FLIP ROBO

IMAGE SCRAPING AND CLASSIFICATION

Submitted by: **SHANTY EMERSON**

ACKNOWLEDGMENT

I would like to express my special thanks of gratitude to FLIP ROBO TECHNOLOGIES. as well as our SME Shubham Yadav who gave me the golden opportunity to do this wonderful project on the topic Project Image Scraping and Classification.

INTRODUCTION

Business Problem Framing

We have scraped total 1795 data from each Sarees, Trousers and Jeans from Amazon website and train model using Deep Learning to predict the category of the image.
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 you 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. This task is divided into two phases: Data Collection and Model Building. Data Collection Phase: In this section, you 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 category. 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. Remember, in case of deep learning models, the data needs to be big for building a good performing model. More the data, better the results. Model Building Phase: After the data collection and preparation is done, you need to build an image classification model that will classify between these 3 categories mentioned above. You can play around with optimizers and learning rates for improving your model’s performance.

Analytical Problem Framing

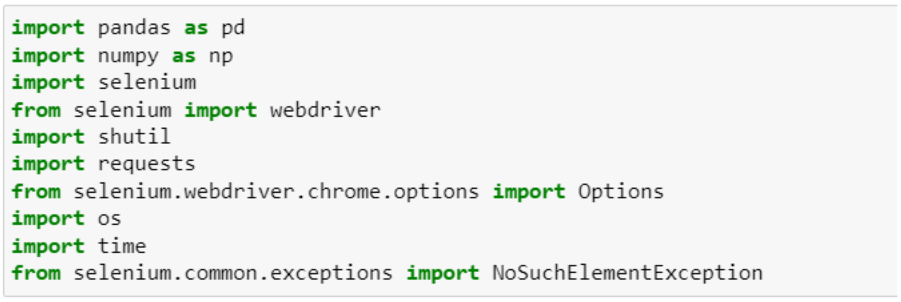
Methodology represents a description about the framework that is undertaken. It consists of various milestones that need to be achieved in order to fulfil the objective.

The following steps represents stepwise tasks that need to be completed:

- Data Scraping
- Data Loading
- Data Visualization
- Model Building
- Model Evaluation
- Saving the model

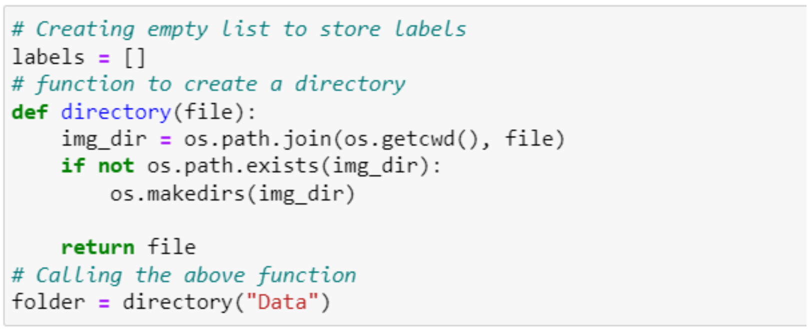
Data Collection

First, we have imported all the required libraries where we need for data collection.



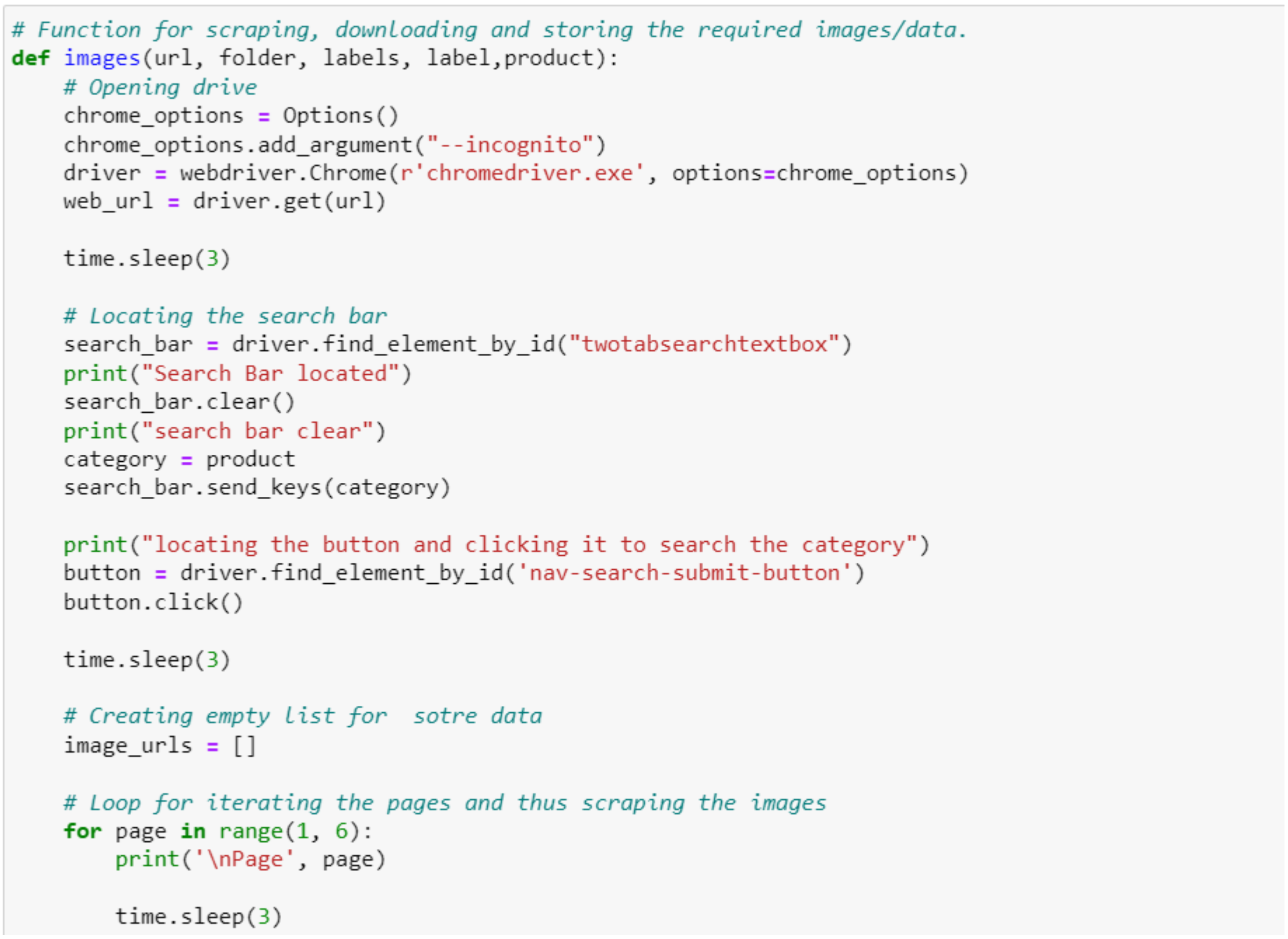
```
import pandas as pd
import numpy as np
import selenium
from selenium import webdriver
import shutil
import requests
from selenium.webdriver.chrome.options import Options
import os
import time
from selenium.common.exceptions import NoSuchElementException
```

Next, I tried to get the labels that so I create an empty list there.

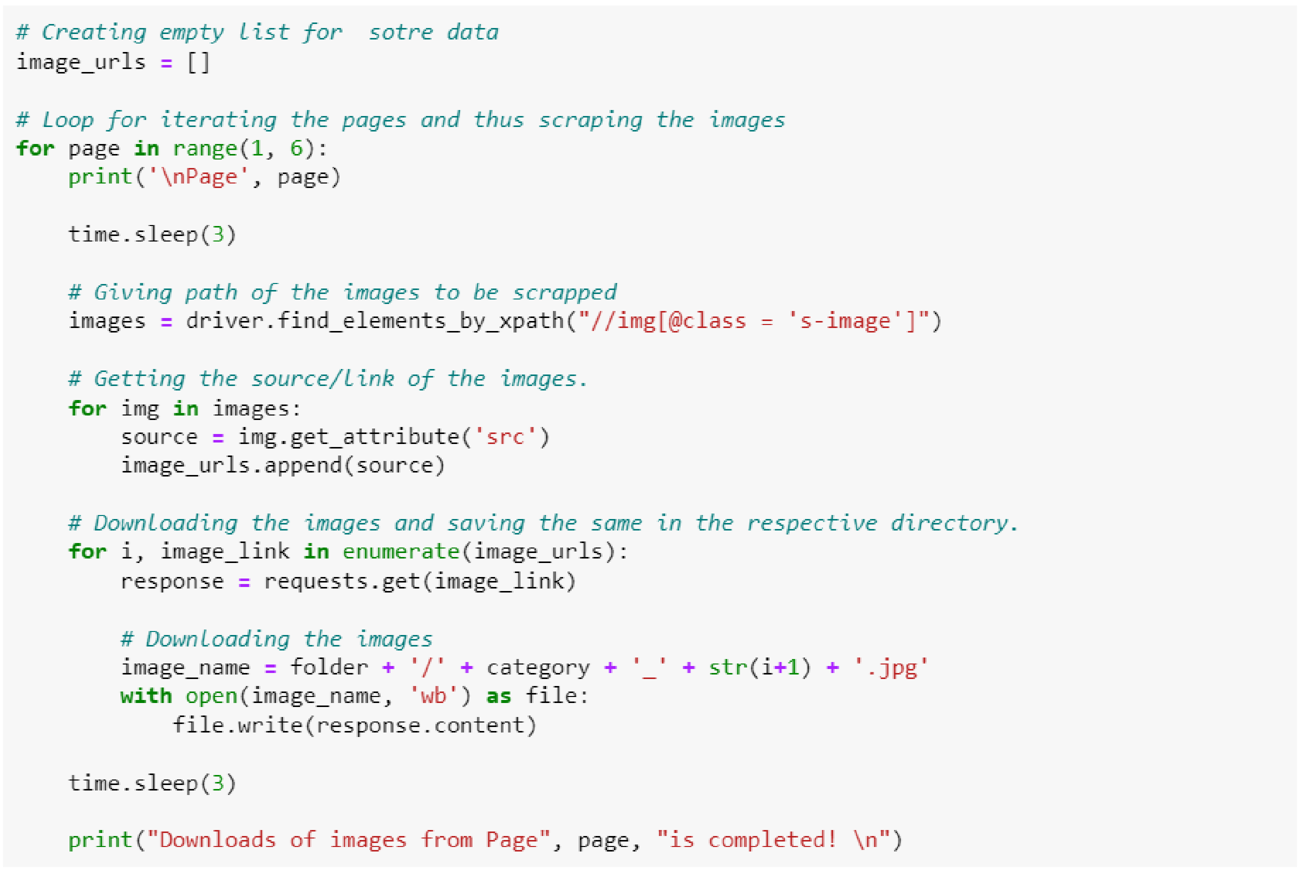


```
# Creating empty list to store labels
labels = []
# function to create a directory
def directory(file):
    img_dir = os.path.join(os.getcwd(), file)
    if not os.path.exists(img_dir):
        os.makedirs(img_dir)

    return file
# Calling the above function
folder = directory("Data")
```

Then created a function for Downloading the image and store in the folder.



```python
# Function for scraping, downloading and storing the required images/data.
def images(url, folder, labels, label,product):
    # Opening drive
    chrome_options = Options()
    chrome_options.add_argument("--incognito")
    driver = webdriver.Chrome(r'chromedriver.exe', options=chrome_options)
    web_url = driver.get(url)

    time.sleep(3)

    # Locating the search bar
    search_bar = driver.find_element_by_id("twotabsearchtextbox")
    print("Search Bar located")
    search_bar.clear()
    print("search bar clear")
    category = product
    search_bar.send_keys(category)

    print("locating the button and clicking it to search the category")
    button = driver.find_element_by_id('nav-search-submit-button')
    button.click()

    time.sleep(3)

    # Creating empty list for  sotre data
    image_urls = []

    # Loop for iterating the pages and thus scraping the images
    for page in range(1, 6):
        print('\nPage', page)

        time.sleep(3)
```



```python
# Creating empty list for  sotre data
image_urls = []

# Loop for iterating the pages and thus scraping the images
for page in range(1, 6):
    print('\nPage', page)

    time.sleep(3)

    # Giving path of the images to be scrapped
    images = driver.find_elements_by_xpath("//img[@class = 's-image']")

    # Getting the source/link of the images.
    for img in images:
        source = img.get_attribute('src')
        image_urls.append(source)

    # Downloading the images and saving the same in the respective directory.
    for i, image_link in enumerate(image_urls):
        response = requests.get(image_link)

        # Downloading the images
        image_name = folder + '/' + category + '_' + str(i+1) + '.jpg'
        with open(image_name, 'wb') as file:
            file.write(response.content)

    time.sleep(3)

    print("Downloads of images from Page", page, "is completed! \n")
```

```
        time.sleep(3)
        if len(image_urls)<=200:
        # Moving to the next page.
            next_page = driver.find_element_by_xpath("//li[@class = 'a-last']/a")
            next_page.click()
        else:
            break


    print("Total images downlaoded of", str(category), ": ", len(image_urls))
    for i in image_urls:
        labels.append(label)
```

Then created a list with those items, created loop then put it on the function with below mention code.

```
items=['jeans','Sarees','Trousers']
```

```
for item in items:
    if item=='jeans':
        label= 0
    elif item=='Sarees':
         label= 1
    else :
         label= 2
    images(url, folder, labels, label,item)
```

Thereafter converted to csv file.

```
 #Converting the 'labels' list into dataframe.

labels = pd.DataFrame(labels)
labels
#converting to csv
labels.to_csv("Data.csv", index = False)
```

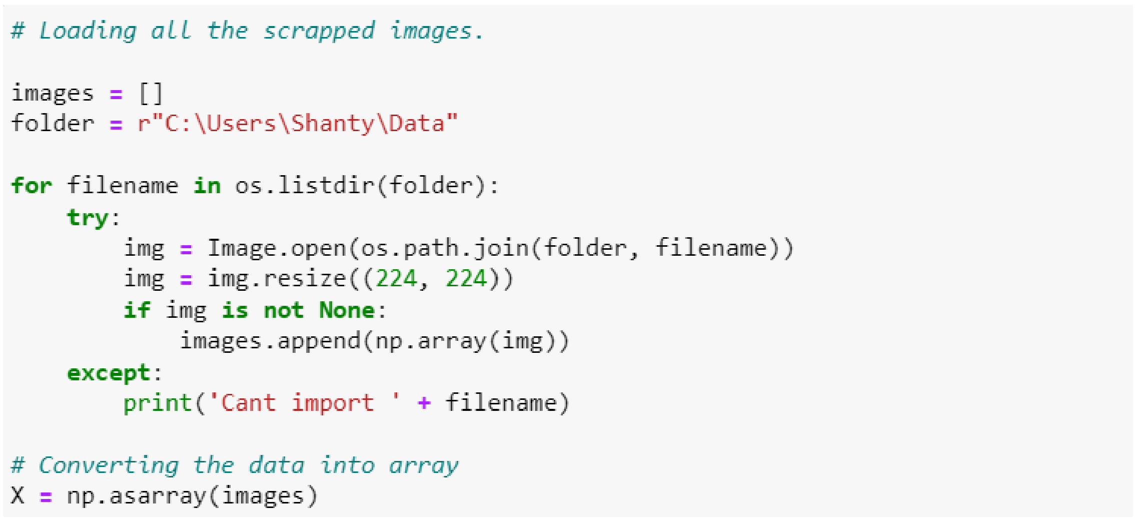
Now, Data is ready for Modelling.

Training Model for Classification

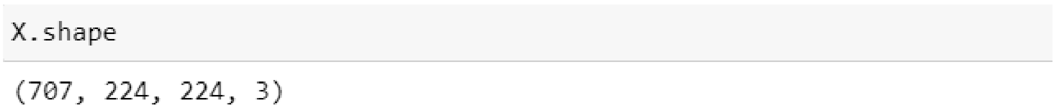
Importing a necessary library for model building.

```
# Importing Useful Libraries
import warnings
warnings.filterwarnings("ignore")
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.cm as cm
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D, Dropout, GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers
import os
from PIL import Image
from sklearn.model_selection import train_test_split
```

Loading the Image Dataset.

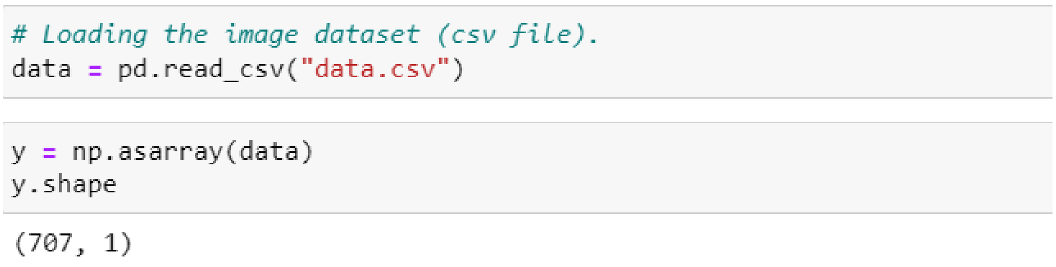


```
# Loading all the scrapped images.

images = []
folder = r"C:\Users\Shanty\Data"

for filename in os.listdir(folder):
    try:
        img = Image.open(os.path.join(folder, filename))
        img = img.resize((224, 224))
        if img is not None:
            images.append(np.array(img))
    except:
        print('Cant import ' + filename)

# Converting the data into array
X = np.asarray(images)
```



```
X.shape
```

```
(707, 224, 224, 3)
```



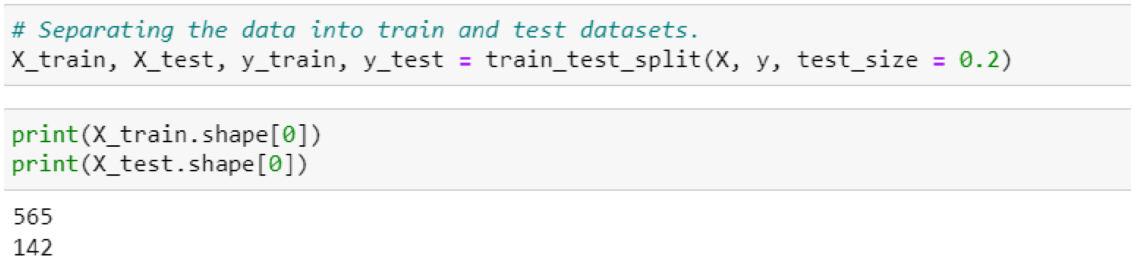
```
# Loading the image dataset (csv file).
data = pd.read_csv("data.csv")
```

```
y = np.asarray(data)
y.shape
```

```
(707, 1)
```

TRAIN TEST SPLIT

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model.

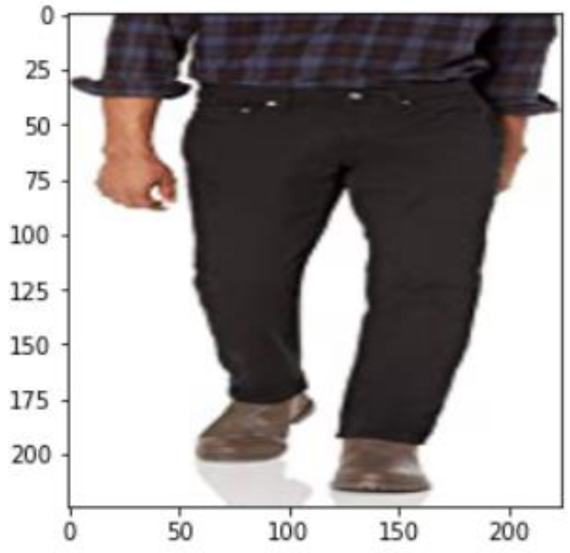


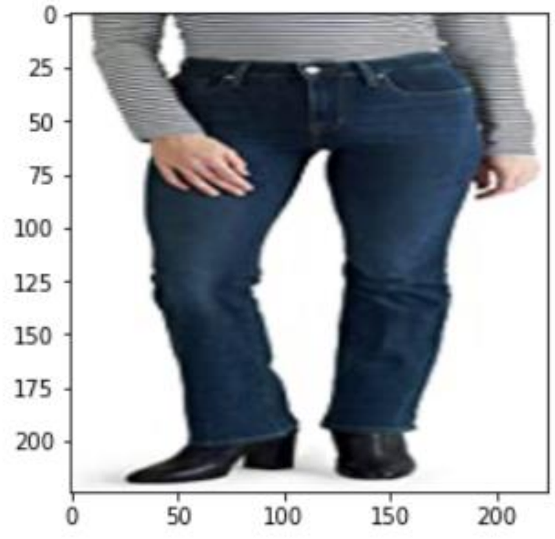
```
# Separating the data into train and test datasets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
```

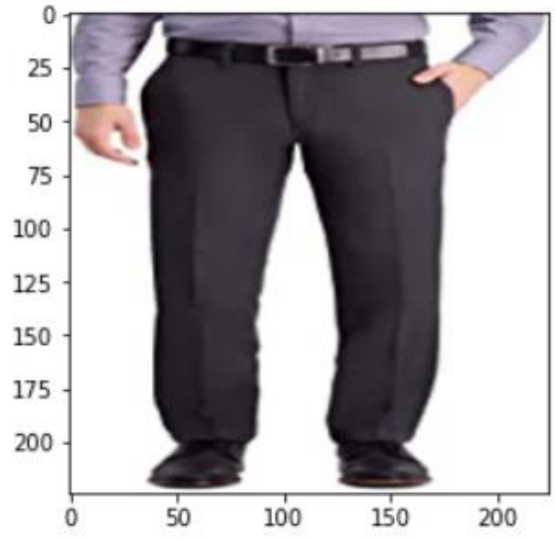
```
print(X_train.shape[0])
print(X_test.shape[0])
```

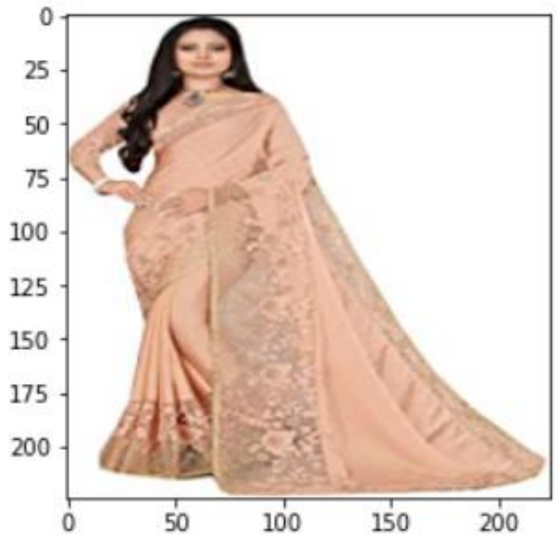
```
565
142
```

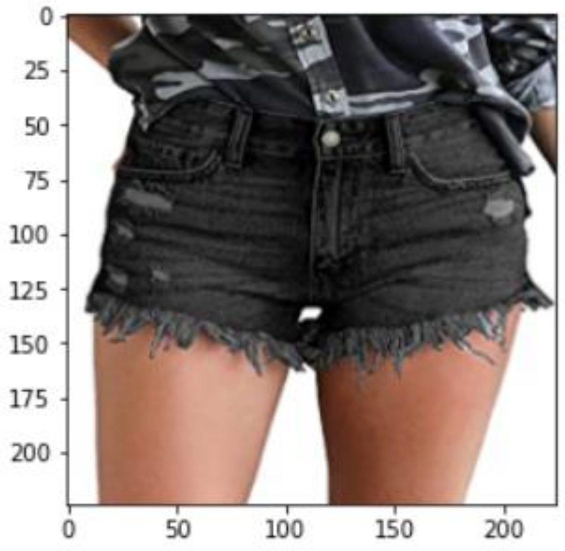
Data Visualization



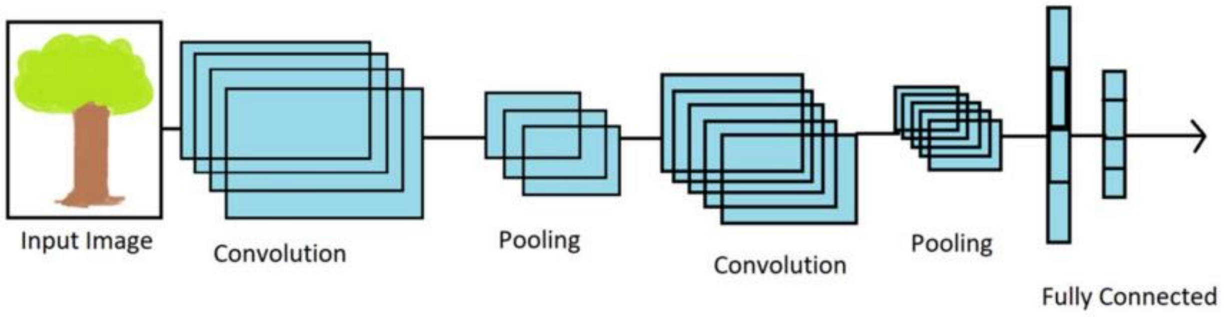








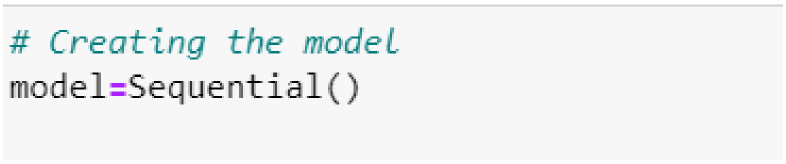
<u>Data Modelling</u>



Convolutional layers are the major building blocks used in convolutional neural networks.

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input result in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.



```
# Creating the model
model=Sequential()
```

1. Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size MxM. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter (MxM).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

2. Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature

map. Depending upon method used, there are several types of Pooling operations.
In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

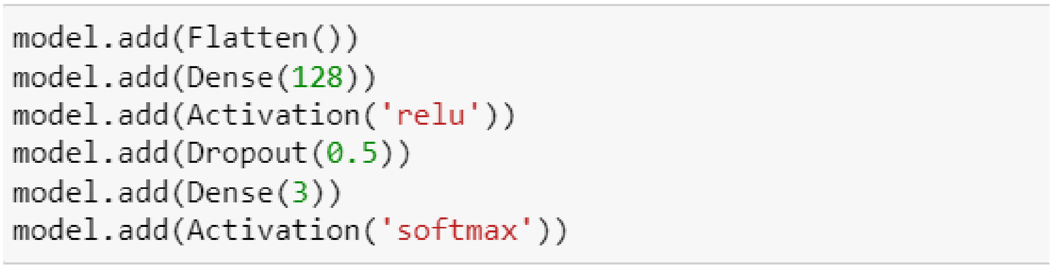
```
# First convolution layer
model.add(Conv2D(32,(3,3),input_shape=X.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

```
# Second convolution layer
model.add(Conv2D(32,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

```
# Third convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

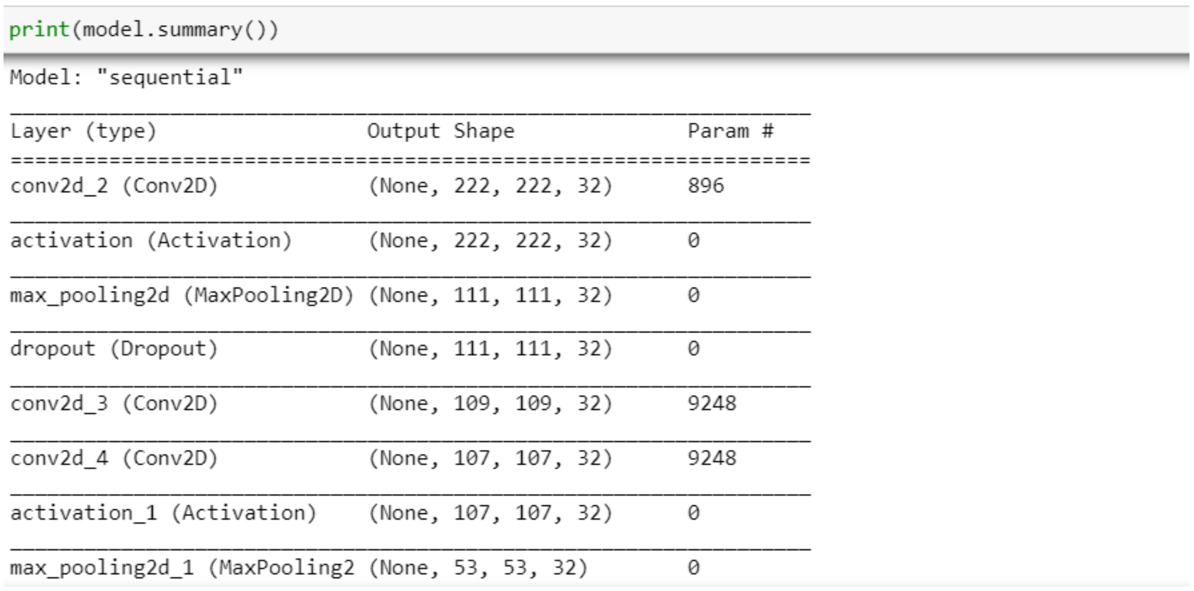
```
# Fourth convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```

```
# Fifth convolution layer
model.add(Conv2D(64,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
```



```
model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(3))
model.add(Activation('softmax'))
```

Getting The summary of our Model: "sequential"

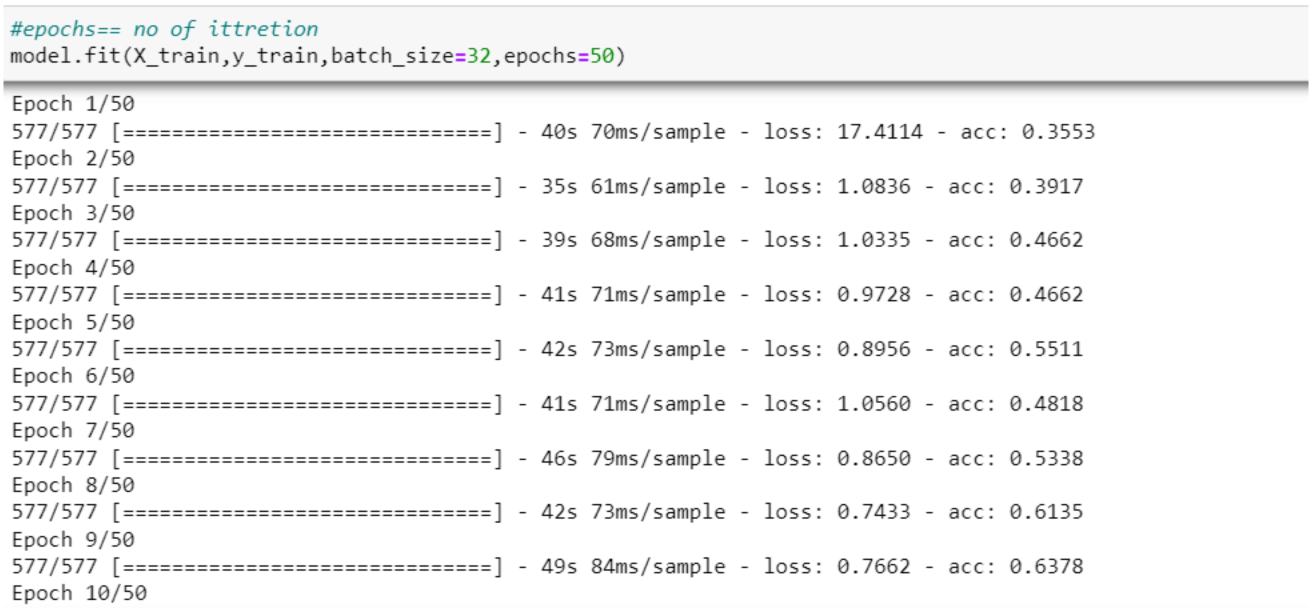


```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_2 (Conv2D)            (None, 222, 222, 32)      896       
_________________________________________________________________
activation (Activation)      (None, 222, 222, 32)      0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 111, 111, 32)      0         
_________________________________________________________________
dropout (Dropout)            (None, 111, 111, 32)      0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 109, 109, 32)      9248      
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 107, 107, 32)      9248      
_________________________________________________________________
activation_1 (Activation)    (None, 107, 107, 32)      0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 53, 53, 32)        0         
```

Next initialize the optimizer.

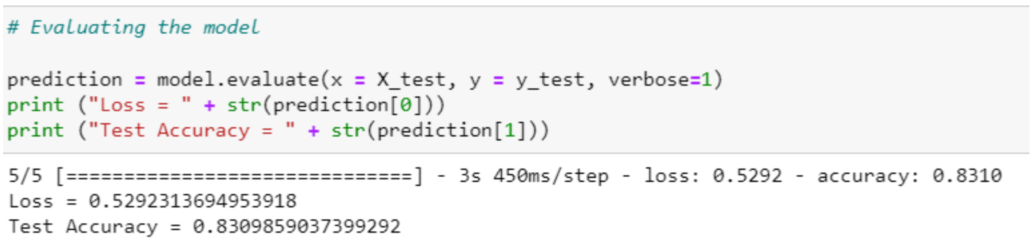
```
from tensorflow.keras.optimizers import Adam
optimizer=Adam(lr=0.001)


# Compiling the model

model.compile(optimizer=optimizer,loss="sparse_categorical_crossentr
opy",metrics=['accuracy'])
Fit the Model ..
model.fit(X_train,y_train,batch_size=32,epochs=50)
 and getting the Below Result
```



```
#epochs== no of ittretion
model.fit(X_train,y_train,batch_size=32,epochs=50)
```

```
Epoch 1/50
577/577 [==============================] - 40s 70ms/sample - loss: 17.4114 - acc: 0.3553
Epoch 2/50
577/577 [==============================] - 35s 61ms/sample - loss: 1.0836 - acc: 0.3917
Epoch 3/50
577/577 [==============================] - 39s 68ms/sample - loss: 1.0335 - acc: 0.4662
Epoch 4/50
577/577 [==============================] - 41s 71ms/sample - loss: 0.9728 - acc: 0.4662
Epoch 5/50
577/577 [==============================] - 42s 73ms/sample - loss: 0.8956 - acc: 0.5511
Epoch 6/50
577/577 [==============================] - 41s 71ms/sample - loss: 1.0560 - acc: 0.4818
Epoch 7/50
577/577 [==============================] - 46s 79ms/sample - loss: 0.8650 - acc: 0.5338
Epoch 8/50
577/577 [==============================] - 42s 73ms/sample - loss: 0.7433 - acc: 0.6135
Epoch 9/50
577/577 [==============================] - 49s 84ms/sample - loss: 0.7662 - acc: 0.6378
Epoch 10/50
```

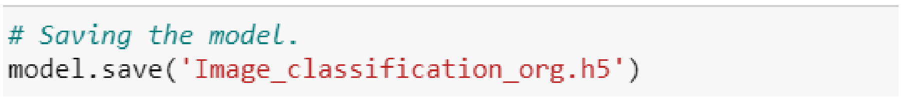
EVALUATING THE MODEL



```
# Evaluating the model

prediction = model.evaluate(x = X_test, y = y_test, verbose=1)
print ("Loss = " + str(prediction[0]))
print ("Test Accuracy = " + str(prediction[1]))
```

```
5/5 [==============================] - 3s 450ms/step - loss: 0.5292 - accuracy: 0.8310
Loss = 0.5292313694953918
Test Accuracy = 0.8309859037399292
```

SAVING THE MODEL



```
# Saving the model.
model.save('Image_classification_org.h5')
```

Hardware and Software Requirements and Tools Used

Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

1. Lenovo flex corei5 laptop

2. Jupyter Notebook :The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis

and is widely used by data scientists.

3. MS PowerPoint: For preparing presentation of project.

4. MS word: For preparing report

5. Pandas: is a software library written for the Python programming language for data manipulation and analysis.

6. Numpy: is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

7. Matplotlib: is a plotting library for the Python programming language and its numerical mathematics extension

8. NumPy.Seaborn: is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

CONCLUSION

The Test Accuracy 83%.
We have got a good accuracy because we have collected more than 200 data if we collect more than 1000 data, we would be able to train the dataset with better accuracy.