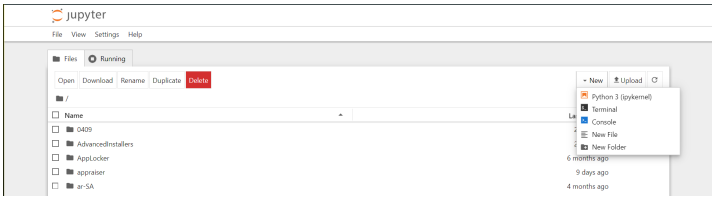
INFOSYS PROJECT - OBJECT TRACKING IN SURVEILLANCE VIDEOS

INSTALL OPENCV IN JUPYTER NOTEBOOK BY USING THE COMMAND

COMMAND -

```
pip -q install opencv-python
```

YOU CAN USE THE TERMINAL OPTION ON JUPYTER NOTEBOOK.

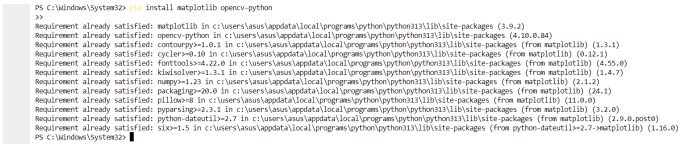


THEN INSTALL OPENCV IN THAT TERMINAL USING THE COMMAND

THEN BEFORE PROCEEDING TOWARDS NEXT STEP INSTALL MATPLOTLIB USING -

pip install matplotlib opencv-python

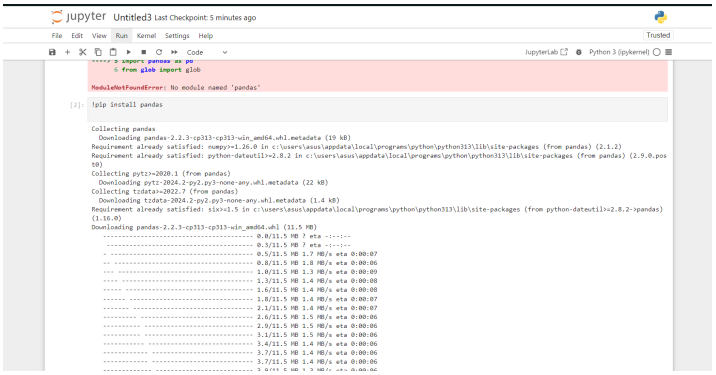
IN MY NOTEBOOK IT IS ALREADY INSTALLED .



NOW INSTALL PANDA ON THE NOTEBOOK -

!pip install pandas

HERE IS A CHANGE YOU DO NOT REQUIRE TO USE TERMINAL OF THE NOTEBOOK YOU CAN SIMPLY USE THE GIVEN COMMAND ON THE NOTEBOOK ITSELF



AS YOU MAY HAVE NOTICE THE RED BOC ABOVE , THAT IS BECAUSE THE PANDA MODULE WAS NOT INSTALLED ON MY NOTEBOOK BUT AFTER INSTALLANTION YOU HAVE TO RUN THE SAME CODE AGAIN AND HERE IS THE NEW RESULT -

THEN IN THE MAIN PROCESS USE THE GIVEN CODE BELOW -

```
import cv2
import numpy as np
# from google.colab.patches import cv2_imshow
import matplotlib.pylab as plt
import pandas as pd
from glob import glob
```

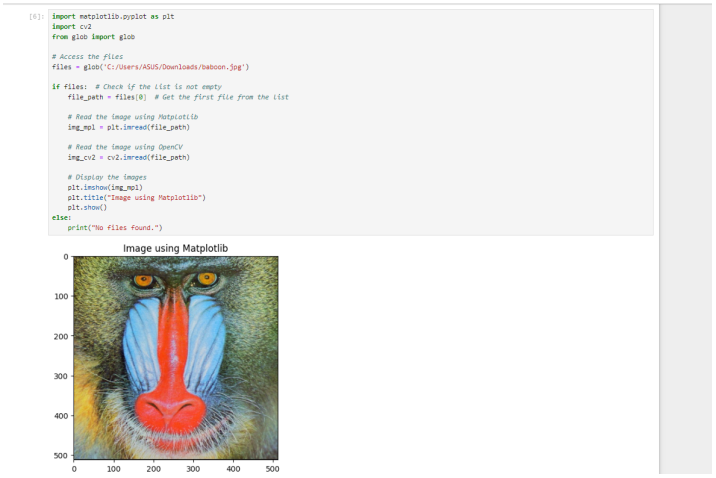
THEN IN THE NEXT STEP YOU WILL USE THE GIVEN CODE -

```
import matplotlib.pyplot as plt
import cv2
from glob import glob

# Access the files
files = glob('C:\Users\ASUS\Downloads\baboon.jpg')

if files:  # Check if the list is not empty
    file_path = files[0]  # Get the first file from the list

    # Read the image using Matplotlib
    img_mpl = plt.imread(file_path)

    # Read the image using OpenCV
    img_cv2 = cv2.imread(file_path)

    # Display the images
    plt.imshow(img_mpl)
    plt.title("Image using Matplotlib")
    plt.show()
else:
    print("No files found.")BUT IN THIS GIVEN CODE ON THE LINE - files = glob ( "C:\Users\ASUS\Downloads\baboon.jpg
```

BUT BEFORE USING THE COMMAND YOU CAN SEE THE LINE - files = glob ('C:\Users\ASUS\Downloads\baboon.jpg')
HERE YOU HAVE TO CHANGE THE LOCATION OF THE PATH WHERE YOU HAVE SAVED THE PHOTO WHICH YOU WANTED TO USE IN THE PROJECT

FOR EXAMPLE I HAVE USED THIS PICTURE -

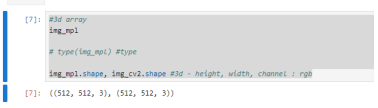
AFTER USING THE GIVEN CODE THE PHOTO WHILE CHANGE INTO THE GIVEN PHOTO -



AFTER GETTING THIS OUTPUT RUN ANOTHER SHELL AND USE THIS CODE -

```
#3d array
img_mpl

# type(img_mpl) #type

img_mpl.shape, img_cv2.shape #3d - height, width, channel : rgb
```

AND THIS IS THE OUTPUT -

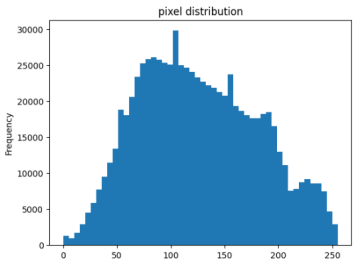


NOW AFTER THAT RUN ANOTHER CODE -

```
pd.Series(img_mpl.flatten()).plot(kind='hist', bins = 50, title='pixel distribution') #bins: specifies the number of bins (g
plt.show()
```

AND HERE IS THE PIXEL DISTRIBUTION OUTPUT -

```
[8]: pd.Series(img_mpl.flatten()).plot(kind='hist', bins = 50, title='pixel distribution') #bins: specifies the number of bins (groups) in the histogram, g(v
plt.show()
```

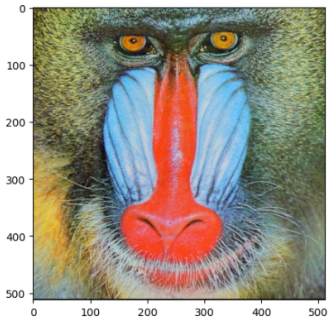


AFTER THAT THE NEXT STEP IS TO CREATE FIGURE SIZE OF THE PICTURE IN AXIS TABLE -

```
fig, ax = plt.subplots(figsize= (5,5))
ax.imshow(img_mpl)
# ax.axis('off')
plt.show()
```

AND HERE IS THE OUTPUT -

```
[9]: fig, ax = plt.subplots(figsize= (5,5))
ax.imshow(img_mpl)
# ax.axis('off')
plt.show()
```



AFTER THAT , THE NEXT STEP IS TO PUT RGB COLOURS IN THE PHOTO -

```
#Display RGB channels

fig, axs = plt.subplots(1,3,figsize = (10,5))
axs[0].imshow(img_mpl[:,:,0], cmap='Reds')
axs[1].imshow(img_mpl[:,:,1], cmap='Greens')
axs[2].imshow(img_mpl[:,:,2], cmap='Blues')
axs[0].axis('off')
axs[1].axis('off')
axs[2].axis('off')
plt.show()
```

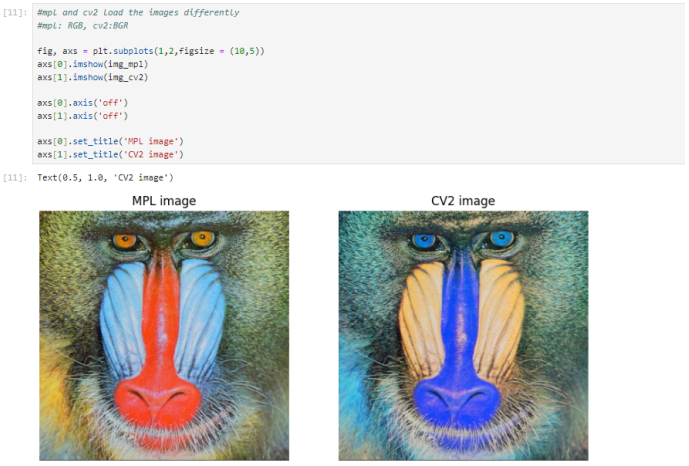
OUTPUT -

```
[10]: #Display RGB channels

fig, axs = plt.subplots(1,3,figsize = (10,5))
axs[0].imshow(img_mpl[:,:,0], cmap='Reds')
axs[1].imshow(img_mpl[:,:,1], cmap='Greens')
axs[2].imshow(img_mpl[:,:,2], cmap='Blues')
axs[0].axis('off')
axs[1].axis('off')
axs[2].axis('off')
plt.show()
```

IN THE NEXT PROCESS WE USE THIS CODE -

```
#mpl and cv2 load the images differently
#mpl: RGB, cv2:BGR

fig, axs = plt.subplots(1,2,figsize = (10,5))
axs[0].imshow(img_mpl)
axs[1].imshow(img_cv2)

axs[0].axis('off')
axs[1].axis('off')

axs[0].set_title('MPL image')
axs[1].set_title('CV2 image')
```

THE OUTPUT -




```
#Converting from BGR to RGB

img_cv2_rgb = cv2.cvtColor(img_cv2, cv2.COLOR_BGR2RGB)
fig, ax = plt.subplots()
ax.imshow(img_cv2_rgb)
ax.axis('off')
plt.show()
```

OUTPUT -

AFTER THIS , IN THE NEXT STEP WE WILL CHECK IF THE IMAGE IS BLUR OR NOT , SO USE THIS CODE - cv2.imwrite('cv2_monkey.jpg', blur)

BUT IF AFTER USING THIS CODE , IF YOU GOT A PROBLEM LIKE THIS -

```
[13]: cv2.imwrite('cv2_monkey.jpg', blur)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[13], line 1
----> 1 cv2.imwrite('cv2_monkey.jpg', blur)

NameError: name 'blur' is not defined
```

THEN USE THIS CODE TO RECTIFY THIS PROCESS -

```
import cv2

# Read the image
img = cv2.imread('C:/Users/ASUS/Downloads/baboon.jpg')

if img is not None:  # Ensure the image was successfully loaded
    # Apply Gaussian Blur
    blur = cv2.GaussianBlur(img, (15, 15), 0)

    # Save the blurred image
    save_status = cv2.imwrite('C:/Users/ASUS/Downloads/cv2_monkey.jpg', blur)

    if save_status:
        print("Image successfully saved as 'cv2_monkey.jpg'")
    else:
        print("Error: Could not save the image.")
else:
    print("Error: Could not load the image. Check the file path.")
```

THIS CODE WILL FIX THE ERROR AND ALSO DO NOT FORGET TO CHANGE THE IMAGE PATH WHICH I HAVE MENTIONED ON THE CODE .

BUT THERE WAS NO ERROR THEN THEIR IS NO NEED TO USE THIS CODE .

THE OUTPUT OF THE CODE - cv2.imwrite('cv2_monkey.jpg', blur) IS -

```
[16]: cv2.imwrite('cv2_monkey.jpg', blur)

[16]: True
```

IN THIS STEP WE WILL UPGRADE THE PICTURE USING -

!pip install imageai --upgrade

```
[17]: !pip install imageai --upgrade

Collecting imageai
  Downloading imageai-3.0.3-py3-none-any.whl.metadata (340 bytes)
Downloading imageai-3.0.3-py3-none-any.whl (69 kB)
Installing collected packages: imageai
Successfully installed imageai-3.0.3
```

THE YOLO FILE PROJECT

HERE YOU HAVE TO DOWNLOAD THE YOLO FILE - tiny-yolov3.pt

download

I HAVE EMBEDED A LINK WHICH WILL REDIRECT YOU TO THE DOWNLOADING PAGE OF THE YOLOV3 FILE.

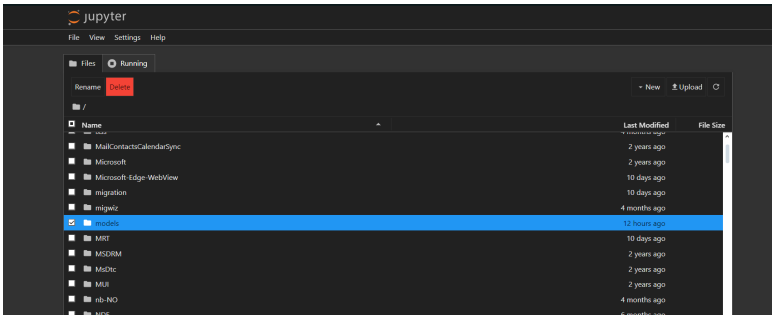
AFTER DOWNLOADING THE FILE , IN THE NEXT STEP YOU HAVE TO CREATE THE MODELS FOLDER ON JUPYTER.

```
import os

# Create the 'models' directory if it doesn't exist
models_dir = "models"
if not os.path.exists(models_dir):
    os.makedirs(models_dir)

print("Models directory is ready!")
```

```
[32]: import os

      # Create the 'models' directory if it doesn't exist
      models_dir = "models"
      if not os.path.exists(models_dir):
          os.makedirs(models_dir)

      print("Models directory is ready!")

      Models directory is ready!
```

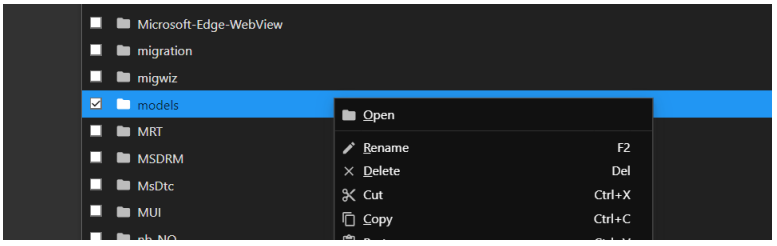
USING THE GIIVEN CODE YOUR MODELS FOLDER WILL BE READY .

NOW YOU CAN ACCESS THE MODEL FOLDER ON THE HOME OF JUPYTER NOTEBOOK.

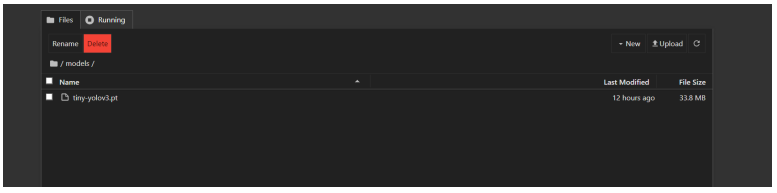


YOU CAN SEE A ✓ MARK ON THE SIDE OF THE FOLDER

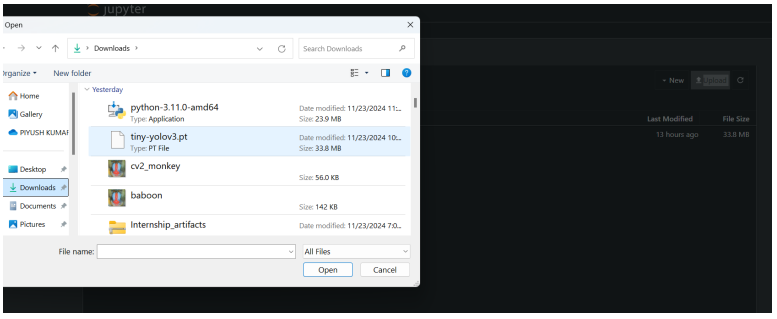
THEN RIGHT-CLICK ON THE FOLDER THEN OPEN THE FOLDER



AFTER THIS YOU CAN SEE I HAVE ALREADY UPLOADED THE DOWNLOADED YOLO FILE IN THE MODEL FOLDER



AFTER OPENING THE MODEL FOLDER THEN CLICK UPLOAD AND IT WILL NAVIGATE YOU TO THE FILE EXPLORER THEN CHOSE THE DOWNLOADED FILE AND IT WILL BE UPLOADED TO THE FOLDER.



THEN YOU ARE READY TO GO !

IN THIS VERY NEXT STEP YOU NEED TO DO SOME MORE INSTALLATION OF MODULES LIKE - torch torchvision torchaudio

```
!pip install torch torchvision torchaudio
```

AFTER INSTALLATION USE THE GIVEN CODE FOR CONFIRMATION -

```
import torch
print(torch.__version__)
print("CUDA available:", torch.cuda.is_available())
```

IN THIS PROCESS WE WILL INSTALL OPEN-CV - PYTHON - HEADLESS

```
!pip install opencv-python-headless
```

WHY IT IS NEED BECAUSE IT IS THE FULL VERSION OF OPEN-CV WITH ADDITIONAL FEATURES

LAST AND FINAL STEP -

CODE -

```
!pip install imageai --upgrade

from imageai.Detection import VideoObjectDetection
import os
import time

# Start timing
start_time = time.time()

# Set the execution path
execution_path = os.getcwd()

# Create the 'models' directory if it doesn't exist
os.makedirs("models", exist_ok=True)

def forFrame(frame_number, output_array, output_count):
    print("FOR FRAME ", frame_number)
    print("Output for each object : ", output_array)
    print("Output count for unique objects : ", output_count)
    print("------------END OF A FRAME --------------")

def forSeconds(second_number, output_arrays, count_arrays, average_output_count):
    print("SECOND : ", second_number)
    print("Array for the outputs of each frame ", output_arrays)
    print("Array for output count for unique objects in each frame : ", count_arrays)
    print("Output average count for unique objects in the last second: ", average_output_count)
    print("------------END OF A SECOND --------------")

def forMinute(minute_number, output_arrays, count_arrays, average_output_count):
    print("MINUTE : ", minute_number)
    print("Array for the outputs of each frame ", output_arrays)
    print("Array for output count for unique objects in each frame : ", count_arrays)
    print("Output average count for unique objects in the last minute: ", average_output_count)
    print("------------END OF A MINUTE --------------")

# Initialize the VideoObjectDetection
video_detector = VideoObjectDetection()
video_detector.setModelTypeAsTinyYOLOv3()

# Path to the model file
model_path = os.path.join(execution_path, "models/tiny-yolov3.pt")
if not os.path.exists(model_path):
    print("Model file not found. Please download tiny-yolov3.pt and place it in the 'models' directory.")
else:
    video_detector.setModelPath(model_path)
    video_detector.loadModel()

    # Path to input video
    input_video_path = os.path.join(execution_path, "C:/Users/ASUS/Downloads/videoplayback.mp4")
    if not os.path.exists(input_video_path):
        print(f"Input video file not found at {input_video_path}. Please provide a valid video file.")
    else:
        # Detect objects in the video
        output_video_path = os.path.join(execution_path, "output_video.mp4")
```

```
        video_detector.detectObjectsFromVideo(
            input_file_path=input_video_path,
            output_file_path=output_video_path,
            frames_per_second=10,
            per_second_function=forSeconds,
            per_frame_function=forFrame,
            per_minute_function=forMinute,
            minimum_percentage_probability=30
        )

        # End timing and calculate the duration
        end_time = time.time()
        execution_duration = end_time - start_time

        print("Video saved at:", output_video_path)
        print("Time taken to run the code:", execution_duration, "seconds")
```

BEFORE EXECUTING THE ENTIRE CODE YOU NEED TO DOWNLOAD ANY SAMPLE VIDEO FROM ONLINE AND THEN YOU HAVE TO COPY THE PATH OF THE VIDEO FROM THE FILE EXPLORER ON YOUR SYSTEM AND PASTE IT ONE THE CODE WHERE I HAVE MENTIONED THE CODE -

```
# Path to input video
input_video_path = os.path.join(execution_path, "C:/Users/ASUS/Downloads/videoplayback.mp4")
if not os.path.exists(input_video_path):
```

"C:\Users\ASUS\Downloads\videoplayback.mp4" - THIS IS THE PATH OF MY VIDEO WHICH I HAVE INSERTED ON THE CODE MENTIONED ABOVE .

NOW AFTER RUNNING THE WHOLE PROGRAM IF YOU GOT THE OUTPUT LIKE THIS -

```
FOR FRAME  1
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  2
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  3
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  4
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  5
Output for each object :  []
```

THEN YOU ARE GOOD TO GO BUT THIS IS THE BEGINNING OUTPUT , AFTER SCROLLING DOWN TO THE END YOU WILL SEE -

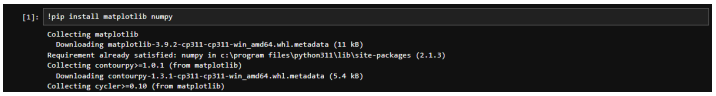
```
FOR FRAME  1456
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  1457
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
FOR FRAME  1458
Output for each object :  []
Output count for unique objects :  {}
------------END OF A FRAME --------------
Video saved at: C:\Windows\System32\output_video.mp4
Time taken to run the code: 61.57493543624878 seconds
```

THIS SHOWS THE CODE EXECUTION IS COMPLETED.

DNN MODULE PROJECT

IN ORDER TO DO THIS MODEL PROJECT YOU NEED TO HAVE SOME MODULES INSTALLED ON YOUR JUPYTER NOTEBOOK.

FIRST ONE IS - !pip install matplotlib numpy



SECOND IS - !pip install opencv-python

pip install opencv-python-headless numpy

AFTER THAT THERE ARE SOME CHANGES YOU NEED TO DO IN YOUR CODE , BUT FIRST YOU NEED TO DOWNLOAD 3 FILES

coco.names.txt

yolov4.cfg

AND THE NEXT IS OF YOLO WEIGHTS FILES - https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.weights

USE THIS GIVEN LINK AS THE FILE SIZE IS 245 MB

NOW CHANGES NEED TO BE DONE ON THE CODE -

AS YOU CAN SEE ON THE GIVEN PICTURE IN —

```
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt

# Load pre-trained DNN model (example with YOLOv4-tiny)
model_config = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.cfg"  # Path to config file
model_weights = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.weights"  # Path to weights file
net = cv2.dnn.readNetFromDarknet(model_config, model_weights)

# Load class names (COCO dataset as example)
with open("C:/Users/ASUS/Downloads/jupyter projects/coco.names.txt", "r") as f:
    class_names = [line.strip() for line in f.readlines()]

# Load video
cap = cv2.VideoCapture("C:/Users/ASUS/Downloads/cars on road.mp4")
```

model_config = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.cfg" # Path to config file
model_weights = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.weights" # Path to weights file

with open("C:/Users/ASUS/Downloads/jupyter projects/coco.names.txt", "r") as f:

cap = cv2.VideoCapture("C:/Users/ASUS/Downloads/cars on road.mp4")

YOU NEED TO SET THE PATH OF THE DOWNLOADED FILES FROM YOUR SYSTEM THEN THE CODE WILL WORK. AND FOR THE 4TH LINE ABOUT VIDEO CAPTURE YOU HAVE TO USE A SAMPLE VIDEO FOR THAT CODE BUT IT SHOULD BE DIFFERENT FROM THE PREVIOUS ONE .

AFTER THE CODE IS EXECUTED YOU CAN SEE THAT THE CODE WILL CAPTURE THE OBJECTS FROM THE VIDEO AND IT WILL CONVERT THE VIDEO IN MULTIPLE PICTURES.

NOW USE THIS CODE -

```
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt

# Load pre-trained DNN model (example with YOLOv4-tiny)
model_config = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.cfg"  # Path to config file
model_weights = "C:/Users/ASUS/Downloads/jupyter projects/yolov4.weights"  # Path to weights file
net = cv2.dnn.readNetFromDarknet(model_config, model_weights)

# Load class names (COCO dataset as example)
```

```
with open("C:/Users/ASUS/Downloads/jupyter projects/coco.names.txt", "r") as f:
    class_names = [line.strip() for line in f.readlines()]

# Load video
cap = cv2.VideoCapture("C:/Users/ASUS/Downloads/jupyter projects/cars on road.mp4")

# Check if the video file opened successfully
if not cap.isOpened():
    print("Error: Could not open video file.")
    exit()

# Get video properties for saving
frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(cap.get(cv2.CAP_PROP_FPS))
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_video = cv2.VideoWriter("processed_video.avi", fourcc, fps, (frame_width, frame_height))

# Detection threshold
conf_threshold = 0.5
nms_threshold = 0.4

# Initialize tracking variables
tracking_objects = {}
track_id = 0

# Function to display a frame in Jupyter
def display_frame(frame):
    """Display a frame using Matplotlib in Jupyter."""
    plt.axis('off')
    plt.imshow(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
    plt.show()

# Function to detect objects
def detect_objects(frame):
    # Prepare the image as input for the model
    blob = cv2.dnn.blobFromImage(frame, scalefactor=1 / 255.0, size=(416, 416), swapRB=True, crop=False)
    net.setInput(blob)

    # Get the names of the output layers
    layer_names = net.getLayerNames()
    output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

    # Perform forward pass and get detections
    detections = net.forward(output_layers)

    boxes = []
    class_ids = []
    confidences = []

    # Parse detections
    for output in detections:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

            if confidence > conf_threshold:  # Set your confidence threshold
                center_x = int(detection[0] * frame.shape[1])
                center_y = int(detection[1] * frame.shape[0])
                w = int(detection[2] * frame.shape[1])
                h = int(detection[3] * frame.shape[0])

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                class_ids.append(class_id)
                confidences.append(float(confidence))

    # Apply Non-Maximum Suppression (NMS) to refine boxes
    indices = cv2.dnn.NMSBoxes(boxes, confidences, conf_threshold, nms_threshold)

    # Extract resulting boxes, class IDs, and confidences
    result_boxes = []
    result_class_ids = []
    result_confidences = []
```

```
    if len(indices) > 0:
        for i in indices.flatten():
            result_boxes.append(boxes[i])
            result_class_ids.append(class_ids[i])
            result_confidences.append(confidences[i])

    return result_boxes, result_class_ids, result_confidences

# Ensure output video writer initializes correctly
output_video = cv2.VideoWriter("processed_video.avi", fourcc, fps, (frame_width, frame_height))
if not output_video.isOpened():
    print("Error: Could not initialize video writer.")
    exit()

# Process video frames
count = 0
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    count += 1
    center_points_cur_frame = []

    # Detect objects
    boxes, class_ids, confidences = detect_objects(frame)
    for i, box in enumerate(boxes):
        x, y, w, h = box
        cx = int((x + x + w) / 2)
        cy = int((y + y + h) / 2)
        center_points_cur_frame.append((cx, cy))

        # Draw rectangle and label
        label = f"{class_names[class_ids[i]]}: {confidences[i]:.2f}"
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # Update tracking objects
    new_tracking_objects = {}
    for pt in center_points_cur_frame:
        same_object_detected = False
        for object_id, prev_pt in tracking_objects.items():
            distance = math.hypot(prev_pt[0] - pt[0], prev_pt[1] - pt[1])
            if distance < 35:
                new_tracking_objects[object_id] = pt
                same_object_detected = True
                break

        if not same_object_detected:
            new_tracking_objects[track_id] = pt
            track_id += 1

    tracking_objects = new_tracking_objects

    # Draw tracking points
    for object_id, pt in tracking_objects.items():
        cv2.circle(frame, pt, 5, (0, 0, 255), -1)
        cv2.putText(frame, str(object_id), (pt[0] - 10, pt[1] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

    # Write frame to output video
    output_video.write(frame)

    # Display frame in Jupyter
    display_frame(frame)

    if count > 100:  # Limit to 100 frames for testing
        break

# Release resources
cap.release()
output_video.release()

print("Video processing complete. Output saved as 'processed_video.avi'.")
```

THE OUTPUT AFTER EXECUTION OF CODE -