Swimming Pool Detection using YOLOv5

For some tasks, there are **pretrained models** that are already trained on huge datasets using a vast computing resource. Trying to a build a model from scratch for such tasks may be reinventing the wheel. We can simply take the pretrained model and train it on our specific data for some time and get very good accuracy withing a few hours, which would have taken us weeks or even months if we had done all the model building and training from scratch.

YOLOv5 (https://github.com/ultralytics/yolov5) is a really great open-source objection detection model. Through **transfer-learning**, this model can be easily trained to a custom dataset for detecting the feature of our interest.

This tutorial will guide us on using this model for swimming pool detection in satellite images.

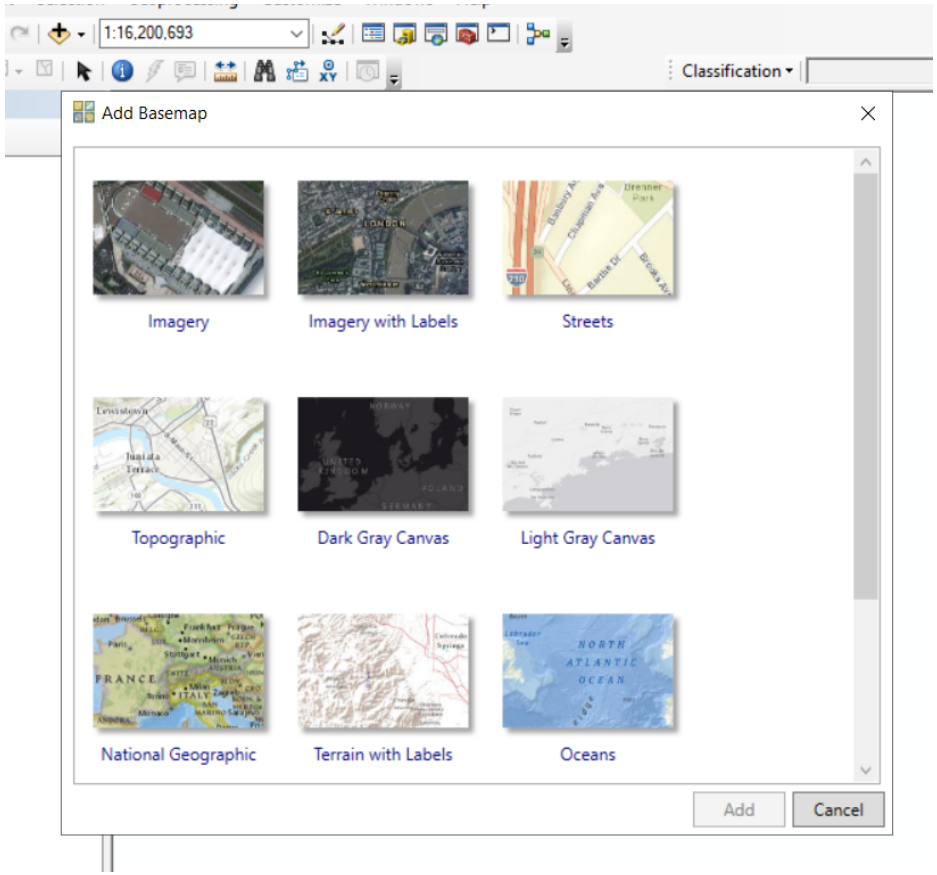
First, we will see how to prepare the training data then we will train, test and deploy the model.

1 - Data Preparation

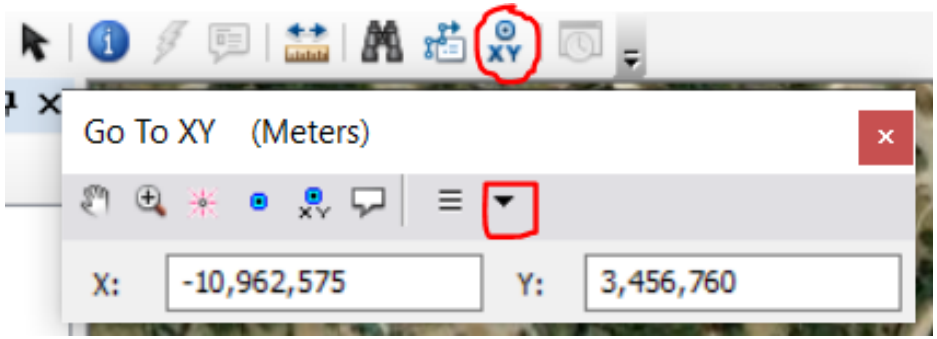
For our swimming pool detection, we will need some satellite images. As swimming pools are not easily identifiable in moderate-resolution images such as Landsat, we will use high-resolution images from base maps. We will use ArcMap, a desktop GIS software to save the satellite images. This will require some manual work.

1.1 Getting Satellite Images

1. Open ArcMap. From the drop-down of Add Data, select **Add Basemap** and choose the **Imagery**.

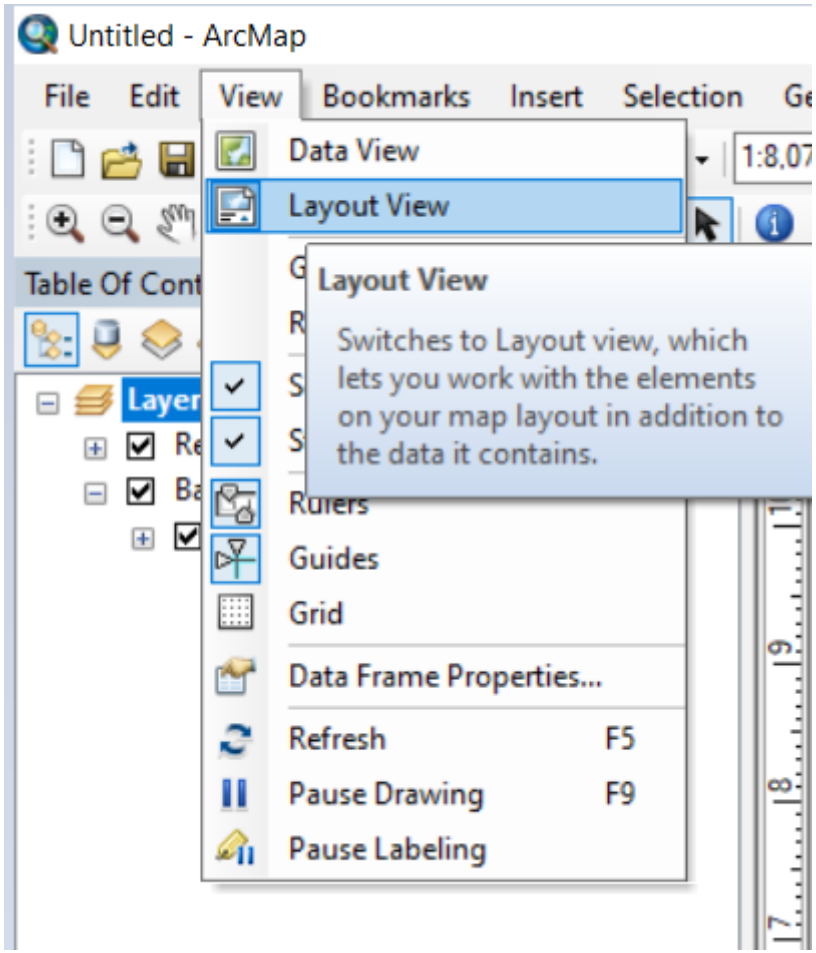


2. Use the **Go to XY** tool. From the dropdown, change the unit to **Meters** .

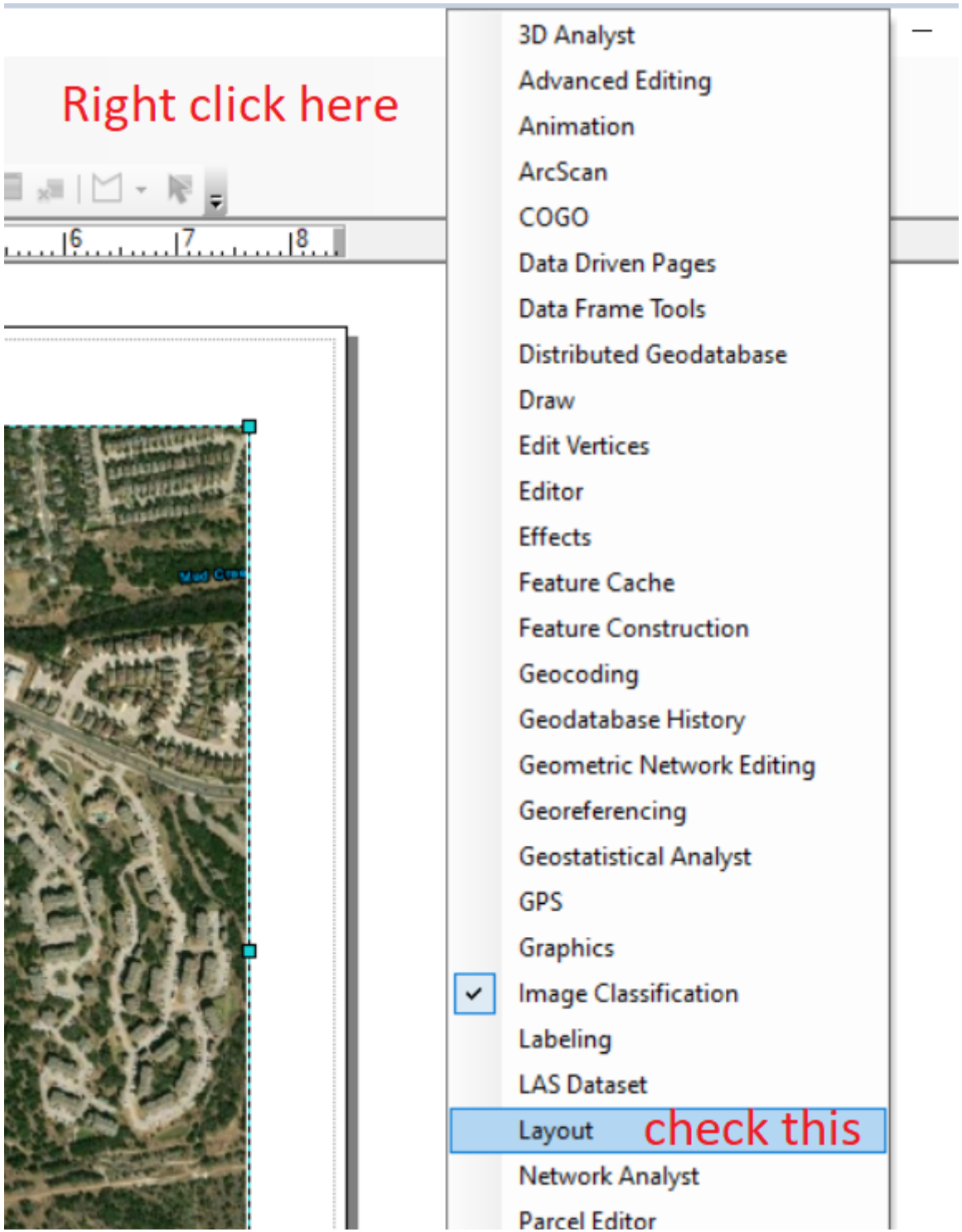


For **longitude**, enter the value **-10962575**, and for **latitude** enter value **3456760**. This will zoom to a portion of **San Antenio, Texas**.

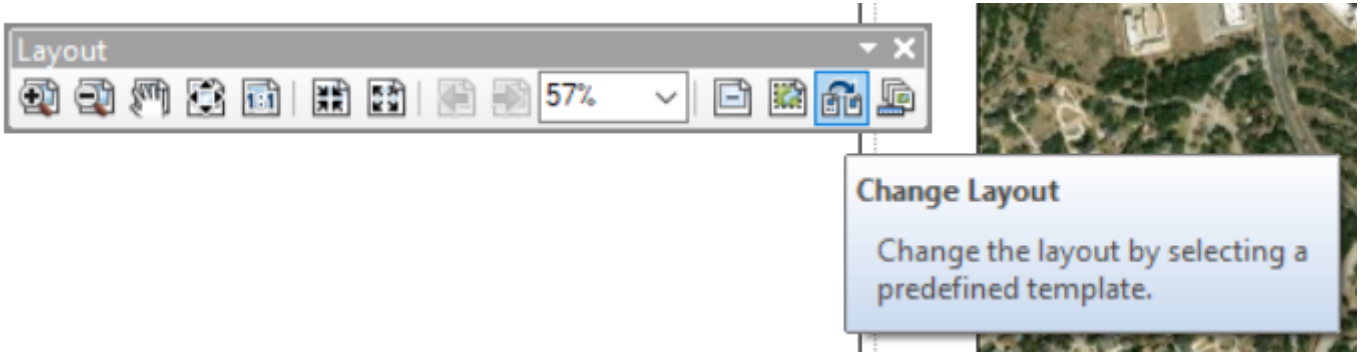
3. From the main menu, go to **View** and to the **Layout View**.



4. Now, activate the **Layout** tools by right cleaning on the gray area above the display port, then checking the **Layout** tool.


Right click here
3D Analyst
Advanced Editing
Animation
ArcScan
COGO
Data Driven Pages
Data Frame Tools
Distributed Geodatabase
Draw
Edit Vertices
Editor
Effects
Feature Cache
Feature Construction
Geocoding
Geodatabase History
Geometric Network Editing
Georeferencing
Geostatistical Analyst
GPS
Graphics
Image Classification
Labeling
LAS Dataset
Layout
check this
Network Analyst
Parcel Editor

5. From the **Layout** tool, click on **Change Layout**. Then choose **ISO A0 Landscape**, then click **finish** and wait for the changes to take effect.

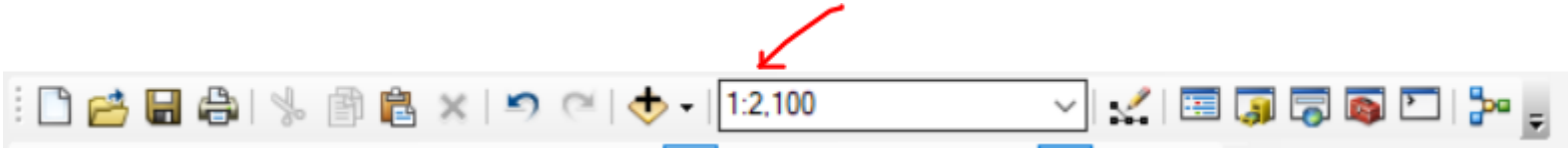


6. Now, click on the map area. It will show a bounding box (map frame), strech it's boundaries beyond the outline of the paper.

Before stretching the map frame

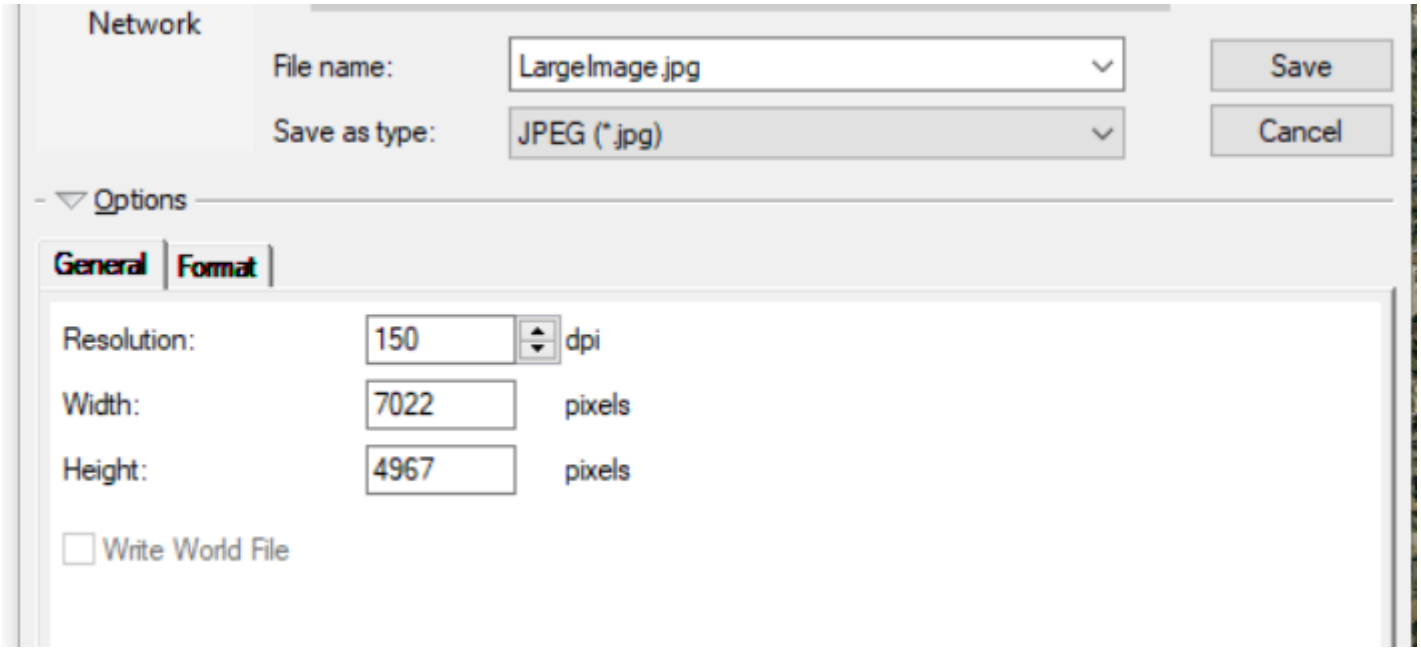
After stretching the map frame

You may want to change the scale at which the map is rendered. From the **Standard Toolbar** change the scale to 1:2100 or higher.



Now it's time to save the map. We will export the map to jpeg format. It definitely distorts the radiometric properties of the images, when saving a satellite image for future analysis that might depend on spectral signature it is not advisable. But for our purpose here, as we can still identify swimming pools from resampled images, we are good to go.

7. From the main menu, go to **File** and click **Export Map** button. Save as type **JPEG** with the **resolution** of **150 dpi** and save at an appropriate location.



1.2 Clipping Satellite Images

For object detection, we will need multiple smaller images. In each image we will annotate the bounding box of every swimming pool we will see for training. So, we need to clip the satellite image. For this we will use Python (https://www.python.org/downloads/). First make sure you have python 3.x installed on your computer.

Go to the folder where you saved the image from step 1.1, then on the address bar, type **cmd** then hit **enter** to open a **command Prompt** at that directory. Alternatively, we can open the command prompt first and change the directory to that location by running **cd '<full path to the folder>'**

We will also need to install `Pillow` package for reading and writing images and `numpy` for clipping images. When in that directory, run following commands to install these packages. Be sure to remove the dollar sign ('$') if running directly in the command line.

In [ ]:

```
$ pip install numpy
```

In [ ]:

```
$ pip install Pillow
```

After the packages are installed, we will need to load required packages for image clipping.

Setting up

In [ ]:

```
import numpy as np
from PIL import Image
import os
```

If we are running the code in command line, change the directory to the folder containing the **LargeImage.jpg** .

In [ ]:

```
os.chdir('/content/')
```

Read image

In [ ]:

```
image = Image.open('LargeImage.jpg')
```

Convert image to numpy array

In [ ]:

```
im = np.asarray(image)
```

Clipping Images

Output image will be of shape (M, N). This will split into x*x images of size (M,N) and a few corner images may also be produced of remainder size. If we want the output be of certain size then provide then set M and N to those values.

In [ ]:

```
# Get shape and size of each output tile
x = 10
M = im.shape[0]//x
N = im.shape[1]//x

tiles = [im[x:x+M,y:y+N,:] for x in range(0,im.shape[0],M) for y in range(0,im.shape[1],N)]
```

Save tiles as jpg

In [9]:

```
for idx, tile in enumerate(tiles):
  tileImage = Image.fromarray(tile)
  tileImage.save(f'tile_{idx}.jpg')
```

Download images from colab to local disk

This part only applies if we use colab to split the image into tiles which we will upload to roboflow for annotation. Otherwise, our tiles are already saved on our local disk and we are ready to move to image annotation part.

In [ ]:

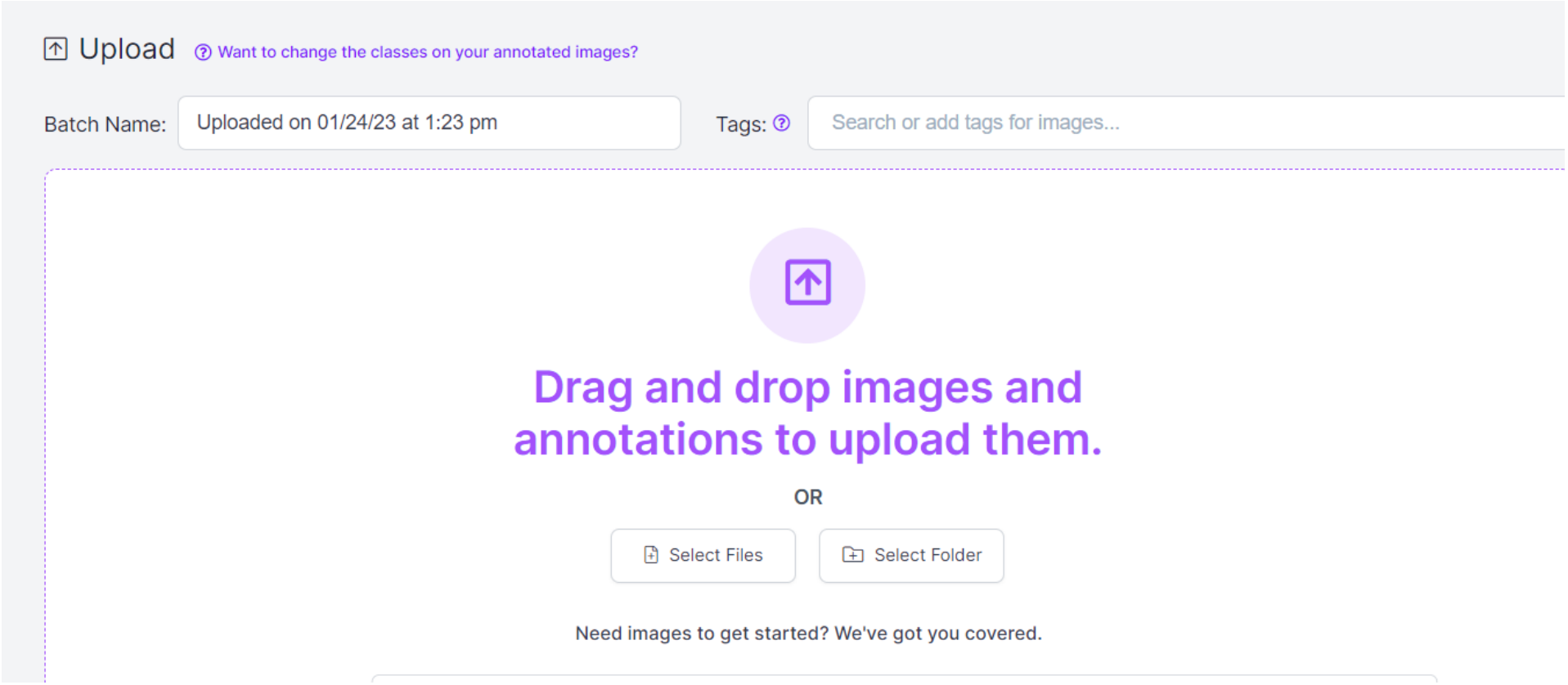
```
from google.colab import files
files.download('/content')
```

1.3 Annotating Images

We will use roboflow (https://roboflow.com/), which is a great tool for annotating images or videos, to annotate the images with bounding box. First we have to sign up at roboflow (https://roboflow.com/), which can be done with any email or github account. We can create a new workspace as a hobbies type and give any name.

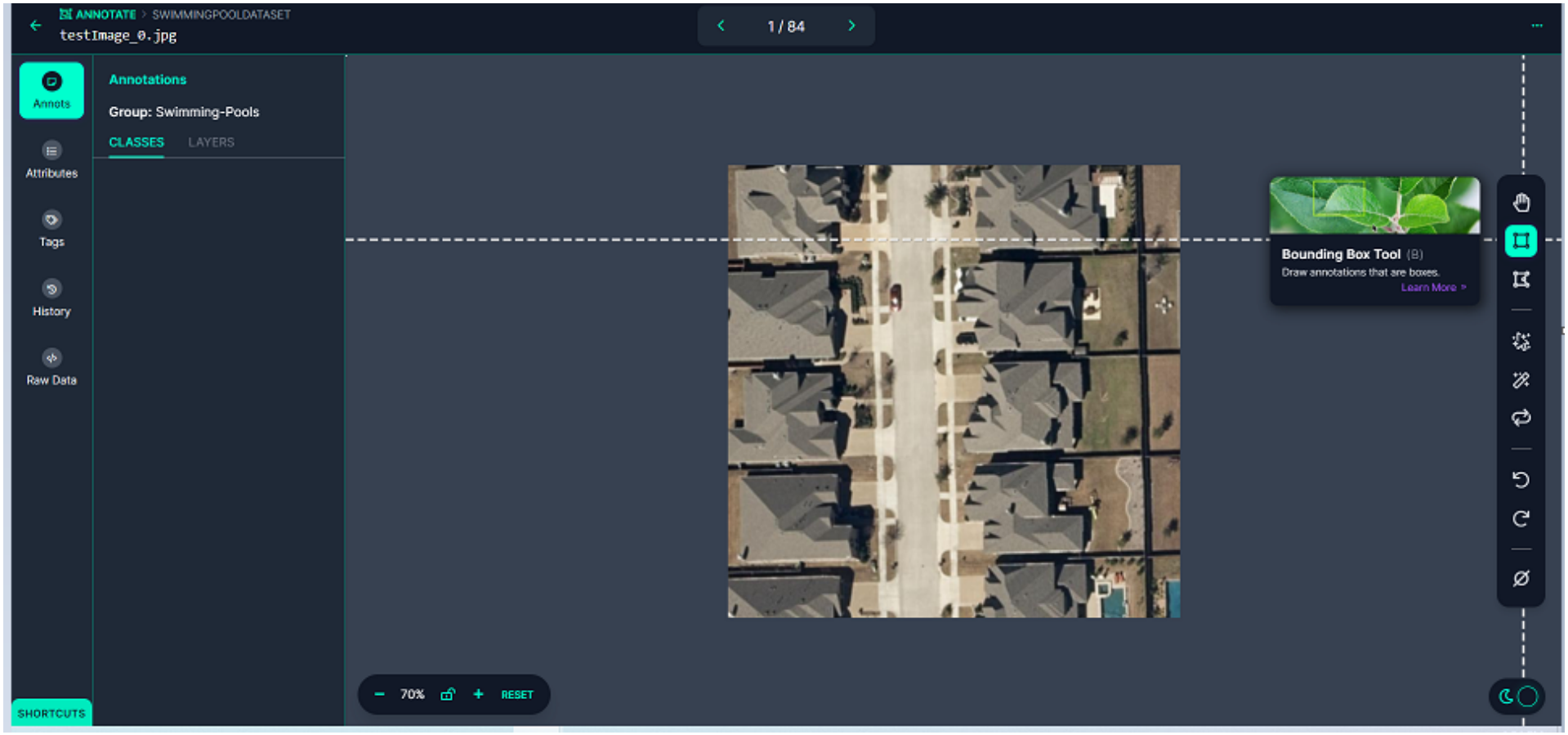
We can skip inviting other people and create a public workspace. Create on **new project** and **Explore Solo** and we are detecting **Swimming pools**. For license type, we can choose any of the options.

Now, we can either drag-and-drop images or add files or folder. After it finishes uploading images, click on **save and continue** on the top-right corner.



Next step is to assign different images to different person. But as there is only one person in this workspace, simply assign images to that account.

Now, for the actual annotation, click on the first image and start drawing a rectanglular bounding box around the image. After drawing one polygon hit enter to save it. We can draw multiple polygons if there are multiple swimming pools or not draw any if there is no swimming pool in the image.



After finishing the **Annotation**, go to back. It will show how many images are labelled or unlabelled and ask to **add the labelled images to dataset**. Click there. Keep 70% for training and 15% each for validation and testing. Click on **Add Images**.

Now click on **Generate New Version** . Keep everything default. Go to number 5 **Generate**.

SwimmingPoolDataset Dataset

Generate New Version

VERSIONS

To train a model, you must first generate a new version of your dataset.

Choose your dataset settings to get started.

Generating New Version

Prepare your images and data for training by compiling them into a version. Experiment with different configurations to achieve better training results.

Source Images

Images: 33
Classes: 1
Unannotated: 0

Train/Test Split

Training Set: 23 images
Validation Set: 4 images
Testing Set: 6 images

Preprocessing

Auto-Orient: Applied
Resize: Stretch to 640×640

Augmentation

Turned Off

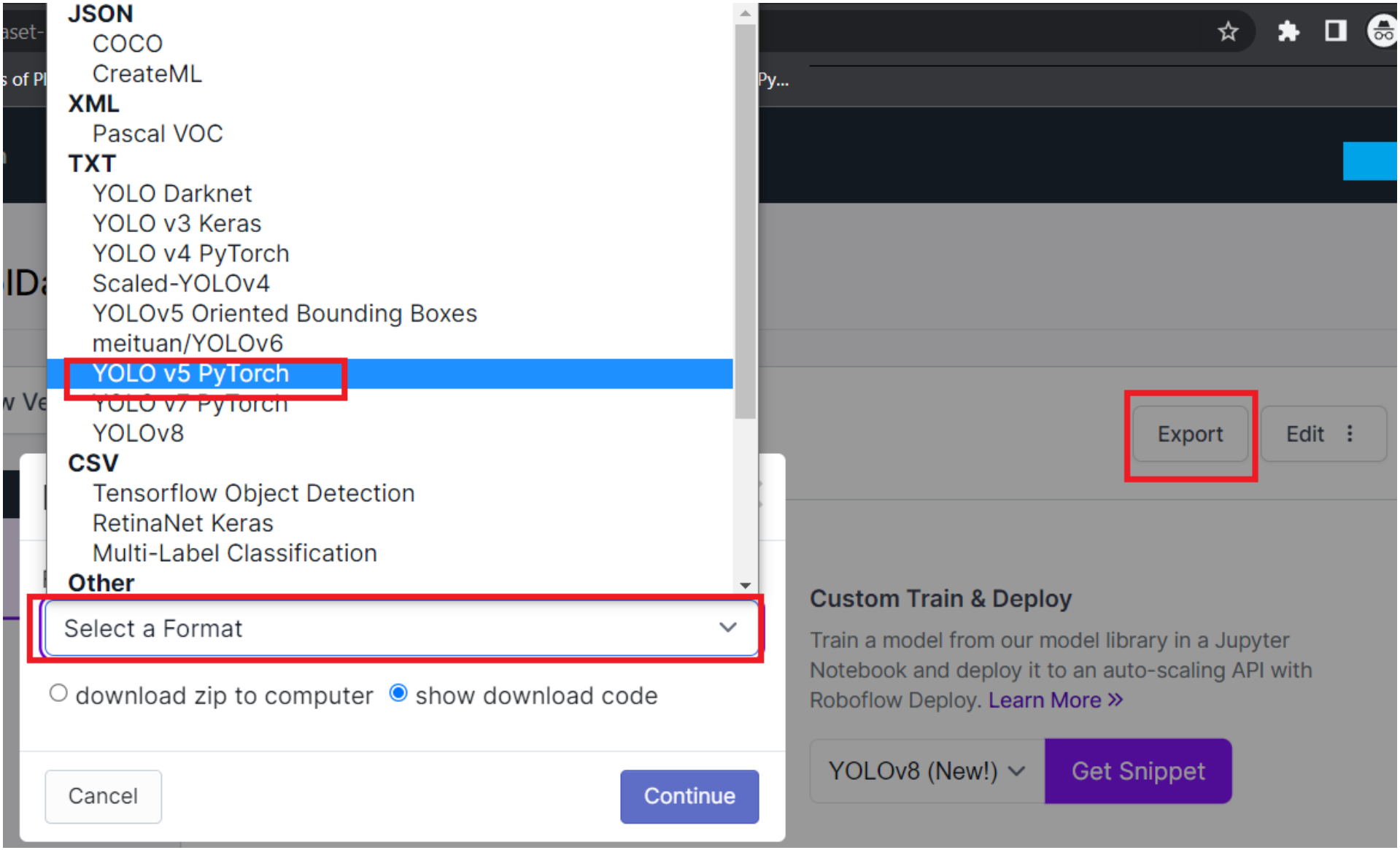
5 **Generate**

Review your selections then click "Generate" to create a moment-in-time snapshot of your dataset with the applied preprocessing steps.

Maximum Version Size: 33
See how this is calculated »

Generate

It will generate a version of data. Go to **Export**, and in the **format**, choose **YOLOv5 PyTorch** annd click on **Continue**. This will generate a **download code** that can be run in jupyter notebook or colab. Copy this code, we will need this to download our dataset, and also pay attention to the warning.



2 - Model Installation and setup

2.1 Installing yolov5

We can clone the **yolov5** github repository to obtain the pretrained model. We will need to install some dependencies for running this model. Also, we will install **roboflow**, which we will need to obtain the data using the **download code** copied at the end of the previous step.

In [ ]:

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output  # to display images

print(f"Setup complete. Using torch {torch.__version__} ({torch.cuda.get_device_properties(0).name if torch.cuda.is_available
```

2.2 Load Data from roboflow

In the cell below, paste the **download code** copied from roboflow to obtain the data we prepared in step 1.3

In [ ]:

```
from roboflow import Roboflow
rf = Roboflow(api_key="<personal_key>") # Replace
project = rf.workspace("<workspace_name>").project("<project_name>") # Replace
dataset = project.version(2).download("yolov8")
```

In [ ]:

```
# set up environment
os.environ["DATASET_DIRECTORY"] = "/content/datasets"
```

3 - Train the model

(**Already have the best.pt ? click here to go to the detection section**.) Run the following cell to train the model.

In [ ]:

```
!python train.py --img 640 --batch 16 --epochs 50 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache
```

3 - Detect swimming pools on test image

In the following cell make sure the location of **best.pt** is correct and the path of folder containing test images is also correct. This exact same code will also be used for detecting in the future, but we will need to change the location of **best.pt** and folder containing images.

The parameter `conf` is set to 0.2. Setting it low will detect more swimming pools but will also detect other things as swimming pool meaning it will increase the **comission error**, and setting it high will detect fewer swimiming pool meaning it will increase the **omission error**. We can play around with this value to see what works best.

In [ ]:

```
!python detect.py --weights /content/yolov5/runs/train/exp/weights/best.pt --img 640 --conf 0.2 --source /content/datasets/Sw
```

Now, let's visually inspect how well the model did.

In [ ]:

```
from matplotlib import pyplot as plt
from numpy import array
from PIL import Image

for imageName in glob.glob('/content/yolov5/runs/detect/exp2/*.jpg'): #assuming JPG
    image = Image.open(imageName)
    plt.imshow(array(image))
    plt.show()
    print("\n")
```

It seems the model is doing preety good job. As the storage of this colab session is temporary, if we want to use this model in the future without training it again, we will need to save the model weights. Run the following code to download the best weights.

In [ ]:

```
#export your model's weights for future use
from google.colab import files
files.download('./runs/train/exp/weights/best.pt')
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

5 - Deploying the model

Now we have the weights saved, we have no need to train the model if we want to simply use the model for detecting swimming pools. For this we will need to prepare the image on which we want to detect the swimming pool by following the method described in section 1.

If we want to use colab for detection, we will have to upload the images and weights to the collab session storage. Then, for the following cells, the location of best.pt,location of image on which we want to run the detection need to be changed. By running the cell below with appropriate parameters we can detect the images

In [12]:

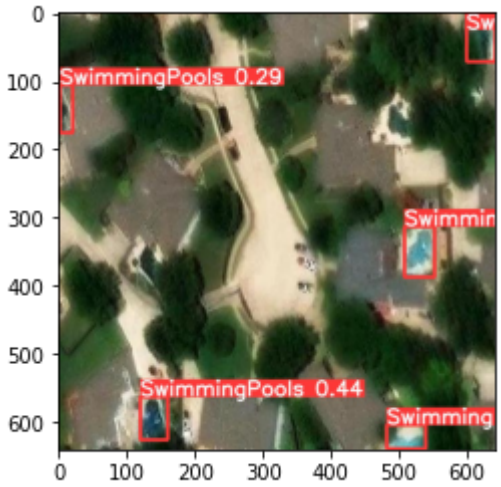
```
!python detect.py --weights /content/best.pt --img 640 --conf 0.2 --source /content/sample_data/exp
```

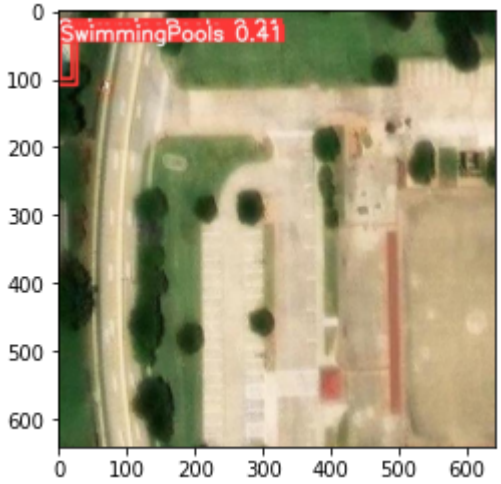
Now run the following cell after changing the location of output images, to visually inspect them.

In [17]:

```
from matplotlib import pyplot as plt
from numpy import array
from PIL import Image

for imageName in glob.glob('/content/yolov5/runs/detect/exp2/*.jpg'): #assuming JPG
    image = Image.open(imageName)
    plt.imshow(array(image))
    plt.show()
    print("\n")
```





Cong

- h tion,
- h
- h

Good luck on your learning journey.