified it into your Facebook's album. The technology itself almost beats the ability of human in image classification or recognition. One of the dominant approaches for this technology is deep learning.

The problem was undertaken to classify images efficiently using best deep learning algorithms and data augmentation techniques.

➢ Mathematical/ Analytical Modelling of the Problem

The data was collected by using web scrapping. In web scrapping I have used selenium library. In this dataset image can be classified in three category 'Saree', 'Trouser' and 'Jeans', As it is containing more than two class So clearly it is a multi-classification problem and I have use Deep Learning Model while building the model.

I have applied CNN Architecture technique to built deep learning model for this problem.

- **Data Sources and their formats**

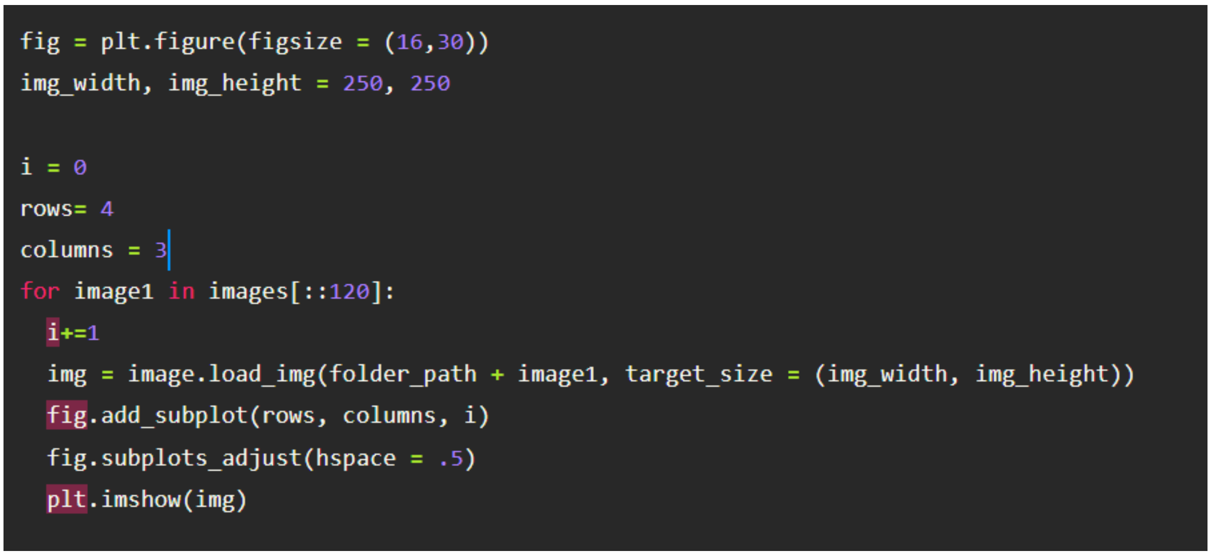
Scraped dataset is having 303 image of Saree and 3.5 images for trouser and 314 images for Jeans, in this dataset Classification of these image would be our target column and I have used a self-created function to categorize these image.

- **Data Information:**

1. **Saree**: 303 images
2. **Trouser**: 305 images
3. **Jeans**: 314 images

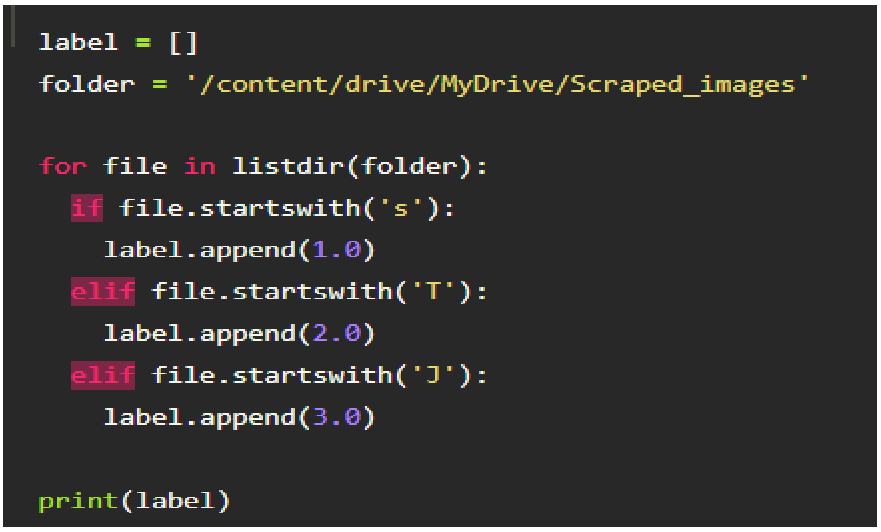
➢ Data Pre-processing:

- *For checking random images*



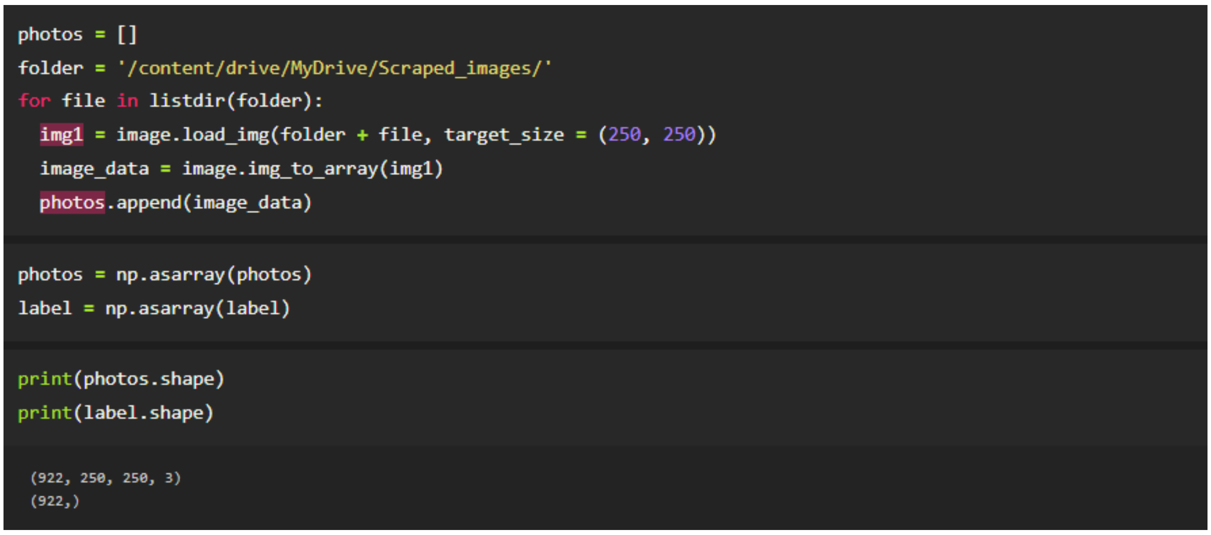
```
fig = plt.figure(figsize = (16,30))
img_width, img_height = 250, 250

i = 0
rows= 4
columns = 3
for image1 in images[::120]:
  i+=1
  img = image.load_img(folder_path + image1, target_size = (img_width, img_height))
  fig.add_subplot(rows, columns, i)
  fig.subplots_adjust(hspace = .5)
  plt.imshow(img)
```

- *Created labels (Target) for respective images*



```
label = []
folder = '/content/drive/MyDrive/Scraped_images'

for file in listdir(folder):
  if file.startswith('s'):
    label.append(1.0)
  elif file.startswith('T'):
    label.append(2.0)
  elif file.startswith('J'):
    label.append(3.0)

print(label)
```

- *Loading images from Google drive:*

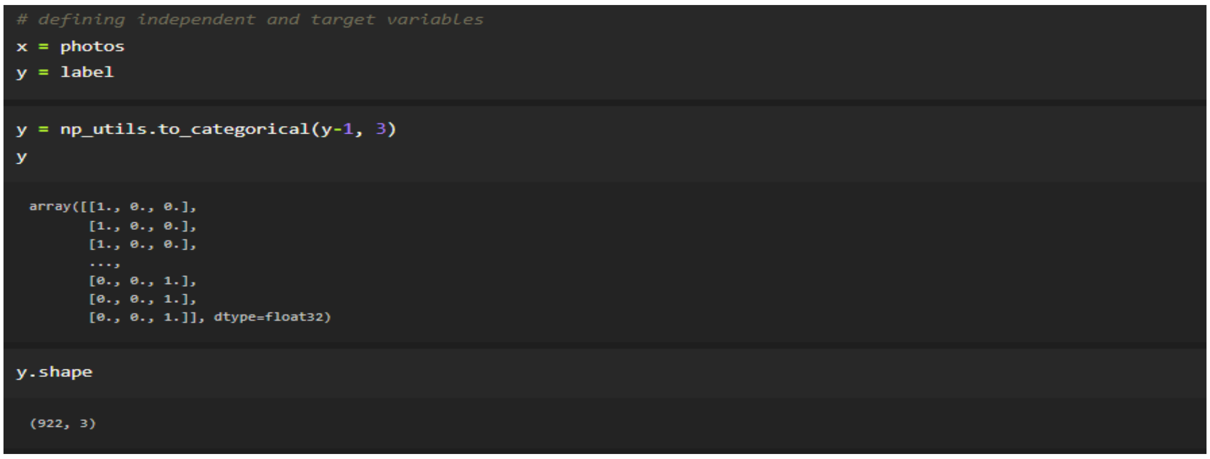


```
photos = []
folder = '/content/drive/MyDrive/Scraped_images/'
for file in listdir(folder):
  img1 = image.load_img(folder + file, target_size = (250, 250))
  image_data = image.img_to_array(img1)
  photos.append(image_data)
```

```
photos = np.asarray(photos)
label = np.asarray(label)
```

```
print(photos.shape)
print(label.shape)
```

```
(922, 250, 250, 3)
(922,)
```



```
# defining independent and target variables
x = photos
y = label
```

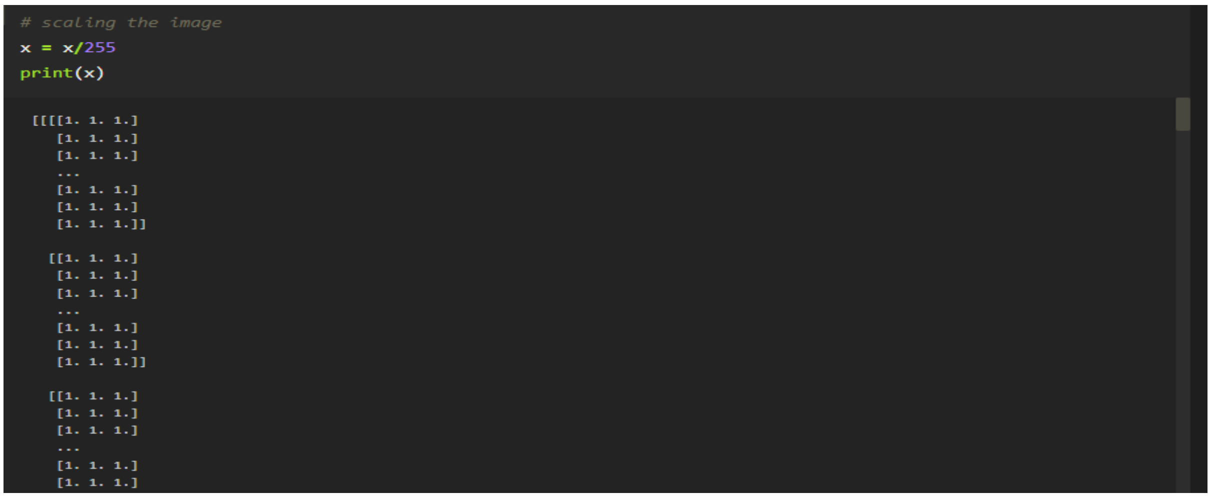
```
y = np_utils.to_categorical(y-1, 3)
y
```

```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       ...,
       [0., 0., 1.],
       [0., 0., 1.],
       [0., 0., 1.]], dtype=float32)
```

```
y.shape
```

```
(922, 3)
```

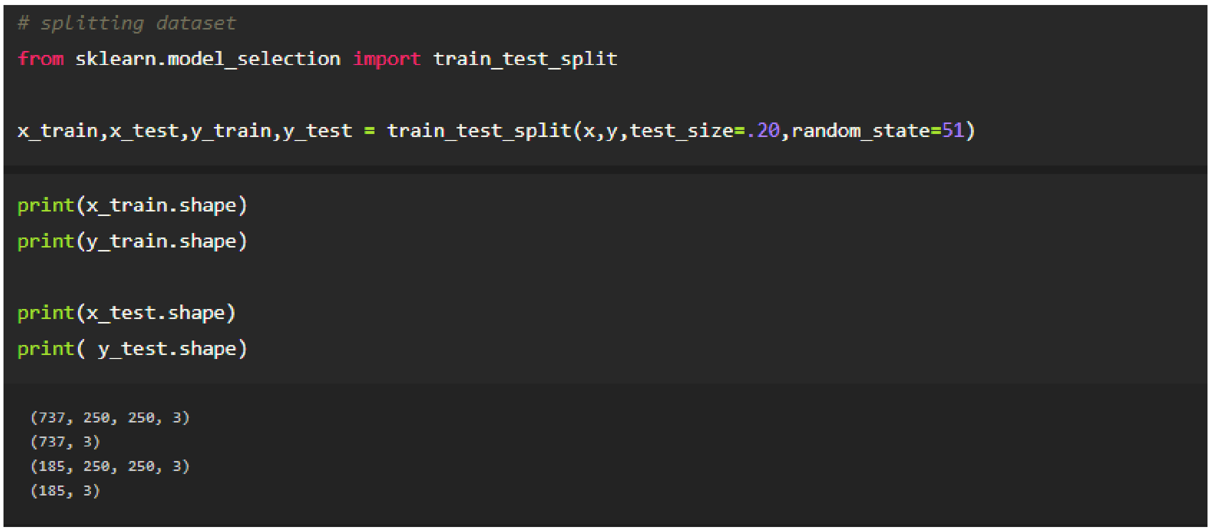
- *Feature need to be scaled*



```
# scaling the image
x = x/255
print(x)
```

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

  [[1. 1. 1.]
   [1. 1. 1.]
   [1. 1. 1.]
   ...
   [1. 1. 1.]
   [1. 1. 1.]
   [1. 1. 1.]]

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

- *Splitting dataset into train and test dataset*

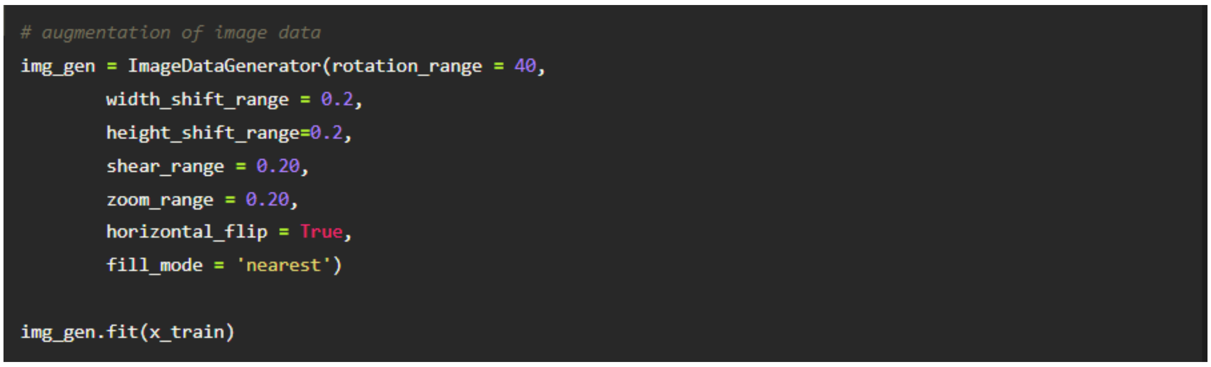


```
# splitting dataset
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.20,random_state=51)
```

```
print(x_train.shape)
print(y_train.shape)

print(x_test.shape)
print( y_test.shape)
```

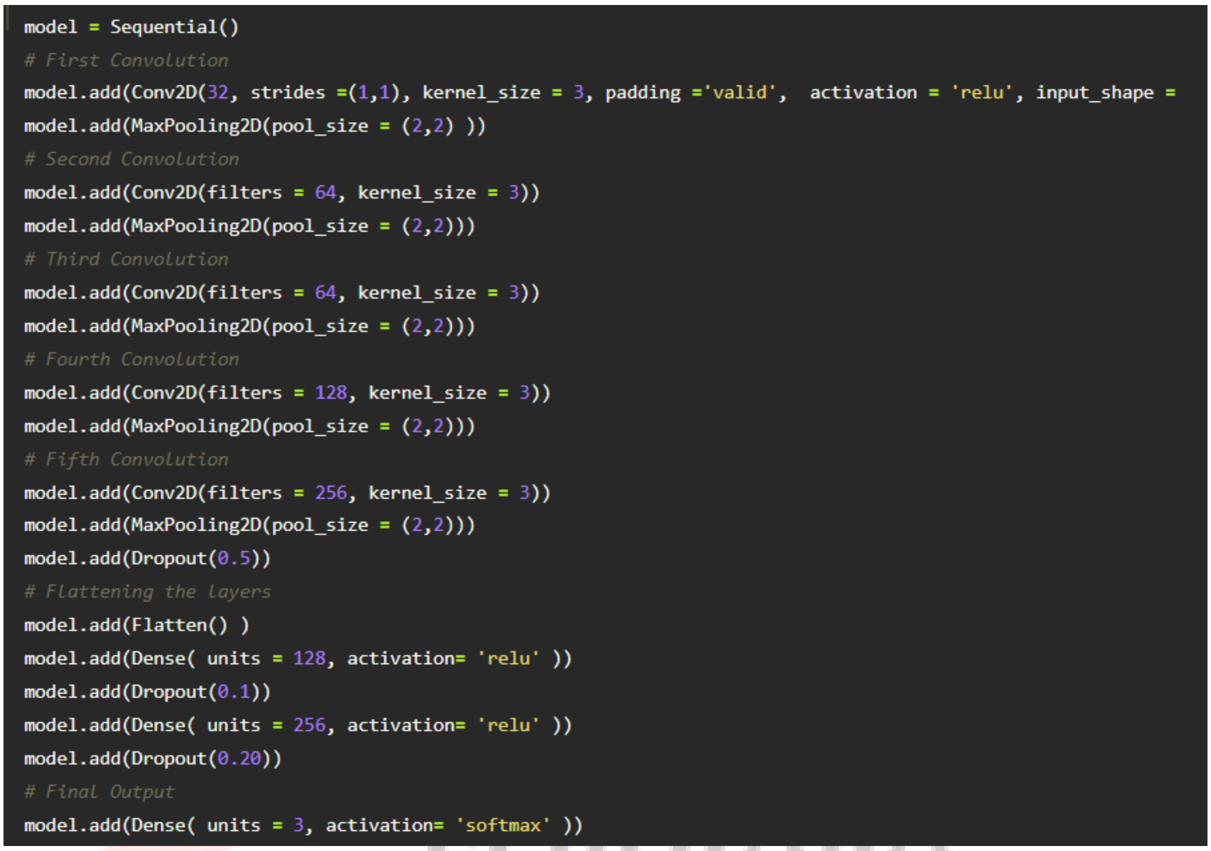
```
(737, 250, 250, 3)
(737, 3)
(185, 250, 250, 3)
(185, 3)
```

- *Taking different view by generating different image view from one single image*



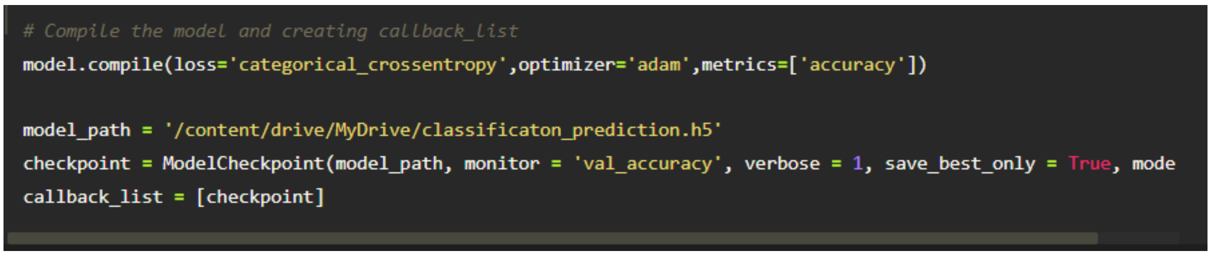
```
# augmentation of image data
img_gen = ImageDataGenerator(rotation_range = 40,
        width_shift_range = 0.2,
        height_shift_range=0.2,
        shear_range = 0.20,
        zoom_range = 0.20,
        horizontal_flip = True,
        fill_mode = 'nearest')

img_gen.fit(x_train)
```

➢ Deep Learning:



```
model = Sequential()
# First Convolution
model.add(Conv2D(32, strides =(1,1), kernel_size = 3, padding ='valid',  activation = 'relu', input_shape =
model.add(MaxPooling2D(pool_size = (2,2) ))
# Second Convolution
model.add(Conv2D(filters = 64, kernel_size = 3))
model.add(MaxPooling2D(pool_size = (2,2)))
# Third Convolution
model.add(Conv2D(filters = 64, kernel_size = 3))
model.add(MaxPooling2D(pool_size = (2,2)))
# Fourth Convolution
model.add(Conv2D(filters = 128, kernel_size = 3))
model.add(MaxPooling2D(pool_size = (2,2)))
# Fifth Convolution
model.add(Conv2D(filters = 256, kernel_size = 3))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.5))
# Flattening the layers
model.add(Flatten() )
model.add(Dense( units = 128, activation= 'relu' ))
model.add(Dropout(0.1))
model.add(Dense( units = 256, activation= 'relu' ))
model.add(Dropout(0.20))
# Final Output
model.add(Dense( units = 3, activation= 'softmax' ))
```

Note: In the deep learning model, I have applied 5 convolutional layers and 4 directly connected layers to predict target classification of give image.



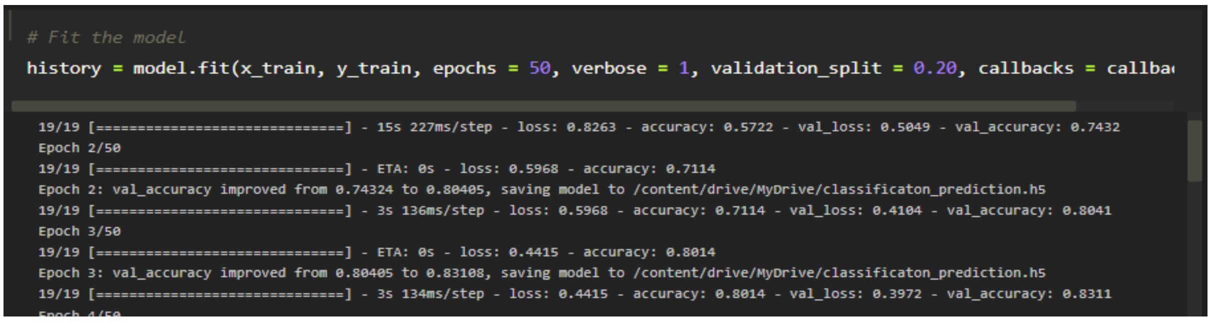
```
# Compile the model and creating callback_list
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])

model_path = '/content/drive/MyDrive/classificaton_prediction.h5'
checkpoint = ModelCheckpoint(model_path, monitor = 'val_accuracy', verbose = 1, save_best_only = True, mode
callback_list = [checkpoint]
```

Here, loss would be calculated by categorical_crossentropy and accuracy would be responsible to calculate the accuracy of the model.

Note:

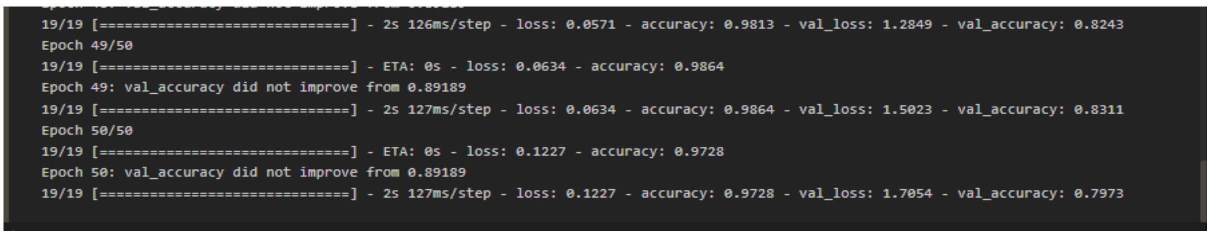
Checkpoint: it is used here, to save the model at point of time, when validation accuracy is highest.

As much validation accuracy would be greater , it would be better to predict the classification of the new type of image.

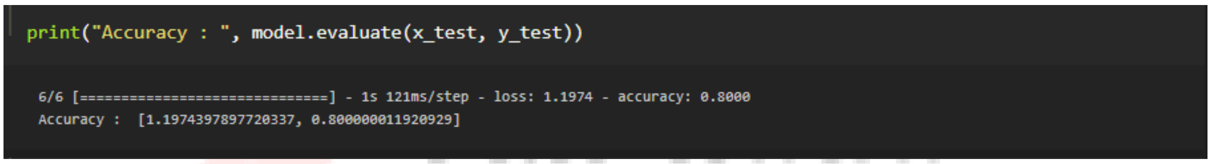


```
# Fit the model
history = model.fit(x_train, y_train, epochs = 50, verbose = 1, validation_split = 0.20, callbacks = callba
```

```
19/19 [==============================] - 15s 227ms/step - loss: 0.8263 - accuracy: 0.5722 - val_loss: 0.5049 - val_accuracy: 0.7432
Epoch 2/50
19/19 [==============================] - ETA: 0s - loss: 0.5968 - accuracy: 0.7114
Epoch 2: val_accuracy improved from 0.74324 to 0.80405, saving model to /content/drive/MyDrive/classificaton_prediction.h5
19/19 [==============================] - 3s 136ms/step - loss: 0.5968 - accuracy: 0.7114 - val_loss: 0.4104 - val_accuracy: 0.8041
Epoch 3/50
19/19 [==============================] - ETA: 0s - loss: 0.4415 - accuracy: 0.8014
Epoch 3: val_accuracy improved from 0.80405 to 0.83108, saving model to /content/drive/MyDrive/classificaton_prediction.h5
19/19 [==============================] - 3s 134ms/step - loss: 0.4415 - accuracy: 0.8014 - val_loss: 0.3972 - val_accuracy: 0.8311
```



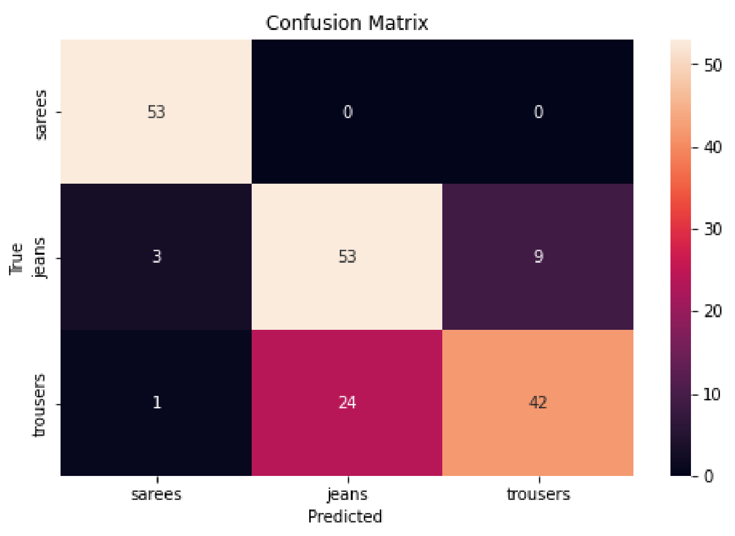
```
19/19 [==============================] - 2s 126ms/step - loss: 0.0571 - accuracy: 0.9813 - val_loss: 1.2849 - val_accuracy: 0.8243
Epoch 49/50
19/19 [==============================] - ETA: 0s - loss: 0.0634 - accuracy: 0.9864
Epoch 49: val_accuracy did not improve from 0.89189
19/19 [==============================] - 2s 127ms/step - loss: 0.0634 - accuracy: 0.9864 - val_loss: 1.5023 - val_accuracy: 0.8311
Epoch 50/50
19/19 [==============================] - ETA: 0s - loss: 0.1227 - accuracy: 0.9728
Epoch 50: val_accuracy did not improve from 0.89189
19/19 [==============================] - 2s 127ms/step - loss: 0.1227 - accuracy: 0.9728 - val_loss: 1.7054 - val_accuracy: 0.7973
```



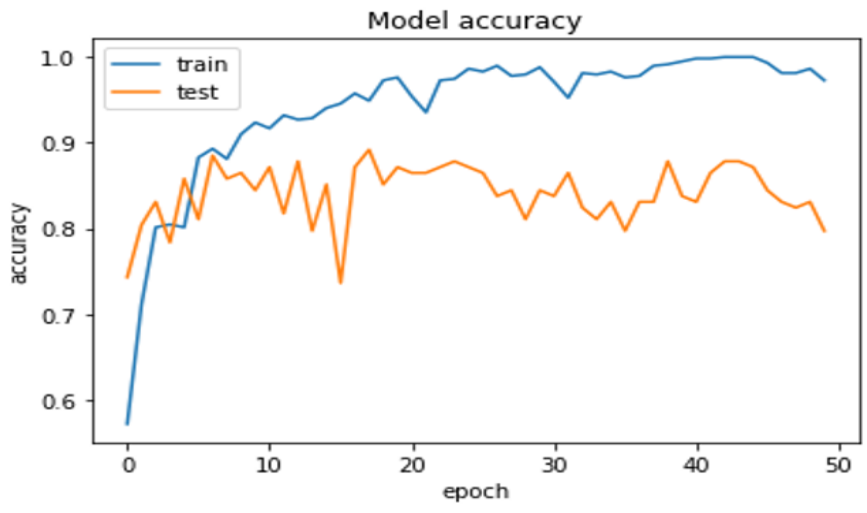
```
print("Accuracy : ", model.evaluate(x_test, y_test))
```

```
6/6 [==============================] - 1s 121ms/step - loss: 1.1974 - accuracy: 0.8000
Accuracy :  [1.1974397897720337, 0.800000011920929]
```

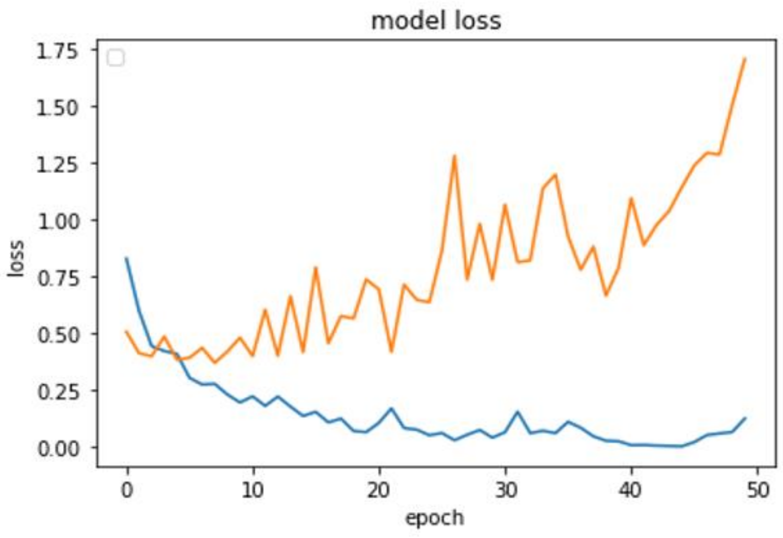
50 Epochs are used here, and saved the model at a point where validation accuracy is maximum.



By this confusion matrix, we can observe that, model is well working of sarees images, but it is bit confused sin case of Jeans and trouser.

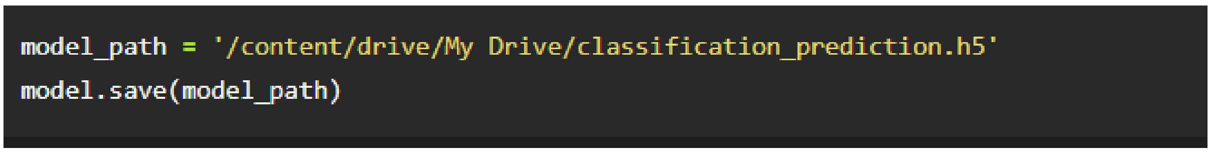


As our model is not giving 100 per cent accuracy, accordingly it is having some difference.



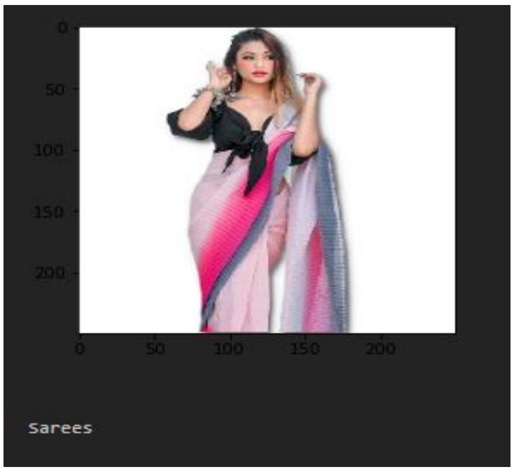
As our model is not giving 100 per cent accuracy, accordingly it is having some difference in data loss.

➢ **Model Saving**

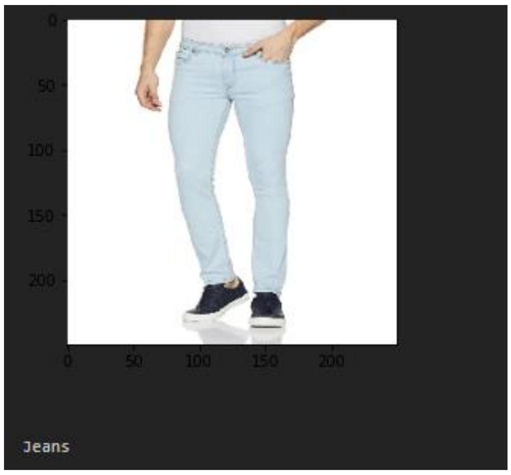


```
model_path = '/content/drive/My Drive/classification_prediction.h5'
model.save(model_path)
```

➢ **Conclusion sof the model**

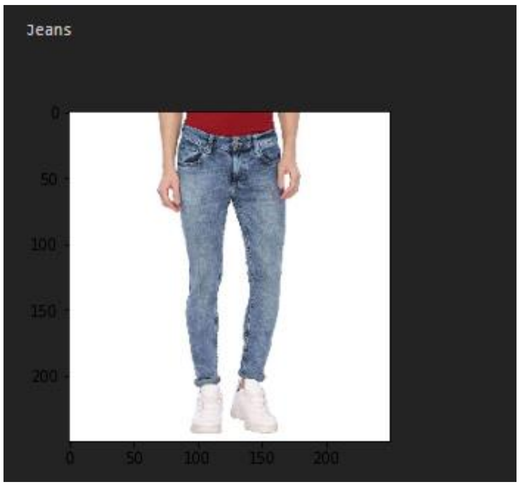


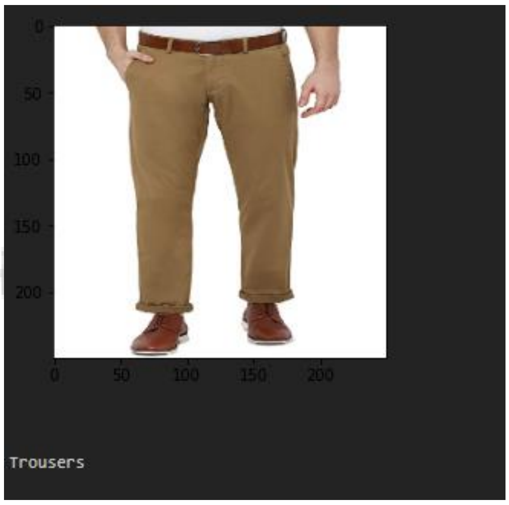
Sarees



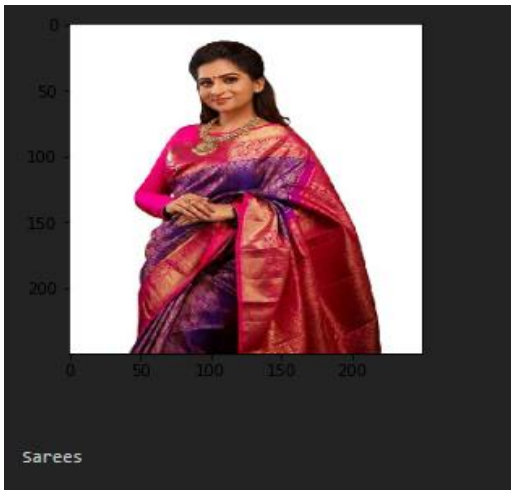
Jeans

Jeans





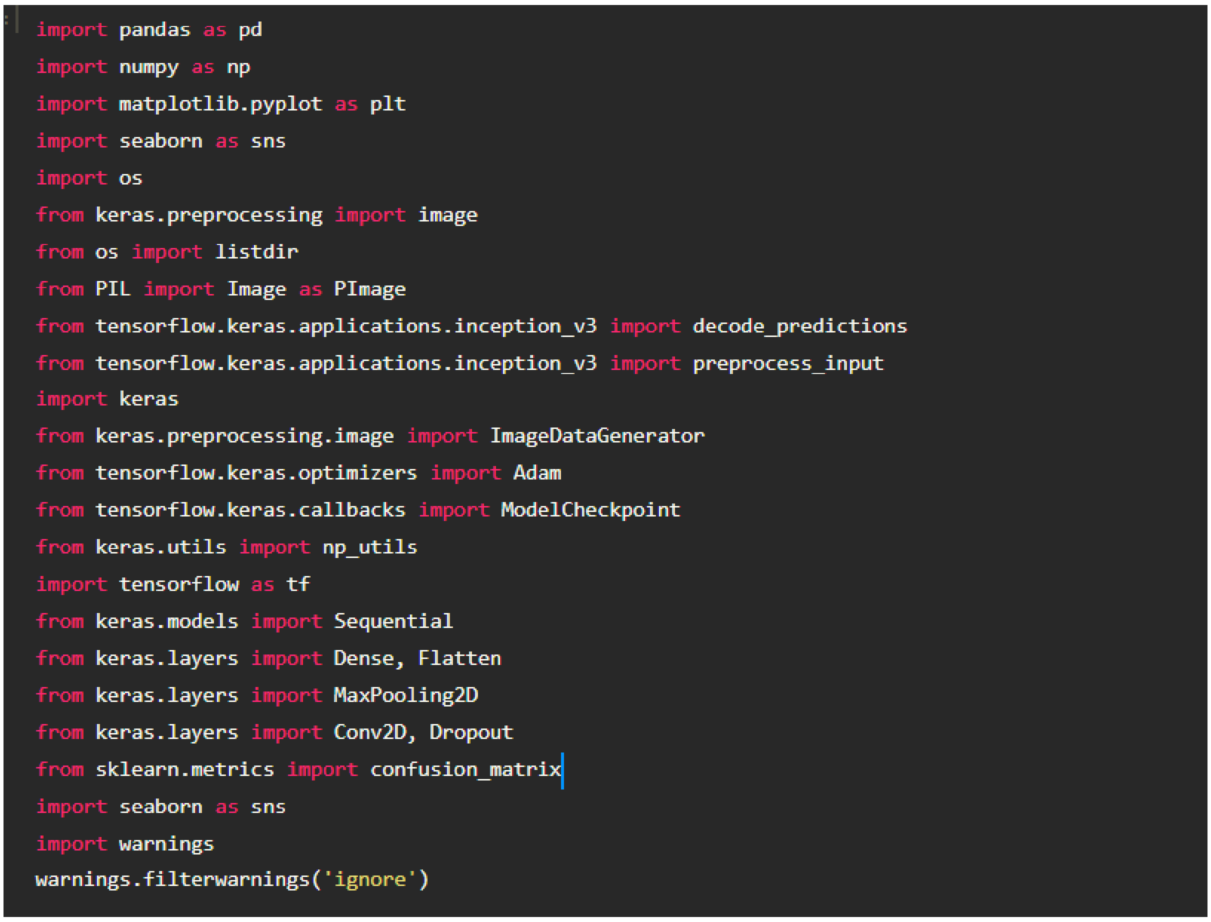
Trousers



Sarees

➢ Hardware and Software Requirements and Tools Used

All used libraries:



```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
from keras.preprocessing import image
from os import listdir
from PIL import Image as PImage
from tensorflow.keras.applications.inception_v3 import decode_predictions
from tensorflow.keras.applications.inception_v3 import preprocess_input
import keras
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint
from keras.utils import np_utils
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import MaxPooling2D
from keras.layers import Conv2D, Dropout
from sklearn.metrics import confusion_matrix
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Pandas: This library used for dataframe operations

Numpy: This library gives statistical computation for smooth functioning

Matplotlib: Used for visualization

Seaborn: This library is also used for visualization

Sklearn: This library having so many machine learning module and we can import them from this library

Warning: To avoid unwanted warning shows in the output

Keras: This library is used to create the CNN Model

Tensorflow: Working in backend of Keras library

OS: It is used to work with file system

I am giving this requirement and tool used, based on my laptop configuration.

Operating System:	**Window 11**
RAM:	**8 GB**
Processor:	**i5 10th Generation**
Software:	**Jupyter Notebook,**

- **Observations from the whole problem.**

i) Model is working fine in case sarees but need to train the model with more images for jeans and trousers.

ii) Almost same number of images is processed for each classification.

- **Learning Outcomes of the Study in respect of Data Science**

My learnings: - the power of visualization is helpful for the understanding of data into the graphical representation its help me to understand that what data is trying to say,

Deep learning using CNN network, I have used in this dataset and save the best model where validation accuracy is maximum.

- **Limitations of this work and Scope for Future Work**

Need to train model with more image for jeans and trouser.