In [ ]:

```
##Lab 15
!pip install ultralytics

from ultralytics import YOLO
import cv2
import matplotlib.pyplot as plt

model = YOLO('yolov8n.pt')  # small YOLOv8 model

img_path = 'sample.jpg'
results = model(img_path)

for result in results:
    boxes = result.boxes.xyxy.cpu().numpy()
    scores = result.boxes.conf.cpu().numpy()
    classes = result.boxes.cls.cpu().numpy()

    img = cv2.imread(img_path)
    for box, score, cls in zip(boxes, scores, classes):
        x1, y1, x2, y2 = map(int, box)
        label = f"{int(cls)}:{score:.2f}"
        cv2.rectangle(img, (x1, y1), (x2, y2), (0,255,0), 2)
        cv2.putText(img, label, (x1, y1-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

Exp-15 : Implement a YOLO Model Detect Objects.

Aim:

To implement a YOLO model for real-time object detection.

Description:

YOLO divides an image into grids and predicts bounding boxes and class probabilities in a single pass, enabling real time detection.

It is based on a single CNN performing both localization and classification.

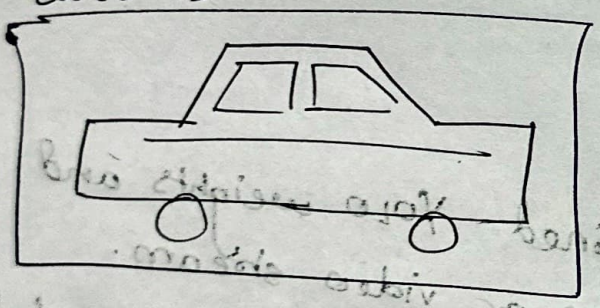
Procedure:

1.) Load pre-trained YOLO weights and configuration
2.) Load an image or video stream.
3.) Perform forward propagation to get bounding boxes and class scores.
4.) Apply Non-Max Suppression (NMS) to remove overlapping boxes.
5.) Display detected objects with labels.

Output:

YOLOv5 object Detection (COCO Pretrained)

car 0.93



Pseudocode:

Load YOLO weights and config
Read Input Image
forward pass through YOLO network
Extract boxes, confidence, and class IDs
Apply NMS
Draw boxes and labels on Image
Display result.

Observation:

YOLO detects multiple objects accurately in real time with bounding boxes and class labels.

Result:

The YOLO model was successfully implemented for object detection, acheiving fast and accurate recognition on test Images.