Dataset preparation:

- ☐ Downloaded random 50-60 images from google of red and white bullseye.
    - ❖ Created a dataset using roboflow , also added some data augmentation like rotation , flip etc, which created a dataset of approx 150 images.
    - ❖ But my model failed as I was getting very poor accuracy.

- ☐ After this , I used a custom dataset uploaded on roboflow.
    - ❖ It had approx 800 images and applied data augmentation on it.
    - ❖ it began to overfit—predicting **all** bullseyes, regardless of their actual color, as red and white.
- ☐ Finally prepared a vast dataset containing all red and white bullseye along with other bullseye, marked other bullseye as null in roboflow.

Model Training and Evaluation:

- ☐ Trained the dataset using yolov8.
- ☐ Watched tutorials on youtube on yolov8 and opencv
- ☐ Tested the model on several images and also implemented the model on video

Important highlights:

- What I observed while testing was it was still overfitting(not like the previous)
  To solve this I observed that its confidence score on images which are not red and white bullseye was approx. less than 0.7,0.8 and and the images which are red and white were greater than 0.9.
- So I implemented code to consider those bullseyes only whose confidence score was greater than 0.8.
- Then I tested it on the same images and it worked.
- Implemented same model on a video which was working fine.

- ☐ Converted the model to onnx and results of evaluation were stored in results.png file Which consisted of several metrics like mAP and bounding box loss , recall etc.

Tools used:

- Python
- Yolov8 model
- Python libraries opencv , ultralytics.
- Roboflow

Key learnings:

- **Importance of a Diverse Dataset:**
  One of the most important takeaways from this project was realizing how crucial the dataset is to a model's performance. Initially, training with only red and white bullseyes caused the model to overfit, resulting in misclassification of other bullseyes.

Expanding the dataset to include various colors and patterns significantly improved the accuracy and generalization of the model.
As Model fails to detect the bullseye when it is to far as it was trained on bullseye Which were close to the camera, So we need images taken from drone when it is Flying, training it on that dataset will definitely improve our drone's performance.

- **Understanding Overfitting and Generalization:**
  The project provided a hands-on example of how overfitting happens when the model sees too few variations during training. By diversifying the training data, the model learned to generalize better to unseen inputs.
- **OpenCV:**
  It was primarily used for testing and visualizing the outputs of the YOLO model. It helped in reading and preprocessing images, displaying detection results with bounding boxes, and debugging the model's performance through real-time visualization. This experience highlighted how essential OpenCV is for evaluating and presenting computer vision results effectively.