Image Scraping and Classification Project

Problem Statement:

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.

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)

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.

Model Building Phase:

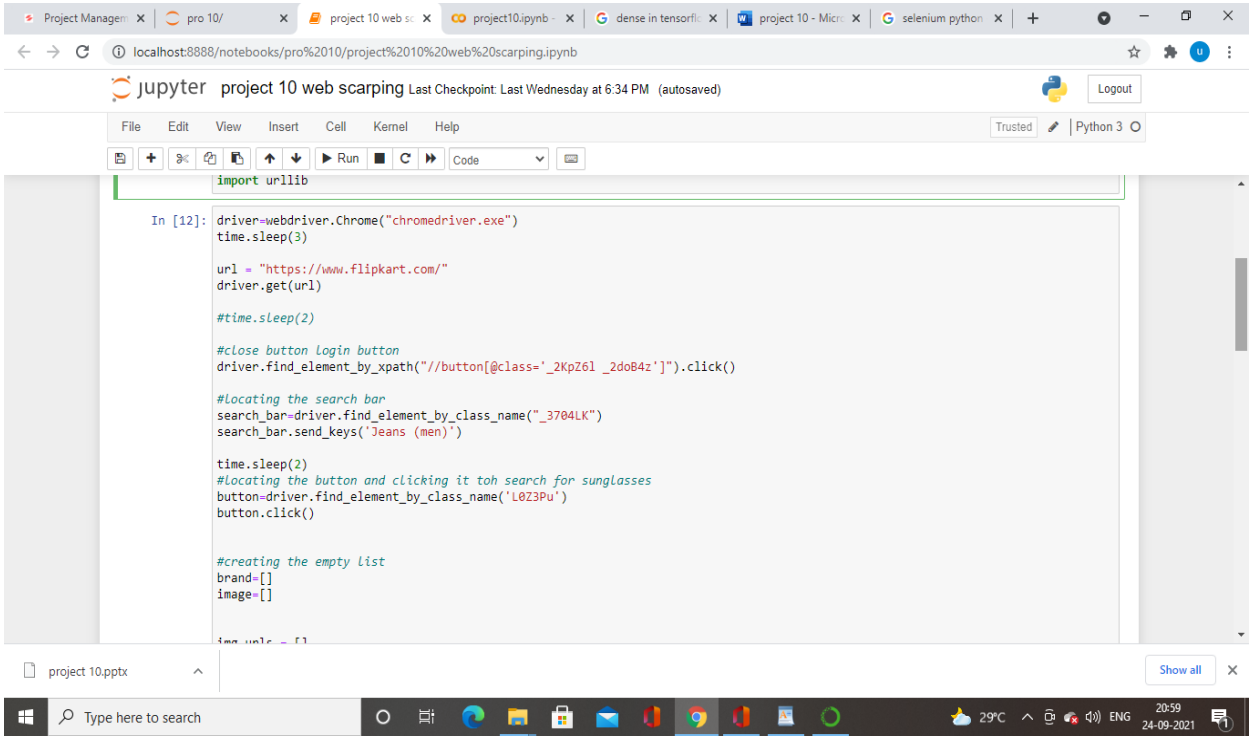
After the data collection and preparation is done, you need

to build an image classification model that will classify between these 3 categories mentioned above

Selenium

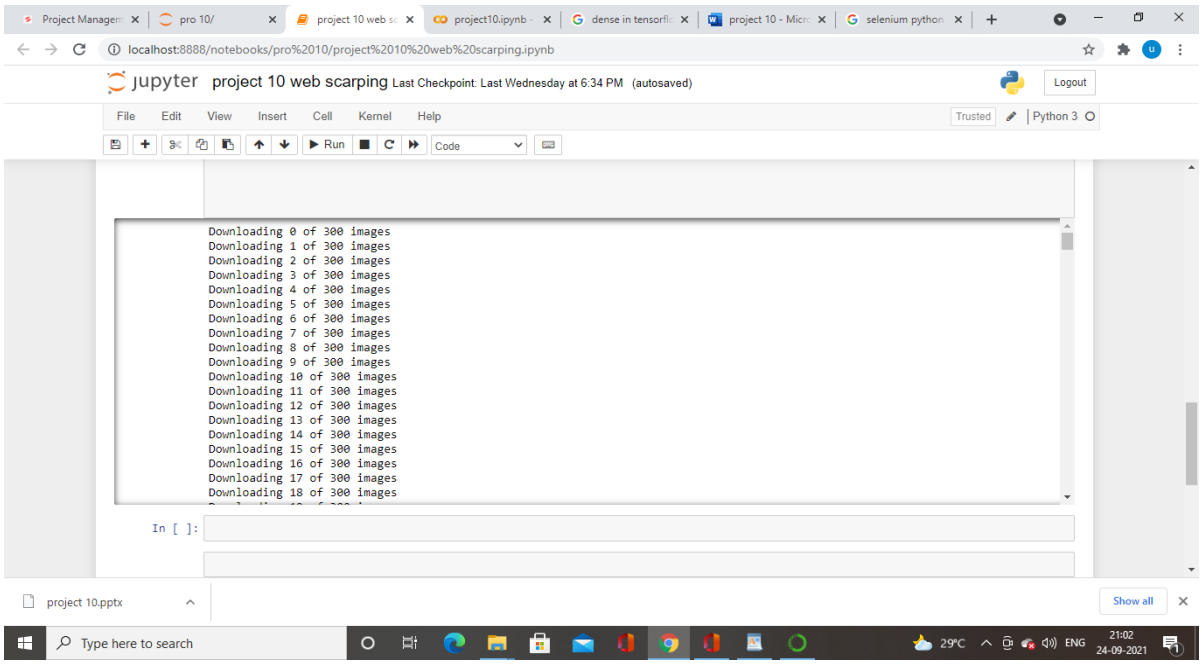
Selenium is a powerful tool for controlling web browsers through programs and performing browser automation. It is functional for all browsers, works on all major OS and its scripts are written in various languages

```
# Importing Libraries

import selenium

import pandas as pd

import time

from bs4 import BeautifulSoup


# Importing selenium webdriver

from selenium import webdriver


# Importing required Exceptions which needs to handled

from selenium.common.exceptions import
StaleElementReferenceException, NoSuchElementException
```



Web driver which is used for google chrome to open website

Downloding 300 images for deeplearning



Deep learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data.

TensorFlow

TensorFlow is an open source framework developed by Google researchers to run machine learning, deep learning and other statistical and predictive analytics workloads. Like similar platforms, it's designed to streamline the process of developing and executing advanced analytics applications for users such as data scientists, statisticians and predictive modelers

```
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
```

These are library which for deep learning for image processing

```
from google.colab import drive
drive.mount('/content/drive')
```

Which is used for getting image from the google drive

```
import cv2
width, height= 224, 224
img =
cv2.imread('/content/drive/MyDrive/project 10/train/Jeans (men)/folder10.jpg')

img_resized = cv2.resize(img,(width, height))
```

Checking for the image size

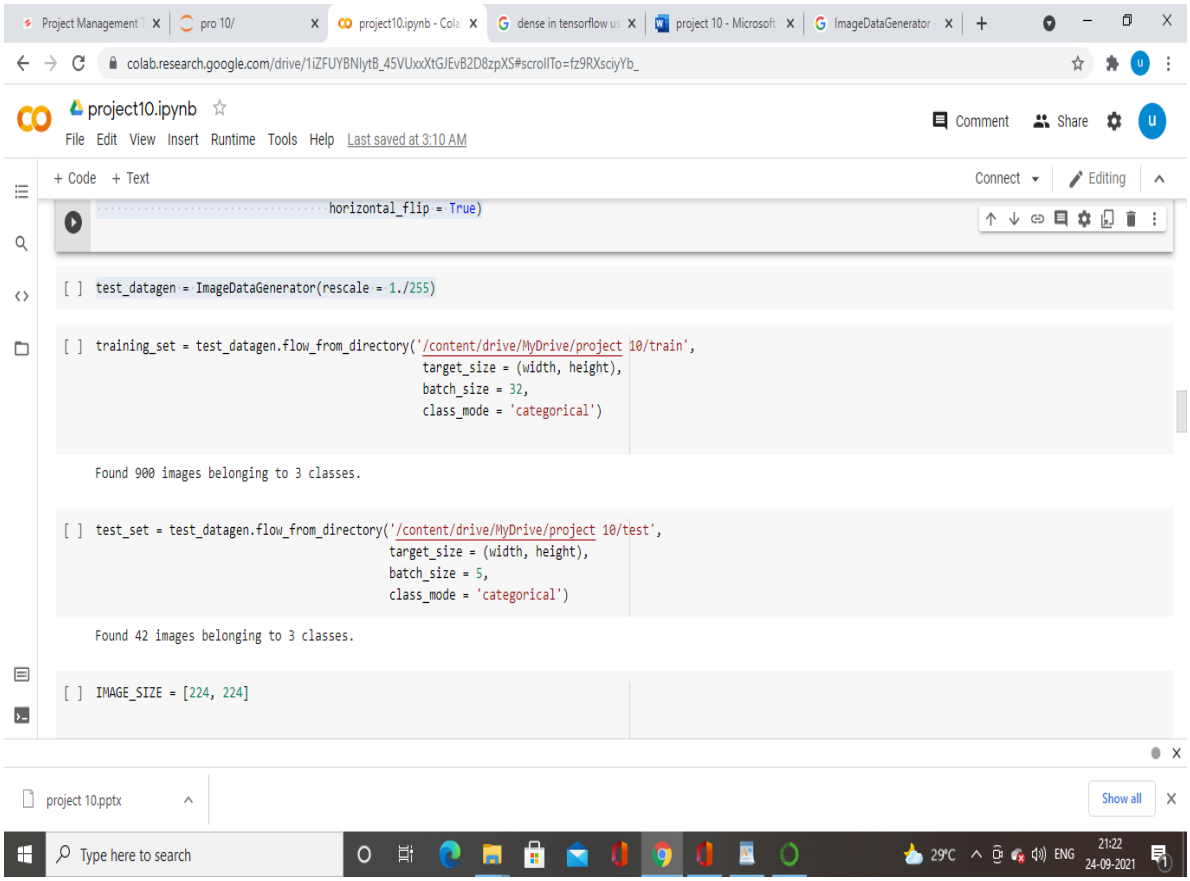
```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
```

Which is used for the converting bgr to rgb

ImageDataGenerator

Keras ImageDataGenerator is a gem! It lets you augment your images in real-time while your model is still training! You can apply any random transformations on each training image as it is passed to the model. This will not only make your model robust but will also save up on the overhead memory!

```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)
```



Getting data from train and test set

```
  # useful for getting number of classes
folders = glob('/content/drive/MyDrive/project 10/train/*')
  # our layers - you can add more if you want
x = Flatten()(vgg.output)
#x = Dense(1000, activation='relu')(x)
prediction = Dense(len(folders),activation='softmax')(x)
# create a model object
model = Model(inputs=vgg.input, outputs=prediction)
# view the structure of the model
model.summary()
```

compile

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

```
input_1 (InputLayer) [(None, 224, 224, 3)] 0
 block1_conv1 (Conv2D) (None, 224, 224, 64) 1792
block1_conv2 (Conv2D) (None, 224, 224, 64) 36928
block1_pool (MaxPooling2D) (None, 112, 112, 64) 0
block2_conv1 (Conv2D) (None, 112, 112, 128) 73856
block2_conv2 (Conv2D) (None, 112, 112, 128) 147584
block2_pool (MaxPooling2D) (None, 56, 56, 128) 0
block3_conv1 (Conv2D) (None, 56, 56, 256) 295168
block3_conv2 (Conv2D) (None, 56, 56, 256) 590080
```

block3_conv3 (Conv2D) (None, 56, 56, 256) 590080
block3_pool (MaxPooling2D) (None, 28, 28, 256) 0
block4_conv1 (Conv2D) (None, 28, 28, 512) 1180160
block4_conv2 (Conv2D) (None, 28, 28, 512) 2359808
block4_conv3 (Conv2D) (None, 28, 28, 512) 2359808
block4_pool (MaxPooling2D) (None, 14, 14, 512) 0
block5_conv1 (Conv2D) (None, 14, 14, 512) 2359808
block5_conv2 (Conv2D) (None, 14, 14, 512) 2359808
block5_conv3 (Conv2D) (None, 14, 14, 512) 2359808
block5_pool (MaxPooling2D) (None, 7, 7, 512) 0

flatten (Flatten) (None, 25088) 0

_____ _____ dense (Dense) (None, 3) 75267

========== Total params: 14,789,955

Trainable params: 75,267

Non-trainable params: 14,714,688

fit the model

```
r = model.fit_generator(
training_set,validation_data=test_set,epochs=2)
```

Epoch 1/2 29/29 [==============================] - 471s 16s/step -

loss: 0.4936 - accuracy: 0.8267 - val_loss: 0.0565 - val_accuracy: 0.9762

Epoch 2/2 29/29 [==============================] - 460s 16s/step -

loss: 0.0386 - accuracy: 0.9967 - val_loss: 0.0213 - val_accuracy: 1.0000