Data Analysis Report capstone project.

Introduction

This project is based on object detection where the bones and joints are detected and from that we will recognize the age of the child whether the child is healthy or not in other words the bones of the child are grown as they need to be or not.

For this purpose, we will be using yolov5 framework and link for setting everything is given below.

Object Detection of Joints on the Bone age Dataset with YOLOv5

Materials and methods

The dataset

We utilize the dataset from the Pediatric Bone Age Challenge [6] organized by the Radiological Society of North America (RSNA). This dataset is now freely available and can be accessed over the website. It is a rather large dataset, e.g. when compared with Digital Hand Atlas [7], and consists of 12611 training images, as well as 1425 validation images and 200 test images. For our purposes it is enough to use the training set only. The dataset consists of 54.2% male and 45.8% female hands. The age distribution is not uniform, but reflects the distribution in the clinical routine, i.e. while there are rather few images of infants with 0-4 years (4.0%) and adolescents over 16 years (2.7%), there are many more for the 4-8 years (21.9%) as well as 8-12 years (36.2%) and 12-16 years (35.2%). The images vary in their size and quality, e.g. especially infants hands are underexposed to minimize radiation exposure and thus include noise.

This piacular paragraph is referred by the research paper

Reference of that is given below with already research which is going on and now further improvement on research.

Reference:- https://doi.org/10.1371/journal.pone.0207496 by S. Koitka, A. Demircioglu, M.S. Kim, C.M. Friedrich, F. Nensa.

The goal of this project is object detection by using bb boxes and then applying various models to find the best model to detect the age of the bone of the child to know whether the child is healthy or not based on the bones whether the

child is grown properly with this purpose first we have xrays images we will draw boxes and understand the joints properly. Trained to validate and apply different models to find the best output we can.

Input:-

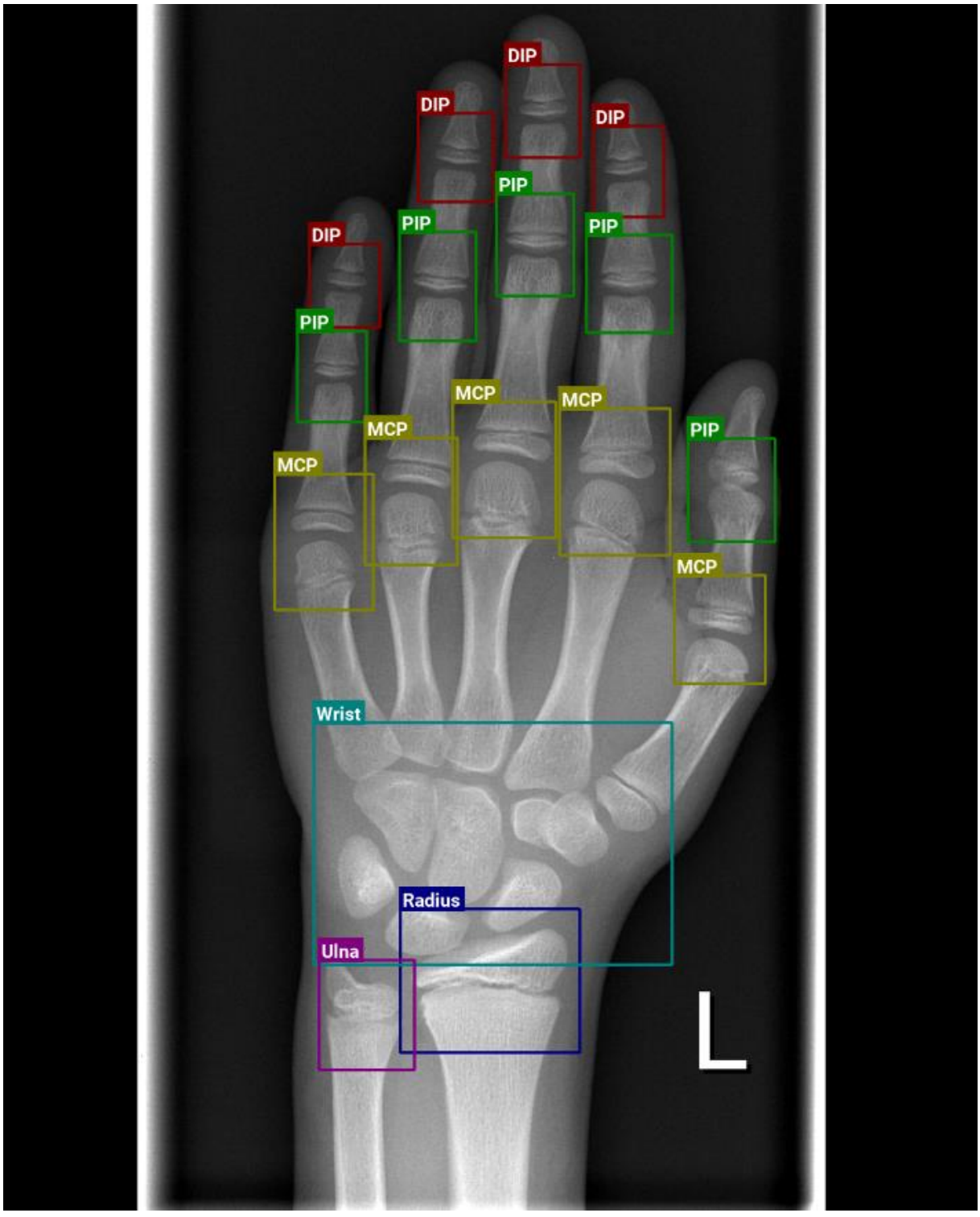
Goal:-

To have BB boxes as shown below with a good model to analyze or predict data analysis.

Conclusion and Analysis results are shared below to find the best model.

Each and every steps with the input results are shared below.

Goal



To perform the following project yolo5 framework has been used.

The data set for BB-Boxes has been stored in CSV file format.

The data set for validation and training has been provided in CSV format. Earlier the dataset was the Xray images from which the BB-boxes was maintained by using the yoloframework this has been stored in csv file format.

All the dataset and everything is given for the checking purpose.

Step 1:- Preparation for Dataset which is images.

```
nimisha.vilayatarani002@99e274fbd95c:/raid/datashare/RSNA2017$ ls
boneage-training-dataset  boneage-validation-dataset  boneage_yolo  rsna-boneage-ossification-roi-detection
```

```
nimisha.vilayatarani002@99e274fbd95c:/raid/datashare/RSNA2017$ cd boneage-training-dataset
nimisha.vilayatarani002@99e274fbd95c:/raid/datashare/RSNA2017/boneage-training-dataset$ ls
10000.png  10836.png  11675.png  12504.png  13328.png  14127.png  14878.png  1584.png  2407.png  3234.png  4056.png  5072.png  5891.png  6708.png  7536.png  8361.png  9182.png
10001.png  10837.png  11676.png  12505.png  13329.png  14128.png  14879.png  1585.png  2408.png  3235.png  4057.png  5073.png  5892.png  6709.png  7537.png  8362.png  9183.png
10002.png  10838.png  11677.png  12506.png  13330.png  14129.png  1488.png  1586.png  2409.png  3236.png  4058.png  5074.png  5893.png  6710.png  7538.png  8363.png  9184.png
10003.png  10839.png  11678.png  12507.png  13331.png  14130.png  14880.png  1587.png  2410.png  3237.png  4059.png  5075.png  5894.png  6712.png  7539.png  8365.png  9185.png
10004.png  10841.png  11679.png  12509.png  13333.png  14131.png  14881.png  1588.png  2411.png  3238.png  4060.png  5076.png  5895.png  6713.png  7540.png  8366.png  9186.png
10005.png  10842.png  11681.png  12511.png  13334.png  14132.png  14882.png  1589.png  2412.png  3239.png  4062.png  5077.png  5896.png  6714.png  7541.png  8367.png  9187.png
10006.png  10843.png  11682.png  12512.png  13335.png  14133.png  14883.png  1590.png  2413.png  3240.png  4063.png  5079.png  5897.png  6715.png  7542.png  8368.png  9188.png
10007.png  10844.png  11683.png  12513.png  13336.png  14134.png  14884.png  1591.png  2414.png  3241.png  4064.png  5080.png  5898.png  6717.png  7544.png  8369.png  9189.png
10008.png  10845.png  11685.png  12514.png  13338.png  14135.png  14885.png  1592.png  2415.png  3242.png  4065.png  5081.png  5899.png  6718.png  7545.png  8370.png  9190.png
10009.png  10846.png  11686.png  12515.png  13339.png  14136.png  14886.png  1593.png  2416.png  3244.png  4066.png  5082.png  5900.png  6719.png  7546.png  8371.png  9191.png
10010.png  10847.png  11687.png  12516.png  13340.png  14137.png  1489.png  1594.png  2417.png  3245.png  4067.png  5084.png  5901.png  6720.png  7547.png  8373.png  9192.png
10011.png  10848.png  11688.png  12517.png  13341.png  14138.png  14890.png  1595.png  2418.png  3246.png  4068.png  5085.png  5902.png  6722.png  7548.png  8374.png  9193.png
10012.png  10849.png  11689.png  12518.png  13342.png  14139.png  14891.png  1596.png  2419.png  3248.png  4069.png  5086.png  5903.png  6723.png  7549.png  8375.png  9194.png
10013.png  10850.png  11690.png  12519.png  13343.png  1414.png  14894.png  1597.png  2421.png  3249.png  4071.png  5087.png  5904.png  6724.png  7550.png  8377.png  9195.png
10014.png  10851.png  11691.png  12520.png  13344.png  14140.png  14895.png  1598.png  2422.png  3250.png  4072.png  5088.png  5905.png  6725.png  7551.png  8378.png  9196.png
10015.png  10852.png  11693.png  12521.png  13346.png  14141.png  14896.png  1599.png  2423.png  3251.png  4073.png  5089.png  5906.png  6726.png  7553.png  8379.png  9197.png
10016.png  10853.png  11695.png  12522.png  13347.png  14142.png  14898.png  1600.png  2424.png  3252.png  4074.png  5090.png  5907.png  6727.png  7554.png  8380.png  9198.png
10017.png  10854.png  11696.png  12523.png  13348.png  14143.png  14899.png  1601.png  2425.png  3253.png  4075.png  5091.png  5908.png  6728.png  7555.png  8381.png  9199.png
10019.png  10855.png  11697.png  12524.png  13349.png  14144.png  14900.png  1602.png  2427.png  3254.png  4076.png  5092.png  5909.png  6729.png  7556.png  8382.png  9200.png
10020.png  10856.png  11698.png  12525.png  13350.png  14145.png  14901.png  1604.png  2428.png  3255.png  4077.png  5094.png  5910.png  6731.png  7557.png  8383.png  9201.png
10021.png  10857.png  11699.png  12527.png  13351.png  14146.png  14902.png  1605.png  2429.png  3256.png  4078.png  5096.png  5911.png  6732.png  7558.png  8385.png  9202.png
10022.png  10858.png  11700.png  12528.png  13352.png  14147.png  14903.png  1606.png  2430.png  3257.png  4079.png  5097.png  5912.png  6733.png  7559.png  8386.png  9203.png
10023.png  10859.png  11701.png  12529.png  13353.png  14148.png  14904.png  1607.png  2431.png  3258.png  4080.png  5098.png  5913.png  6734.png  7560.png  8387.png  9204.png
10024.png  10860.png  11703.png  12530.png  13355.png  14149.png  14905.png  1608.png  2432.png  3259.png  4081.png  5100.png  5914.png  6735.png  7561.png  8388.png  9206.png
10025.png  10861.png  11704.png  12531.png  13356.png  1415.png  14906.png  1609.png  2433.png  3260.png  4082.png  5101.png  5915.png  6736.png  7562.png  8390.png  9207.png
10026.png  10862.png  11705.png  12532.png  13357.png  14151.png  14907.png  1610.png  2435.png  3261.png  4083.png  5102.png  5916.png  6737.png  7563.png  8391.png  9208.png
10029.png  10863.png  11706.png  12533.png  13358.png  14152.png  14908.png  1612.png  2436.png  3262.png  4084.png  5103.png  5917.png  6738.png  7564.png  8392.png  9209.png
10030.png  10864.png  11707.png  12534.png  13359.png  14153.png  14909.png  1613.png  2437.png  3263.png  4085.png  5104.png  5918.png  6739.png  7565.png  8393.png  9210.png
10031.png  10865.png  11708.png  12535.png  13360.png  14154.png  1491.png  1614.png  2438.png  3264.png  4087.png  5105.png  5919.png  6740.png  7566.png  8394.png  9211.png
10033.png  10866.png  11709.png  12536.png  13361.png  14155.png  14910.png  1615.png  2439.png  3265.png  4089.png  5106.png  5920.png  6741.png  7567.png  8395.png  9212.png
10035.png  10867.png  11710.png  12537.png  13362.png  14156.png  14911.png  1616.png  2440.png  3266.png  4090.png  5107.png  5921.png  6742.png  7569.png  8396.png  9213.png
10036.png  10868.png  11711.png  12538.png  13363.png  14157.png  14912.png  1617.png  2442.png  3267.png  4091.png  5108.png  5922.png  6743.png  7570.png  8397.png  9214.png
10037.png  10869.png  11712.png  12539.png  13364.png  14159.png  14913.png  1618.png  2443.png  3268.png  4092.png  5109.png  5923.png  6744.png  7571.png  8398.png  9215.png
10038.png  10870.png  11713.png  12540.png  13365.png  1416.png  14914.png  1619.png  2444.png  3269.png  4093.png  5110.png  5925.png  6746.png  7572.png  8399.png  9216.png
10039.png  10871.png  11714.png  12541.png  13366.png  14160.png  14915.png  1620.png  2445.png  3270.png  4094.png  5111.png  5926.png  6747.png  7573.png  8400.png  9217.png
```

Step2:- Dataset for validation

```
nimisha.vilayatarani002@99e274fbd95c:/raid/datashare/RSNA2017/boneage-validation-dataset/boneage-validation-dataset-1$ ls
1386.png  1805.png  2244.png  2770.png  3131.png  3570.png  4051.png  4812.png  5222.png  5722.png  6282.png  6703.png  7126.png  7568.png  7991.png  8486.png  8874.png  9442.png
1392.png  1809.png  2252.png  2796.png  3139.png  3575.png  4061.png  4851.png  5225.png  5728.png  6285.png  6704.png  7127.png  7580.png  7992.png  8492.png  8875.png  9449.png
1397.png  1822.png  2257.png  2800.png  3143.png  3603.png  4070.png  4854.png  5231.png  5736.png  6287.png  6711.png  7175.png  7587.png  7998.png  8503.png  8880.png  9482.png
1401.png  1824.png  2264.png  2803.png  3182.png  3605.png  4086.png  4862.png  5266.png  5746.png  6290.png  6716.png  7194.png  7597.png  8005.png  8508.png  8886.png  9483.png
1410.png  1836.png  2281.png  2809.png  3199.png  3608.png  4088.png  4863.png  5272.png  5771.png  6302.png  6721.png  7196.png  7600.png  8018.png  8529.png  8893.png  9490.png
1413.png  1842.png  2302.png  2816.png  3201.png  3620.png  4095.png  4864.png  5288.png  5787.png  6304.png  6730.png  7200.png  7623.png  8026.png  8534.png  8985.png  9499.png
1421.png  1847.png  2314.png  2820.png  3204.png  3625.png  4119.png  4888.png  5324.png  5799.png  6323.png  6745.png  7221.png  7629.png  8031.png  8535.png  9007.png  9512.png
1449.png  1859.png  2315.png  2822.png  3208.png  3626.png  4125.png  4892.png  5350.png  5801.png  6324.png  6752.png  7223.png  7645.png  8033.png  8549.png  9014.png  9514.png
1450.png  1861.png  2334.png  2824.png  3221.png  3629.png  4127.png  4910.png  5357.png  5802.png  6333.png  6755.png  7226.png  7668.png  8059.png  8568.png  9015.png  9515.png
1456.png  1862.png  2335.png  2831.png  3222.png  3640.png  4156.png  4926.png  5360.png  5836.png  6347.png  6766.png  7227.png  7675.png  8070.png  8569.png  9049.png  9536.png
1465.png  1875.png  2338.png  2833.png  3228.png  3647.png  4158.png  4932.png  5388.png  5842.png  6361.png  6797.png  7241.png  7679.png  8072.png  8576.png  9050.png  9558.png
1490.png  1881.png  2362.png  2840.png  3243.png  3659.png  4169.png  4957.png  5392.png  5846.png  6367.png  6815.png  7247.png  7684.png  8090.png  8588.png  9054.png  9559.png
1506.png  1882.png  2377.png  2842.png  3247.png  3683.png  4171.png  4964.png  5398.png  5847.png  6373.png  6818.png  7275.png  7695.png  8126.png  8608.png  9069.png  9570.png
1510.png  1890.png  2392.png  2860.png  3276.png  3699.png  4178.png  4967.png  5412.png  5865.png  6388.png  6822.png  7280.png  7699.png  8135.png  8611.png  9079.png  9576.png
1528.png  1900.png  2397.png  2863.png  3290.png  3727.png  4226.png  4970.png  5417.png  5871.png  6393.png  6823.png  7288.png  7705.png  8136.png  8612.png  9080.png  9580.png
1532.png  1908.png  2399.png  2869.png  3296.png  3733.png  4242.png  4975.png  5419.png  5880.png  6396.png  6849.png  7292.png  7715.png  8196.png  8626.png  9088.png  9587.png
1537.png  1914.png  2420.png  2891.png  3298.png  3761.png  4251.png  4981.png  5451.png  5883.png  6404.png  6864.png  7293.png  7720.png  8206.png  8646.png  9101.png  9588.png
1539.png  1924.png  2426.png  2910.png  3314.png  3762.png  4261.png  4987.png  5454.png  5924.png  6448.png  6865.png  7310.png  7722.png  8207.png  8649.png  9102.png  9607.png
1552.png  1930.png  2434.png  2915.png  3317.png  3769.png  4267.png  4994.png  5467.png  5933.png  6459.png  6866.png  7318.png  7725.png  8227.png  8663.png  9111.png  9612.png
1561.png  1946.png  2441.png  2919.png  3319.png  3774.png  4274.png  4998.png  5470.png  5944.png  6467.png  6870.png  7319.png  7733.png  8242.png  8665.png  9117.png  9637.png
1564.png  1950.png  2459.png  2931.png  3322.png  3795.png  4283.png  5018.png  5490.png  5953.png  6472.png  6889.png  7321.png  7734.png  8249.png  8670.png  9118.png  9638.png
1569.png  1960.png  2463.png  2940.png  3326.png  3814.png  4292.png  5025.png  5498.png  5971.png  6477.png  6891.png  7332.png  7768.png  8260.png  8671.png  9143.png  9641.png
1573.png  1970.png  2472.png  2941.png  3333.png  3834.png  4317.png  5032.png  5506.png  6012.png  6496.png  6892.png  7338.png  7771.png  8262.png  8674.png  9152.png  9647.png
1579.png  1975.png  2474.png  2945.png  3336.png  3841.png  4335.png  5033.png  5517.png  6023.png  6505.png  6919.png  7371.png  7777.png  8279.png  8689.png  9161.png  9650.png
1583.png  1981.png  2524.png  2960.png  3340.png  3842.png  4336.png  5034.png  5523.png  6056.png  6507.png  6928.png  7377.png  7786.png  8281.png  8690.png  9180.png  9664.png
1603.png  1991.png  2536.png  2967.png  3358.png  3852.png  4338.png  5037.png  5547.png  6080.png  6514.png  6930.png  7378.png  7790.png  8284.png  8691.png  9205.png  9669.png
1611.png  2016.png  2542.png  2968.png  3402.png  3853.png  4348.png  5044.png  5583.png  6087.png  6518.png  6938.png  7400.png  7803.png  8296.png  8694.png  9231.png  9673.png
1622.png  2018.png  2548.png  2972.png  3404.png  3857.png  4349.png  5046.png  5607.png  6091.png  6584.png  6944.png  7401.png  7804.png  8297.png  8697.png  9240.png  9674.png
1633.png  2019.png  2549.png  2973.png  3422.png  3862.png  4565.png  5050.png  5608.png  6093.png  6603.png  6945.png  7407.png  7814.png  8298.png  8701.png  9241.png  9675.png
1655.png  2036.png  2561.png  2981.png  3428.png  3868.png  4572.png  5071.png  5609.png  6100.png  6615.png  6957.png  7408.png  7816.png  8300.png  8725.png  9246.png  9679.png
1657.png  2047.png  2569.png  2995.png  3440.png  3869.png  4581.png  5078.png  5612.png  6101.png  6623.png  6969.png  7417.png  7824.png  8306.png  8729.png  9286.png  9685.png
1676.png  2054.png  2593.png  3009.png  3442.png  3879.png  4584.png  5083.png  5615.png  6165.png  6633.png  6970.png  7432.png  7846.png  8317.png  8731.png  9309.png  9690.png
1687.png  2064.png  2602.png  3044.png  3450.png  3895.png  4594.png  5093.png  5618.png  6173.png  6639.png  6971.png  7444.png  7853.png  8332.png  8733.png  9312.png  9694.png
1693.png  2068.png  2610.png  3045.png  3452.png  3932.png  4655.png  5095.png  5624.png  6176.png  6642.png  7004.png  7459.png  7867.png  8337.png  8741.png  9326.png  9697.png
1731.png  2090.png  2619.png  3046.png  3458.png  3944.png  4658.png  5099.png  5635.png  6177.png  6649.png  7039.png  7476.png  7879.png  8342.png  8765.png  9327.png  9708.png
```

Data representation in yolo5.

The column filename refers to the underlying X-ray image with width as image width and height as image height. The columns xmin, ymin, xmax, ymax define

the corner pixels of a bounding box (BBox) of the X-ray image with class as the joint class. Multiple bounding boxes (BBox) are possible per X-ray image.

Train DataFrame:

	filename	width	height	class	xmin	ymin	xmax	ymax
0	6429.png	1092	1491	DIP	181	423	288	543
1	6429.png	1092	1491	DIP	344	223	447	337
2	6429.png	1092	1491	DIP	502	136	613	264
3	6429.png	1092	1491	DIP	739	183	841	302
4	6429.png	1092	1491	PIP	906	580	1024	715

Validation DataFrame:

	filename	width	height	class	xmin	ymin	xmax	ymax
0	2382.png	1130	1468	DIP	183	580	247	653
1	2382.png	1130	1468	DIP	324	403	385	472
2	2382.png	1130	1468	DIP	508	334	574	412
3	2382.png	1130	1468	DIP	710	386	771	458
4	2382.png	1130	1468	PIP	921	820	991	902

Step 3:- Code for conversion in yolo5 format is:-

```
import os, shutil
import pandas as pd
import numpy as np

# returns a YOLO formatted bbox
def to_yolo_format(bboxen, class_names):
    class_idx = class_names.index(bboxen['class'])

    # calculate normalized xywh coordinates of bbox
    x_center = ((bboxen['xmin'] + bboxen['xmax']) / 2.0) / bboxen['width']
    y_center = ((bboxen['ymin'] + bboxen['ymax']) / 2.0) / bboxen['height']
    width = (bboxen['xmax'] - bboxen['xmin']) / bboxen['width']
    height = (bboxen['ymax'] - bboxen['ymin']) / bboxen['height']
```

Last executed at 2024-06-20 09:44:21 in 483ms

```
ROOT_DIR = os.path.abspath(os.getcwd()) # current path of the notebook
DATA_DIR = "/raid/datashare/RSNA2017/"
IMAGE_FILES = os.path.join(DATA_DIR, "boneage-training-dataset/")
TRAIN_BBOXEN = os.path.join(DATA_DIR, "boneage_yolo/train.csv")
VALID_BBOXEN = os.path.join(DATA_DIR, "boneage_yolo/valid.csv")
YOLO_DATASET_ROOT = os.path.join(ROOT_DIR, "boneage_yolo/")

# load ROIs CSVs for training and validation images
train_bboxen = pd.read_csv(TRAIN_BBOXEN, header=0)
valid_bboxen = pd.read_csv(VALID_BBOXEN, header=0)

# extract unique class names
CLASS_NAMES = train_bboxen['class'].unique().tolist()
```

```
DATASET_YAML = (f"path: {YOLO_DATASET_ROOT}  # dataset root dir\n"
                "train: images/train  # train images (relative to 'path')\n"
                "val: images/valid  # val images (relative to 'path')\n"
                "test:  # test images (optional)\n"
                "\n"
                "# Classes\n"
                f"nc: {len(CLASS_NAMES)} # number of classes\n"
                f"names: {CLASS_NAMES} # class names")

with open(os.path.normpath(f"{YOLO_DATASET_ROOT}/boneage_yolo.yaml"), "w") as f:
        f.write(DATASET_YAML)
```

executed at 2024-06-20 10:00:30 in 6ms

Step 4:- just for ease create a button to code on the terminal by just one click we can go directly go to the terminal.

```
from IPython.display import display, HTML

# Define the HTML for buttons with the full URLs
html_buttons = """
<div style="display: flex; justify-content: space-between; width: 400px; margin-top: 20px;">
    <button onclick="window.open('http://172.22.50.111:9000/user/nimisha.vilayatarani002/terminals/1', '_blank')" style="padding: 10px 20px; font-size: 16px;">Open Terminal</button>

"""

# Display the HTML
display(HTML(html_buttons))

```

Last executed at 2024-06-20 10:54:34 in 4ms

Open Terminal

```
1 cd yolov5
```

```
Last executed at 2024-06-20 10:58:32 in 6ms
```

```
/home/nimisha.vilayatarani002/yolov5
```

```
1
```

Step:5 setup yolo5 frame work

```
1 pip install --no-cache-dir -r requirements.txt codecarbon
```

```
ast executed at 2024-06-20 10:59:08 in 4.11s
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://pypi.ngc.nvidia.com
Requirement already satisfied: codecarbon in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (2.4.2)
Requirement already satisfied: gitpython>=3.1.30 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 5)) (3.1.43)
Requirement already satisfied: matplotlib>=3.3 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 6)) (3.8.3)
Requirement already satisfied: numpy>=1.23.5 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 7)) (1.26.4)
Requirement already satisfied: opencv-python>=4.1.1 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 8)) (4.9.0)
Requirement already satisfied: pillow>=10.3.0 in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (from -r requirements.t
(10.3.0)
Requirement already satisfied: psutil in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 10)) (5.9.8)
Requirement already satisfied: PyYAML>=5.3.1 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 11)) (6.0.1)
Requirement already satisfied: requests>=2.32.0 in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (from -r requirements
2)) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 13)) (1.12.0)
Requirement already satisfied: thop>=0.1.1 in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (from -r requirements.txt
(0.1.1.post2209072238)
Requirement already satisfied: torch>=1.8.0 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 15)) (2.2.1)
Requirement already satisfied: torchvision>=0.9.0 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 16)) (0.17.1)
Requirement already satisfied: tqdm>=4.64.0 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 17)) (4.66.2)
Requirement already satisfied: ultralytics>=8.0.232 in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (from -r requirem
e 18)) (8.2.28)
Requirement already satisfied: pandas>=1.1.4 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 27)) (2.2.1)
Requirement already satisfied: seaborn>=0.11.0 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 28)) (0.13.2)
Requirement already satisfied: setuptools>=65.5.1 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 42)) (69.2.0)
Requirement already satisfied: wheel>=0.38.0 in /opt/conda/lib/python3.11/site-packages (from -r requirements.txt (line 50)) (0.43.0)
Requirement already satisfied: arrow in /opt/conda/lib/python3.11/site-packages (from codecarbon) (1.3.0)
Requirement already satisfied: click in /opt/conda/lib/python3.11/site-packages (from codecarbon) (8.1.7)
Requirement already satisfied: prometheus-client in /opt/conda/lib/python3.11/site-packages (from codecarbon) (0.20.0)
Requirement already satisfied: py-cpuinfo in /opt/conda/lib/python3.11/site-packages (from codecarbon) (9.0.0)
Requirement already satisfied: pynvml in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (from codecarbon) (11.5.0)
Requirement already satisfied: rapidfuzz in /home/nimisha.vilayatarani002/.local/lib/python3.11/site-packages (from codecarbon) (3.9.3)
Requirement already satisfied: gitdb<5,>=4.0.1 in /opt/conda/lib/python3.11/site-packages (from gitpython>=3.1.30->-r requirements.txt (li
```

Step:6 To install all dependencies for running it separately on the Anaconda environment.

```
1 mamba activate
```

```
Last executed at 2024-06-20 11:04:28 in 778ms
```

```
Run 'mamba init' to be able to run mamba activate/deactivate
and start a new shell session. Or use conda to activate/deactivate.


Note: you may need to restart the kernel to use updated packages.
```

```
mamba create -y -n "diffproml" python=3.12
```

```
[+] 0.0s
[+] 0.1s
conda-forge/linux-64 ──────╌╌╌╌╌╌╌╌╌╌╌╌╌╌   0.0 B /  ??.?MB @  ??.?MB/s  0.1s
conda-forge/noarch   ──────╌╌╌╌╌╌╌╌╌╌╌╌╌╌   0.0 B /  ??.?MB @  ??.?MB/s  0.1s[+] 0.2s
conda-forge/linux-64 ──────────────────────  797.1kB /  35.5MB @   4.4MB/s  0.2s
conda-forge/noarch   ──────────────────────  216.4kB /  15.1MB @   1.2MB/s  0.2s[+] 0.3s
conda-forge/linux-64 ──╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌   4.4MB /  35.5MB @  15.5MB/s  0.3s
conda-forge/noarch   ──╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌   2.5MB /  15.1MB @   8.8MB/s  0.3s[+] 0.4s
conda-forge/linux-64 ────╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌╌   9.5MB /  35.5MB @  24.4MB/s  0.4s
conda-forge/noarch   ──────────╌╌╌╌╌╌╌╌╌╌   7.5MB /  15.1MB @  19.5MB/s  0.4s[+] 0.5s
conda-forge/linux-64 ────────╌╌╌╌╌╌╌╌╌╌╌╌  14.3MB /  35.5MB @  29.3MB/s  0.5s
conda-forge/noarch   ──────────────────╌╌╌╌  12.4MB /  15.1MB @  25.7MB/s  0.5s[+] 0.6s
conda-forge/linux-64 ────────╌╌╌╌╌╌╌╌╌╌╌╌  14.3MB /  35.5MB @  29.3MB/s  0.6s
conda-forge/noarch   ──────────────────────  15.1MB /  15.1MB @  28.1MB/s  0.6s[+] 0.7s
conda-forge/linux-64 ────────╌╌╌╌╌╌╌╌╌╌╌╌  14.3MB /  35.5MB @  29.3MB/s  0.7s
conda-forge/noarch   ──────────────────────  15.1MB /  15.1MB @  28.1MB/s  0.7s[+] 0.8s
conda-forge/linux-64 ────────╌╌╌╌╌╌╌╌╌╌╌╌  14.3MB /  35.5MB @  29.3MB/s  0.8s
conda-forge/noarch   ──────────────────────  15.1MB /  15.1MB @  28.1MB/s  0.8s[+] 0.9s
conda-forge/linux-64 ────────╌╌╌╌╌╌╌╌╌╌╌╌  14.3MB /  35.5MB @  29.3MB/s  0.9s
conda-forge/noarch   ──────────────────────  15.1MB /  15.1MB @  28.1MB/s  0.9s[+] 1.0s
conda-forge/linux-64 ────────╌╌╌╌╌╌╌╌╌╌╌╌  14.3MB @  29.3MB/s            1.0s
conda-forge/noarch   ──────────────────────  15.1MB @  28.1MB/s Downloaded  1.0sconda-forge/noarch                          @  28.1MB/s  1.0s
[+] 1.1s
conda-forge/linux-64 ──────────╌╌╌╌╌╌╌╌╌╌  21.7MB /  35.5MB @  19.9MB/s  1.1s[+] 1.2s
conda-forge/linux-64 ──────────────────╌╌╌  31.0MB /  35.5MB @  26.0MB/s  1.2s[+] 1.3s
conda-forge/linux-64 ──────────────────╌╌╌  31.0MB /  35.5MB @  26.0MB/s  1.3s[+] 1.4s
conda-forge/linux-64 ──────────────────╌╌╌  31.0MB /  35.5MB @  26.0MB/s  1.4s[+] 1.5s
conda-forge/linux-64 ──────────────────╌╌╌  31.0MB /  35.5MB @  26.0MB/s  1.5s[+] 1.6s
conda-forge/linux-64 ──────────────────────  35.5MB @  28.6MB/s Downloaded  1.6sconda-forge/linux-64                        @  28.6MB/s  1.6s
Transaction

  Prefix: /home/nimisha.vilayatarani002/.conda/envs/diffproml
```

```
1 mamba activate "diffproml"
```

```
Last executed at 2024-06-20 12:49:45 in 762ms
```

```
Run 'mamba init' to be able to run mamba activate/deactivate
and start a new shell session. Or use conda to activate/deactivate.


Note: you may need to restart the kernel to use updated packages.
```

Step 7 :- YOLOv5 Model Training

With !python train.py the training script of the YOLOv5 framework can be called. Here are some parameters to define.

- --img: Set the image size
- --batch: batch size (please do not change, because GPU resources are shared between users)
- --epochs: number of training epochs
- --data: path to the YAML configuration file of the dataset
- --weights: path to pre-trained model weights (see picture for different model sizes)
- --hyp: path to YAML configuration file with additional training hyperparameters

`input code for training the yolo5

```
from codecarbon import EmissionsTracker

YOLO_DATASET_ROOT = "/raid/jupyterhub/datasets/boneage_yolo/" # do not change this path as it contains the Yolo pre-processed dataset

tracker = EmissionsTracker(measure_power_secs=9999)
tracker.start()

try:
    %cd yolov5
    !python train.py --img 640 --batch 24 --epochs 5 --data $YOLO_DATASET_ROOT/boneage_yolo.yaml --weights yolov5s.pt
finally:
    emissions = tracker.stop()

print(f"~{emissions * 1_000}g CO2 produced")
```

st executed at 2024-04-12 13:40:50 in 2m 33.14s

Output for different Epoch values as result of changing the number of epochs so as to provide better training to the model.

```
Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 0/19      5.37G     0.1265       0.13    0.05689        650        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513   0.000615     0.0142   0.000328    7.9e-05

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 1/19      6.06G     0.1137     0.1513      0.054        507        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513     0.0222     0.0552     0.0114     0.0023

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 2/19      6.06G      0.102     0.1465    0.04743        620        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.234      0.135     0.0259     0.0053

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 3/19      6.06G    0.09365     0.1376    0.04069        751        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.553      0.165     0.0576     0.0118

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 4/19      6.06G       0.086     0.1351    0.03631        529        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.447      0.229      0.129     0.0294

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 5/19      6.06G    0.08162      0.134    0.03342        607        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.287      0.371      0.113     0.0245

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 6/19      6.06G    0.07592     0.1371    0.03036        594        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.375       0.59       0.32     0.0891

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 7/19      6.06G    0.07658     0.1295    0.02791        658        640: 1
           Class     Images  Instances          P          R      mAP50
```

```
Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 8/19      6.06G     0.0732     0.1337    0.02511        768        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.352      0.618      0.287     0.0857

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
 9/19      6.06G     0.0711     0.1286     0.0234        643        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.305      0.487      0.147     0.0345

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
10/19      6.06G    0.07214     0.1246     0.0222        663        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.471        0.6      0.452      0.137
```

```
Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
11/19      6.06G    0.06859      0.122    0.02087        559        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.323      0.605      0.375       0.11

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
12/19      6.06G    0.06363     0.1211    0.01993        686        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513       0.56       0.72      0.654       0.23

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
13/19      6.06G    0.06067     0.1211    0.01837        484        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.505      0.492      0.401       0.117

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
14/19      6.06G    0.05809     0.1239    0.01753        540        640: 1
           Class     Images  Instances          P          R      mAP50

16/19      6.06G    0.05079     0.1213    0.01578        684        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.722      0.692      0.706       0.226

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
17/19      6.06G     0.0503     0.1274    0.01577        679        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513       0.77      0.662      0.664       0.189

Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
18/19      6.06G    0.04796     0.1242    0.01552        622        640: 1
           Class     Images  Instances          P          R      mAP50
             all         89       1513      0.775      0.771      0.855       0.378
```

Step 8 :- Model results

Here as with xray image we make the BB-Boxes from the xray image we train the model for providing the best we have then store the output yolo5 format which is csv the dataset which is been provided to us is .

- distal interphalangeal joints (DIP, below in green).
- proximal interphalangeal joints (PIP, below in turquoise)

- metacarpophalangeal joints (MCP, below in pale green)
- wrist
- ulna
- radius

```
Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50
                   all         89       1513      0.878      0.772      0.881      0.377
                   DIP         89        356      0.783      0.933      0.898      0.313
                   PIP         89        445      0.845      0.939      0.918      0.322
                   MCP         89        445      0.819      0.957      0.942      0.434
                Radius         89         89      0.974      0.826      0.963      0.524
                  Ulna         89         89          1          0      0.589      0.208
                 Wrist         89         89      0.848      0.978      0.975      0.463
Results saved to runs/train/exp
```

Step 9:- Loading some images of the validation dataset. The N parameter can be used to specify the number of validation images.

```
model_weights = "~/yolov5/runs/train/exp/weights/best.pt" # set path to trained model weights
YOLO_DATASET_ROOT = "/raid/jupyterhub/datasets/boneage_yolo/" # do not change this path as it contains the Yolo pre-processed dataset

!python val.py --img 640 --batch 24 --iou-thres 0.6 --data $YOLO_DATASET_ROOT/boneage_yolo.yaml --weights $model_weights
```

st executed at 2024-06-20 13:05:45 in 32.97s

Step 10:- visualization we will able to see boxes on the x-ray images which would work as input for our regression model further analysis now data we will use this for further analysis and finding the best model for finding the bone age of a child to know by seeing the bone age whether the child is healthy or not is that bone grows as per the age.

```
val: data=/raid/jupyterhub/datasets/boneage_yolo/boneage_yolo.yaml, weights=['/home/nimisha.vijayaraman1002/yolov5/runs/train/exp/weights/best.pt'], b
atch_size=24, imgsz=640, conf_thres=0.001, iou_thres=0.6, max_det=300, task=val, device=, workers=8, single_cls=False, augment=False, verbose=False, sa
ve_txt=False, save_hybrid=False, save_conf=False, save_json=False, project=runs/val, name=exp, exist_ok=False, half=False, dnn=False
YOLOv5 🚀 v7.0-318-gc0380fd8 Python-3.11.8 torch-2.2.1+cu121 CPU

Fusing layers...
Model summary: 157 layers, 7026307 parameters, 0 gradients, 15.8 GFLOPs
val: Scanning /raid/jupyterhub/datasets/boneage_yolo/labels/valid... 89 images,
val: WARNING ⚠ Cache directory /raid/jupyterhub/datasets/boneage_yolo/labels is not writeable: [Errno 13] Permission denied: '/raid/jupyterhub/datase
ts/boneage_yolo/labels/valid.cache.npy'
                 Class     Images  Instances          P          R      mAP50
                   all         89       1513      0.876      0.775      0.878      0.374
                   DIP         89        356      0.781      0.927       0.89      0.316
                   PIP         89        445      0.817      0.939      0.904       0.32
                   MCP         89        445      0.831      0.962      0.949      0.442
                Radius         89         89      0.963      0.843      0.965      0.509
                  Ulna         89         89          1          0      0.592       0.19
                 Wrist         89         89      0.864      0.978      0.971      0.468
Speed: 0.8ms pre-process, 239.9ms inference, 20.9ms NMS per image at shape (24, 3, 640, 640)
Results saved to runs/val/exp
```

```
model_weights = f"{os.path.expanduser('~')}/yolov5/runs/train/exp/weights/best.pt" # set path to trained model weights
model = torch.hub.load('ultralytics/yolov5', 'custom', path=model_weights) # load model
```

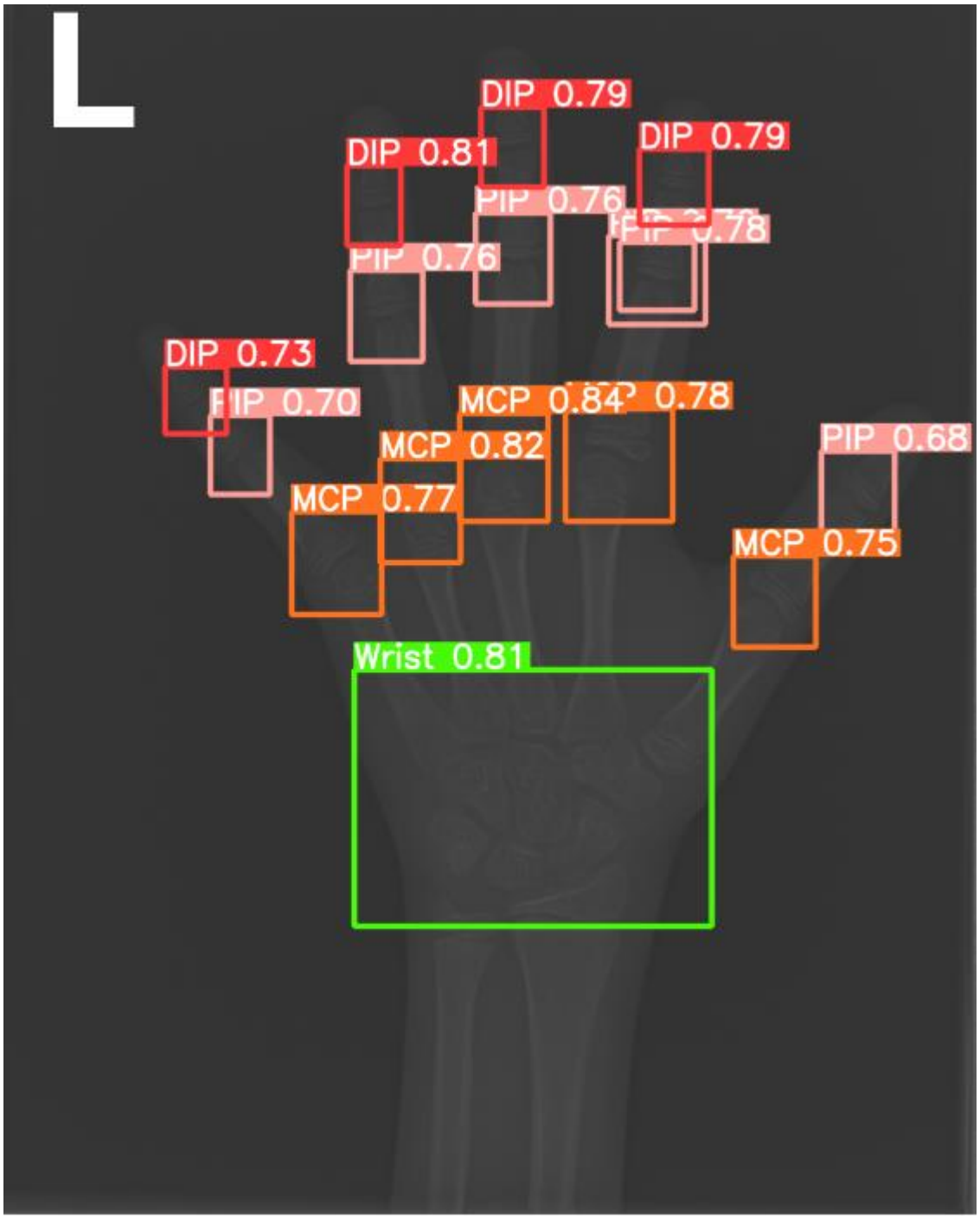
Last executed at 2024-04-12 13:41:24 in 4.07s

```
# Load list of images of image file names
def load_images(image_files: list):
    return [Image.open(file).convert("RGB") for file in image_files]

N = 1 # number of visualized images
YOLO_DATASET_ROOT = "/raid/jupyterhub/datasets/boneage_yolo/" # do not change this path as it contains the Yolo pre-processed dataset
IMAGES = YOLO_DATASET_ROOT + "images/valid/" # path to validation image files
img_files = sorted(glob.glob(IMAGES + '*.png'))[:N] # get fist N image filenames
imgs = load_images(img_files) # load images to list
```

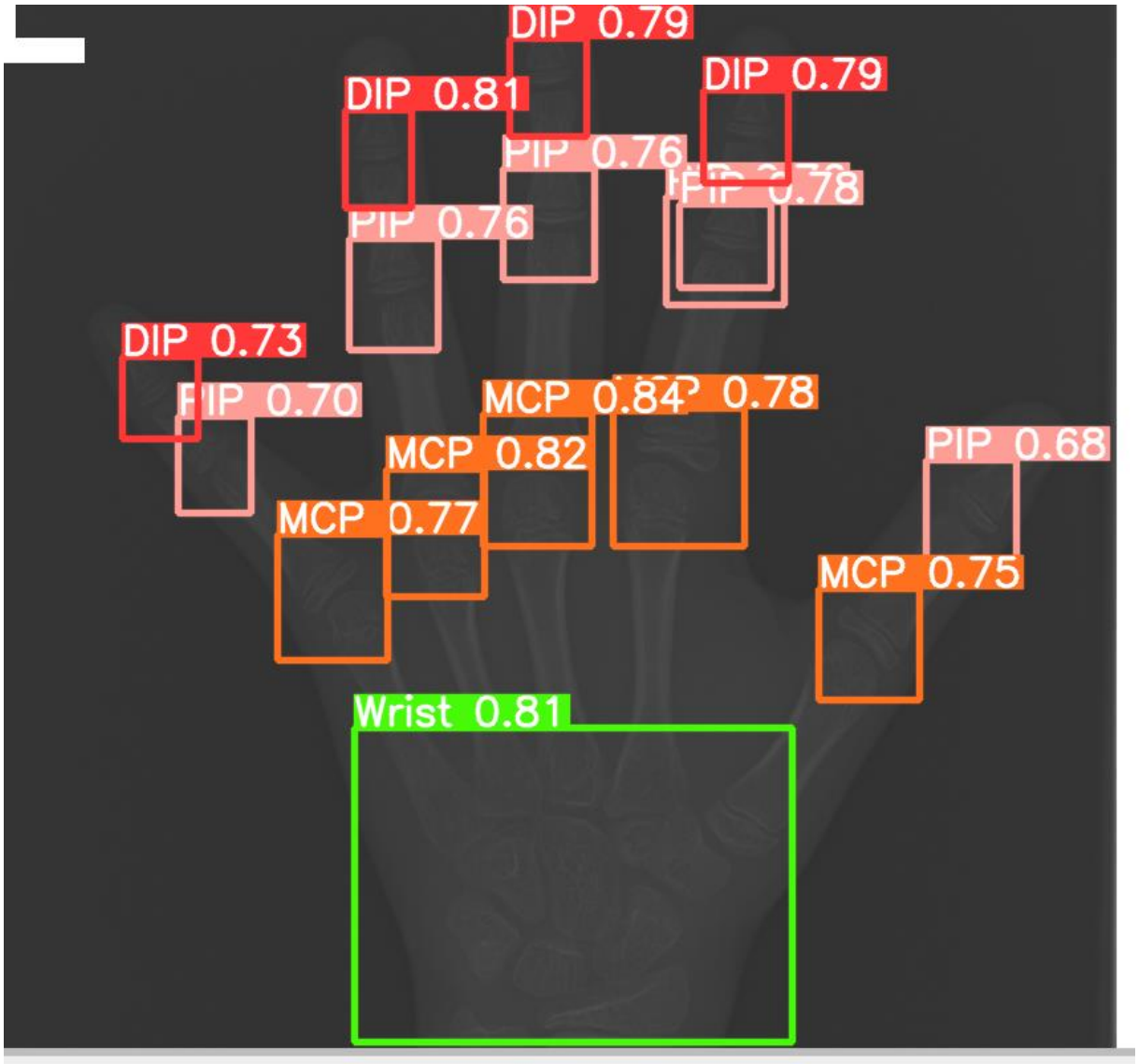
```
# Inference settings
model.conf = 0.6 # min. confidence of 0.6
model.iou = 0.6 # min IoU of 0.6 for none-max-suppression (filtering of overlapping bboxes)

# iterate over images
for i, img in enumerate(imgs):
    result = model(img, size=640) # nms inference for single image
    result.print() # print inference results
    with pd.option_context('display.float_format', '{:0.3f}'.format): # print coordinates of bboxes
        print(result.pandas().xyxy[0])
    result.show() # show inference results

    #result.save(ROOT_DIR + "/BoneAgeInference"+ str(i) + ".png")
    result.save()
```

t executed at 2024-06-20 13:49:18 in 362ms

	xmin	ymin	xmax	ymax	confidence	class	name
0	427.386	459.165	509.927	559.775	0.842	2	MCP
1	354.464	501.242	428.642	597.314	0.817	2	MCP
2	323.231	230.565	373.683	304.950	0.813	0	DIP
3	330.944	696.979	660.205	933.937	0.810	5	Wrist
4	593.208	215.340	657.175	285.869	0.791	0	DIP
5	447.977	176.479	505.782	250.100	0.789	0	DIP
6	575.909	301.817	644.274	364.662	0.782	1	PIP
7	525.188	455.565	624.297	559.872	0.781	2	MCP
8	272.357	550.660	355.017	645.394	0.766	2	MCP
9	326.507	327.509	393.718	411.823	0.765	1	PIP
10	442.942	274.170	511.621	358.321	0.764	1	PIP
11	565.154	295.862	654.899	377.129	0.757	1	PIP
12	680.675	591.981	756.533	675.777	0.751	2	MCP
13	155.606	416.660	212.441	478.169	0.732	0	DIP
14	197.834	461.488	252.802	534.445	0.699	1	PIP
15	761.933	494.351	829.999	587.108	0.685	1	PIP



From the x-ray, we were successfully able to draw BB-boxes and convert our dataset into the yolo5 framework.



Step 11:- perform the four different models such as Linear Regression, Decision Tree Regressor, Random Forest Regressor, and Support Vector Regressor. And then we tried to find out which is the best model based on Mean Squared Error, R squared.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score

# Load the datasets
train_df = pd.read_csv('/raid/datashare/RSNA2017/boneage_yolo/train.csv')
valid_df = pd.read_csv('/raid/datashare/RSNA2017/boneage_yolo/valid.csv')

# Combine train and validation datasets to ensure consistent encoding
combined_df = pd.concat([train_df, valid_df])

# Encode the class labels
le = LabelEncoder()
combined_df['class'] = le.fit_transform(combined_df['class'])

# Split back into train and validation sets
train_df = combined_df.iloc[:len(train_df)]
valid_df = combined_df.iloc[len(train_df):]

# Define features (X) and target (y)
X_train = train_df.drop(['filename', 'xmin', 'ymin', 'xmax', 'ymax'], axis=1)
y_train = train_df[['xmin', 'ymin', 'xmax', 'ymax']].values  # Ensure y_train is 2D
X_valid = valid_df.drop(['filename', 'xmin', 'ymin', 'xmax', 'ymax'], axis=1)
y_valid = valid_df[['xmin', 'ymin', 'xmax', 'ymax']].values  # Ensure y_valid is 2D

# Initialize the models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree Regressor': DecisionTreeRegressor(),
    'Random Forest Regressor': RandomForestRegressor(),
    'Support Vector Regressor': SVR()
}

# Train and evaluate the models
results = {}
for name, model in models.items():
    if name == 'Support Vector Regressor':
        svr_models = {}
        y_preds = np.zeros_like(y_valid)

        for i, target in enumerate(['xmin', 'ymin', 'xmax', 'ymax']):
            svr = SVR()
            svr.fit(X_train, y_train[:, i])
            y_preds[:, i] = svr.predict(X_valid)
            svr_models[target] = svr

```

```python
# Initialize the models
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree Regressor': DecisionTreeRegressor(),
    'Random Forest Regressor': RandomForestRegressor(),
    'Support Vector Regressor': SVR()
}

# Train and evaluate the models
results = {}
for name, model in models.items():
    if name == 'Support Vector Regressor':
        svr_models = {}
        y_preds = np.zeros_like(y_valid)

        for i, target in enumerate(['xmin', 'ymin', 'xmax', 'ymax']):
            svr = SVR()
            svr.fit(X_train, y_train[:, i])
            y_preds[:, i] = svr.predict(X_valid)
            svr_models[target] = svr

        mse = mean_squared_error(y_valid, y_preds)
        r2 = r2_score(y_valid, y_preds)
        results[name] = {'MSE': mse, 'R2': r2}
        print(f"{name} - Mean Squared Error: {mse}, R-squared: {r2}")
    else:
        model.fit(X_train, y_train)
        y_pred = model.predict(X_valid)

        mse = mean_squared_error(y_valid, y_pred)
        r2 = r2_score(y_valid, y_pred)

        results[name] = {'MSE': mse, 'R2': r2}
        print(f"{name} - Mean Squared Error: {mse}, R-squared: {r2}")

# Display results
print("\nModel Comparison:")
for name, result in results.items():
    print(f"{name} - MSE: {result['MSE']:.4f}, R2: {result['R2']:.4f}")

```

```
ast executed at 2024-06-21 11:08:53 in 4.18s
```

Output

```
Linear Regression - Mean Squared Error: 43692.02219467589, R-squared: 0.6346023719066136
Decision Tree Regressor - Mean Squared Error: 32133.32184838901, R-squared: 0.710198645706994
Random Forest Regressor - Mean Squared Error: 30107.988332416855, R-squared: 0.7263082679545344
Support Vector Regressor - Mean Squared Error: 79530.62640449438, R-squared: 0.37318387047258844

Model Comparison:
Linear Regression - MSE: 43692.0222, R2: 0.6346
Decision Tree Regressor - MSE: 32133.3218, R2: 0.7102
Random Forest Regressor - MSE: 30107.9883, R2: 0.7263
Support Vector Regressor - MSE: 79530.6264, R2: 0.3732
```

Output in tabular form

Model	Mean Squared Error	R-squared
Linear Regression	43692.02	0.6346
Decision Tree Regressor	32345.35	0.7079

Random Forest Regressor	30079.01	0.7262
Support Vector Regressor	46071.58	0.6204

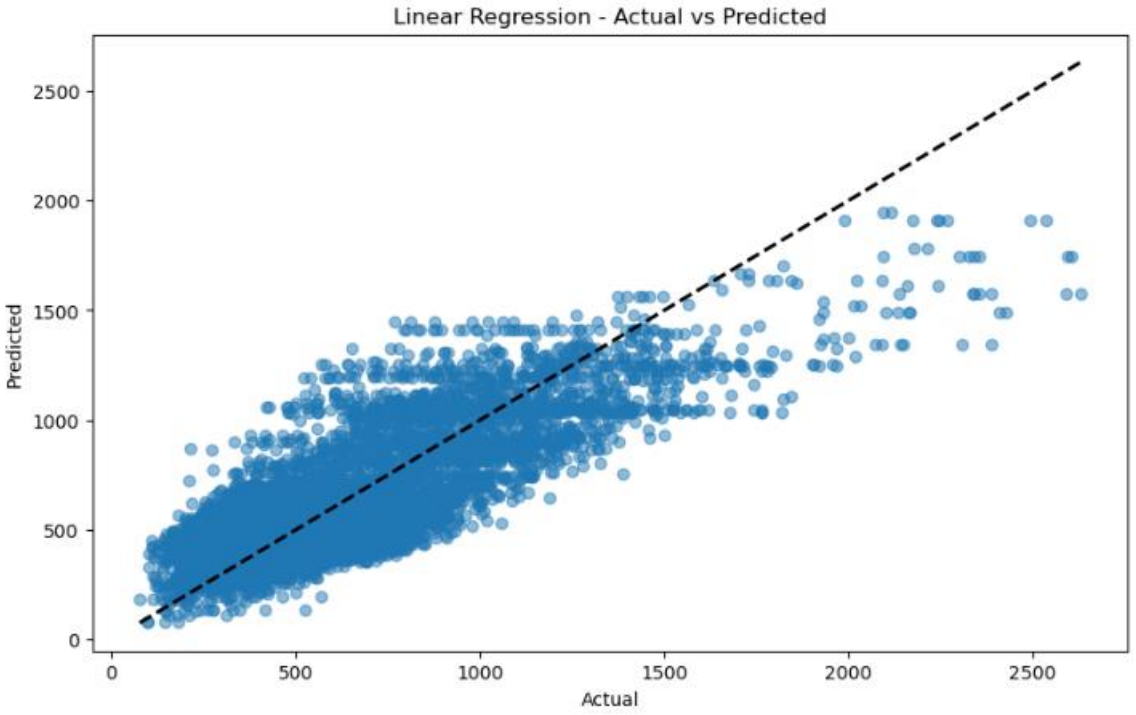
Through this, it is easily understood that data tree regression model provide the result and the reason for this is the minimum value of mean squared error which means that less possible chances to make errors and a high value R squared which is equally good.

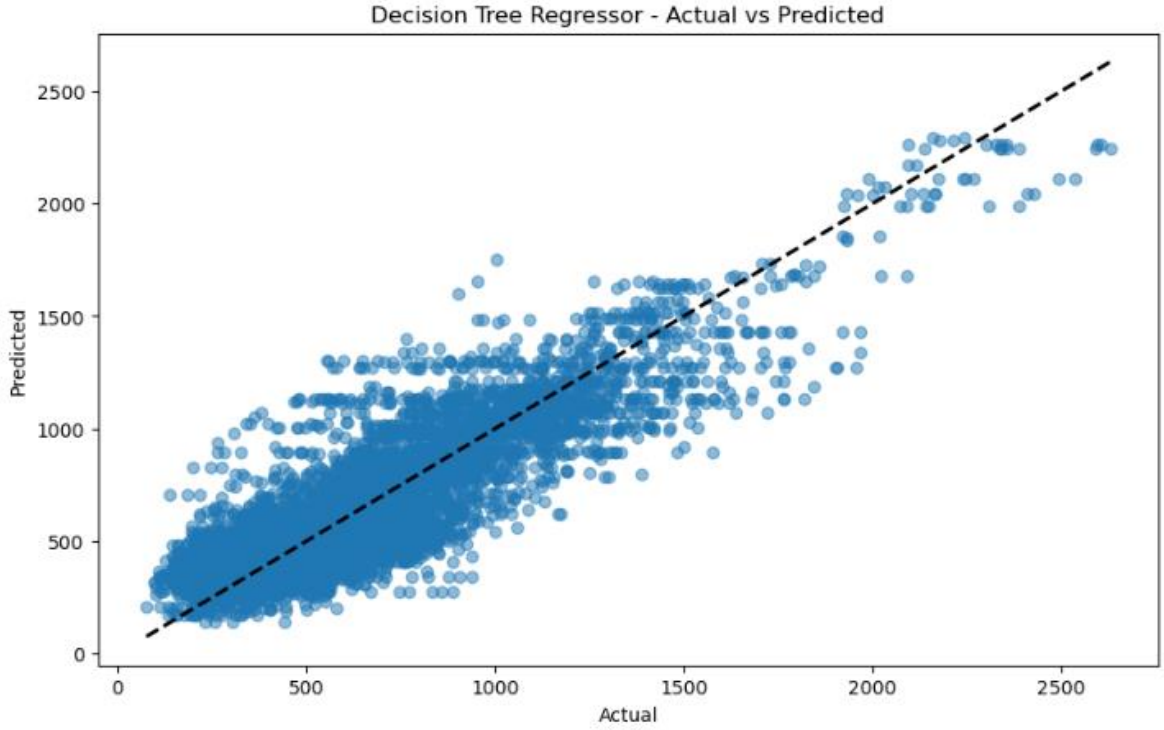
Step 12:- Here is the graphical representation through which we can visualize the result in this we can see that which model performs well on the basis of predicting actual value and final value.

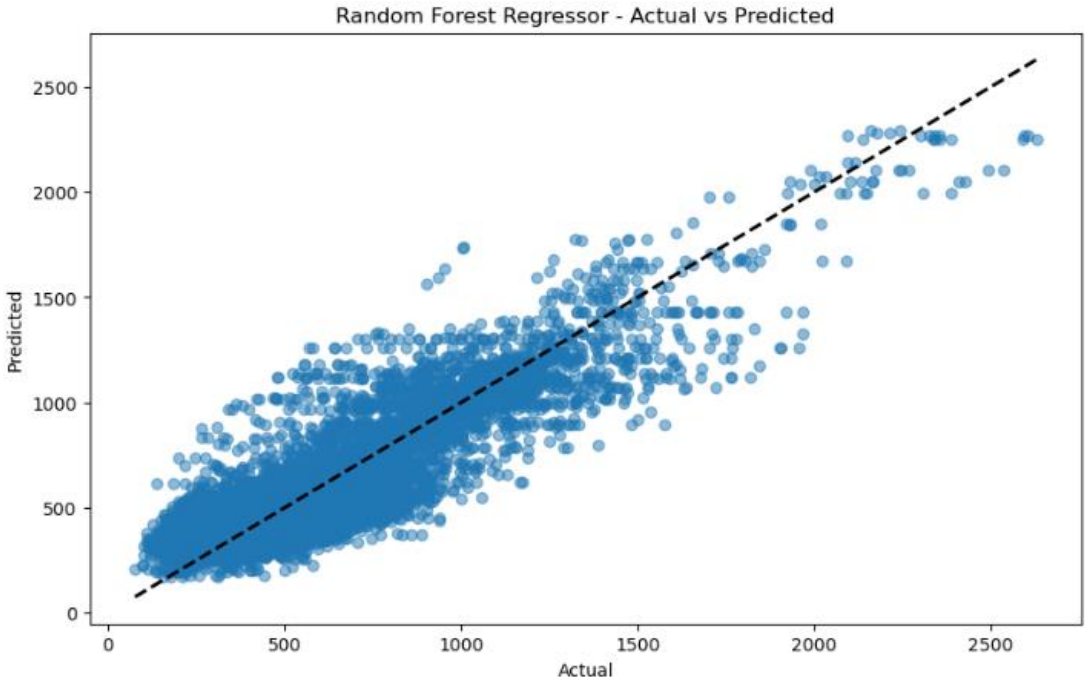
Here is the input code and output graph.

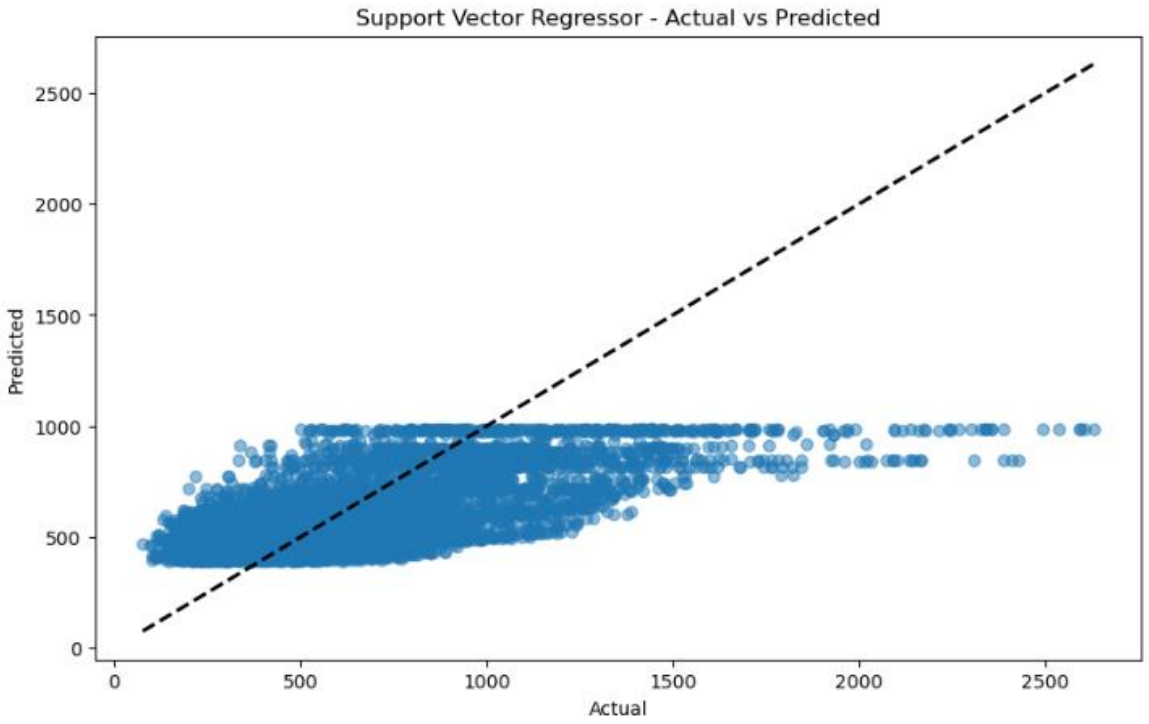
```
import matplotlib.pyplot as plt
import cv2
import random

# Function to visualize actual vs predicted values
def visualize_results(y_valid, y_pred, title):
    plt.figure(figsize=(10, 6))
    plt.scatter(y_valid, y_pred, alpha=0.5)
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(title)
    plt.plot([y_valid.min(), y_valid.max()], [y_valid.min(), y_valid.max()], 'k--', lw=2)
    plt.show()

# Function to plot bounding boxes
def plot_bounding_boxes(image_path, true_boxes, pred_boxes):
    image = cv2.imread(image_path)
    for box in true_boxes:
        cv2.rectangle(image, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (0, 255, 0), 2)  # Green for true boxes
    for box in pred_boxes:
        cv2.rectangle(image, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (0, 0, 255), 2)  # Red for predicted boxes
    plt.figure(figsize=(8, 8))
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.axis('off')  # Hide the axis
    plt.show()

# Example usage
# Visualize results for each model
for name, model in models.items():
    if name == 'Support Vector Regressor':
        y_preds = np.zeros_like(y_valid)
        for i, target in enumerate(['xmin', 'ymin', 'xmax', 'ymax']):
            y_preds[:, i] = svr_models[target].predict(X_valid)
        visualize_results(y_valid.flatten(), y_preds.flatten(), f'{name} - Actual vs Predicted')
    else:
        y_pred = model.predict(X_valid)
        visualize_results(y_valid.flatten(), y_pred.flatten(), f'{name} - Actual vs Predicted')

# Plot bounding boxes for a few random samples
for _ in range(3):
    idx = random.randint(0, len(valid_df)-1)

    true_boxes = [valid_df.iloc[idx][['xmin', 'ymin', 'xmax', 'ymax']].values]
    pred_boxes = [models['Random Forest Regressor'].predict([X_valid.iloc[idx]])[0]]


ast executed at 2024-06-21 11:28:29 in 1.90s
```

Last executed at 2024-06-21 11:28:29 in 1.90s









Conclusion:-

In this project, we explored various regression models to predict bone age from X-ray image data. The dataset consisted of annotated bounding boxes around specific bone regions, which were used to derive features for training and evaluation.

Four regression models were evaluated:

- **Linear Regression**
- **Decision Tree Regressor**
- **Random Forest Regressor**
- **Support Vector Regressor**

Each model was trained and evaluated using metrics such as Mean Squared Error (MSE) and R-squared (R2) to assess their predictive performance. Based on the results, the **Random Forest Regressor** emerged as the top-performing model, demonstrating the lowest MSE and highest R-squared among all models tested. This indicates that the Random Forest model not only minimized prediction errors but also explained a significant portion of the variance in bone age predictions.

The success of the Random Forest Regressor can be attributed to its ability to handle complex relationships and interactions in the data, which is crucial in medical image analysis where non-linear patterns may exist. Additionally, Random Forests are robust against overfitting and are known for their generalization capabilities.

Moving forward, further optimization of the Random Forest model through hyperparameter tuning and potentially exploring ensemble methods could enhance its performance even more. Moreover, deploying the model in a real-world setting would involve considerations such as scalability, interpretability, and integration with existing medical imaging systems.

Overall, this project highlights the effectiveness of machine learning techniques in predicting bone age from X-ray images, paving the way for enhanced diagnostic tools and clinical decision support systems in radiology.