**Documentation**

**Our approach:**

Our approach is divided into three main steps, which we will discuss further in the next part, but in a nutshell, our core idea is based on object detection.So far, the items we must anticipate are crop fields.

The labels we have are in the centre of the image.We built bounding boxes in order to make object detection work So the squares that enclose the circles are our bounding boxes.

**Section 1:**

In this section we will introduce our Yolo labels.

first , in the begining of our notebook we started by loading the data from zindi to google colab

then we started adding new features to our data such as

* extra : this give us an insight if the row belongs to extra train or to unique train
* r : is the radius in km
* delta : is distance between x and y
* ratio : is the distance delta divided by r
* is\_center : is about the distance if the delta <0.005 then is\_center will take 1 else it is 0

second we worked on extracting bounding boxes from image that’s why we created the following columns after loading all images

* win : is the calculated from the radius
* yolo : contains the cordinates for our bounding box

**Section 2:**

In this section we trained our object detection model which is YOLOV5, the choice of our model was based on its performance on previous object detection tasks.

To prevent our model from overfitting, we used stratified kfold and splited our data to 5 folds based on the year column.

So after various fine tuning of the hyperparameters , each fold was trained with the following parameters:

lr0: 0.005 # 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)

lrf: 0.1 # 0.2 # final OneCycleLR learning rate (lr0 \* lrf)

momentum: 0.937 # SGD momentum/Adam beta1

weight\_decay: 0.0005 # optimizer weight decay 5e-4

warmup\_epochs: 3.0 # warmup epochs (fractions ok)

warmup\_momentum: 0.8 # warmup initial momentum

warmup\_bias\_lr: 0.1 # warmup initial bias lr

box: 0.05 # box loss gain

cls: 0.5 # cls loss gain

cls\_pw: 1.0 # cls BCELoss positive\_weight

obj: 1.0 # obj loss gain (scale with pixels)

obj\_pw: 1.0 # obj BCELoss positive\_weight

iou\_t: 0.20 # IoU training threshold

anchor\_t: 4.0 # anchor-multiple threshold

# anchors: 3 # anchors per output layer (0 to ignore)

fl\_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)

hsv\_h: 0.015 # image HSV-Hue augmentation (fraction)

hsv\_s: 0.7 # image HSV-Saturation augmentation (fraction)

hsv\_v: 0.4 # image HSV-Value augmentation (fraction)

degrees: 0.0 # image rotation (+/- deg) # ========== ADDED

translate: 0.1 # image translation (+/- fraction)

scale: 0.5 # image scale (+/- gain)

shear: 0.0 # image shear (+/- deg)

perspective: 0.0 # image perspective (+/- fraction), range 0-0.001

flipud: 0.5 # image flip up-down (probability) # ========== ADDED

fliplr: 0.5 # image flip left-right (probability)

mosaic: 1.0 # image mosaic (probability)

mixup: 0.0 # image mixup (probability)

and a batch size of 128 and epochs of 80.

**Section 3:**

Since for each location of the crop we have 4 images at diffrent times jun17,dec17,jun18,dec18 so basicaly we will have 4 predictions one for each image.

That’s why we used voting mechanism where we take in consideration all the predictions and take the median value of all them in order to center the true value as close as possible.

All the voting mechanism was done for each fold and the final prediction was the average value of all 5 folds predictions