**[Data\_Preprocessing.ipynb](https://github.com/DeveloperVivek9/Calorie-Tracker/blob/main/Deployment/Data_Preprocessing.ipynb)**

**Packages required**

import cv2

from detecto.utils import read\_image

from glob import glob

from google.colab import drive

import io

import json

import os

import pandas as pd

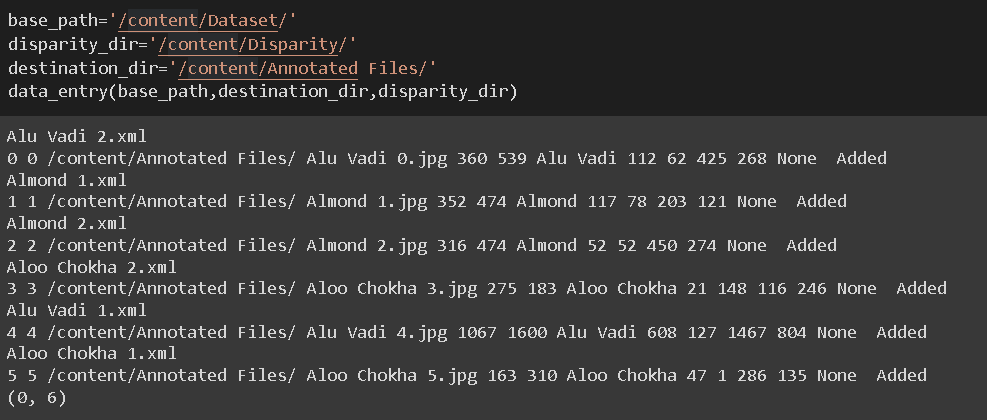
import re

import shutil

#from xml.etree import ElementTree

import xml.etree.cElementTree as ET

We store user images as well as other unannotated images in a folder let’s say Unannotated folder. When the annotators have annotated those images there, we transfer those images to Annotated folder if they are proper. An image is proper if it is readable and has bounding boxes in the format [xmin,ymin,xmax,ymax] (PASCAL VOC)

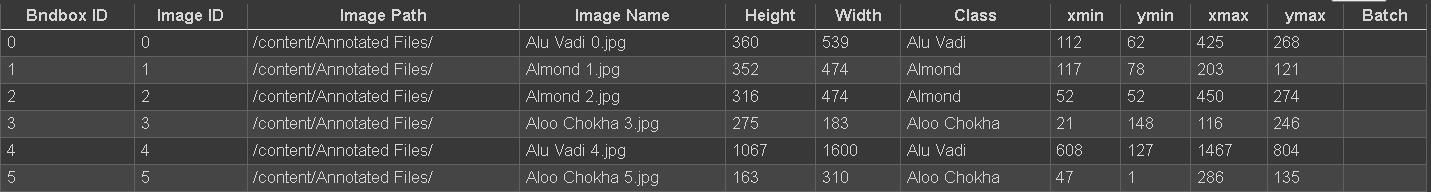


base\_path -> Path of all images [user input or manually uploaded]. We take annotated images from there to:

destination\_dir -> Annotated directory if data is proper. Gets added to Image Metadata.csv as well.

disparity\_dir -> Disparity directory if data is not proper. Gets added to Bad\_Files.csv as well.

We add proper data to the csv file which contains all bounding box information. The csv file is of the format:



Bndbox ID -> ID for bounding box. Unique for every row

Image ID -> ID for image. NOT unique for every row since one image can have multiple bounding boxes

Image Path -> Directory image is in after moving from Unannotated folder to Annotated folder.

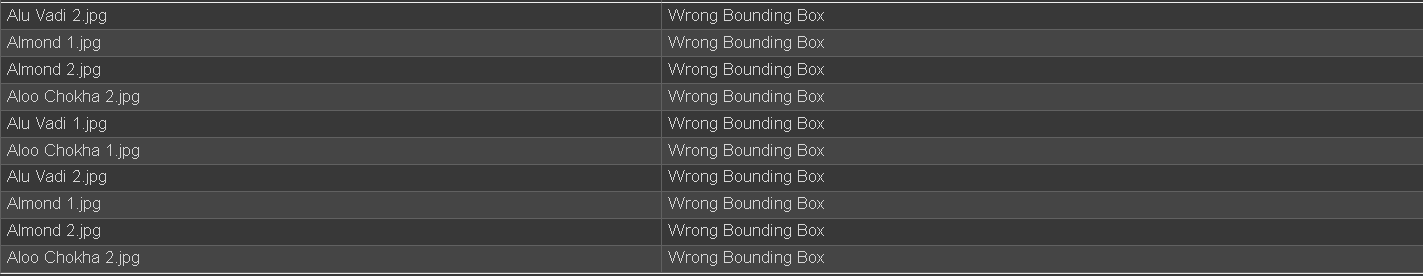
Height -> Height of the image

Width -> Width of the image

xmin, ymin, xmax, ymax -> corresponds to 1 bounding box.

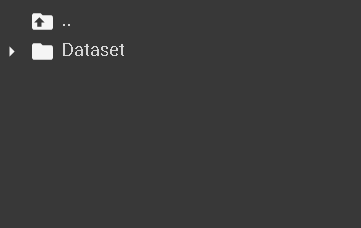
Batch -> Batch in which it was trained. None if not trained.

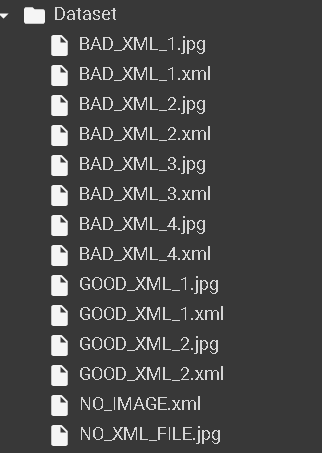
Unreadable images or xml files without images or xml files without proper bounding box gets written to Bad\_files.csv and gets moved to a disparity folder.



**WORKING**

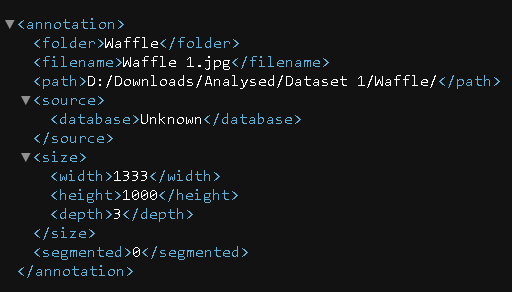
Imagine the following dataset:



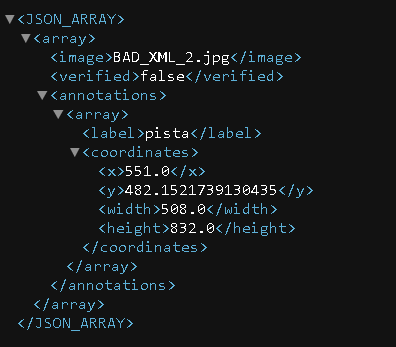


Bad XML files contents are as below :

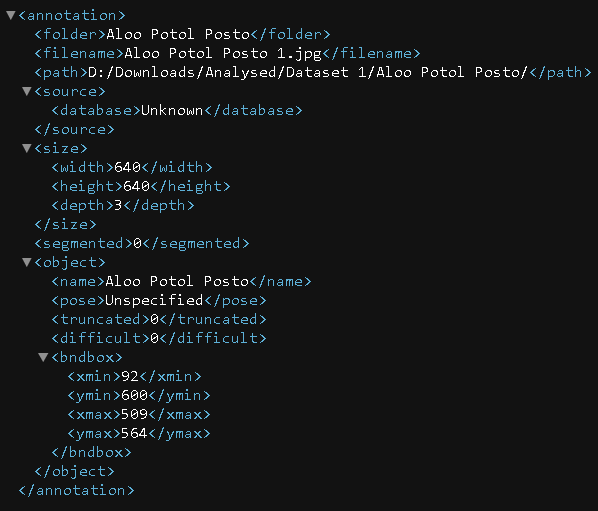
1) No bounding box in this xml file.



2) Bounding box format different from [xmin,ymin,xmax,ymax]



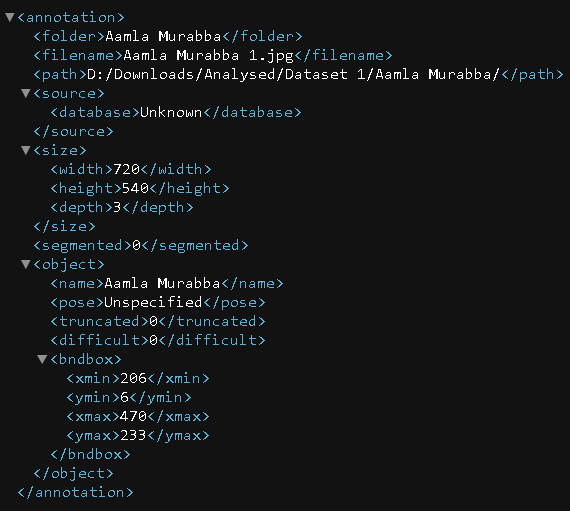
3) Bounding box coordinates wrong. Here ymin > = ymax which should not be possible.



4) Bounding box format different from [xmin,ymin,xmax,ymax]

Good XML File contents are as below:

1) Single Bounding Box



2) Multiple Bounding Box



In the dataset, then there is an xml file without an image, and an image without xml file. Let’s see how we split this dataset.



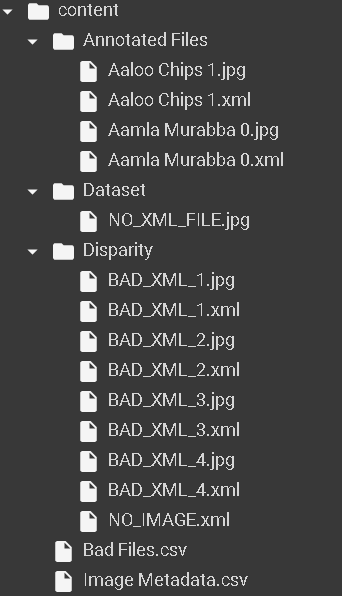
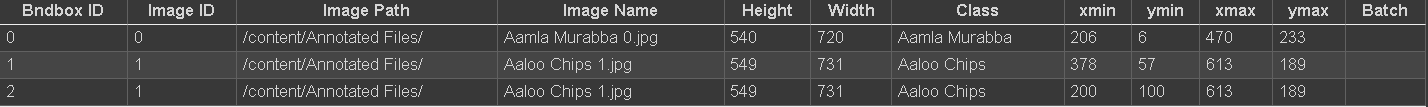
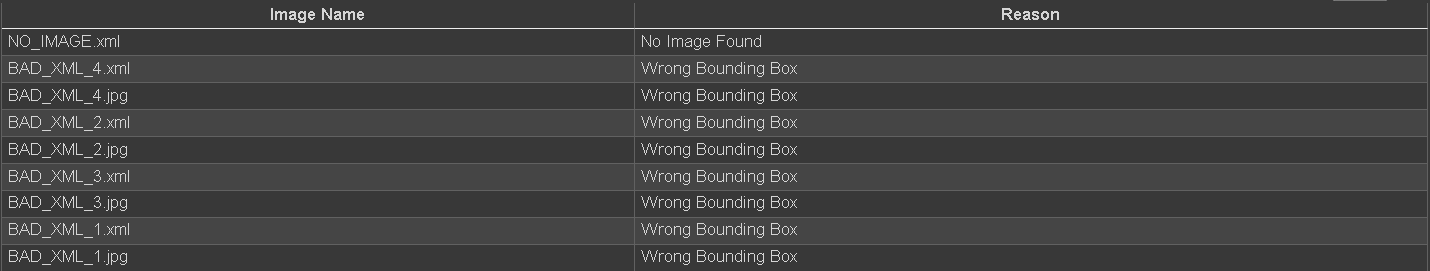


Image Metadata csv



Bad file csv



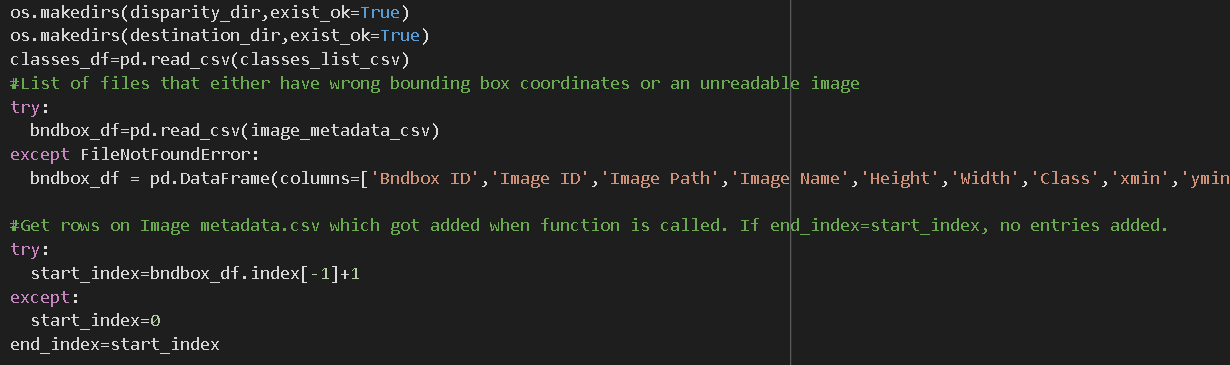
**Explanation**

Bad file csv contains xml and images that we can’t use for the model. Reason why is stated in the second column.

Bounding boxes from the two correct xml files taken and written in csv file. Total of three bounding boxes from 2 xml file -> 3 records where image id is same in two rows since one image contained multiple bounding boxes.

Image without annotation kept in unannotated folder for annotation.

**Code Explanation**

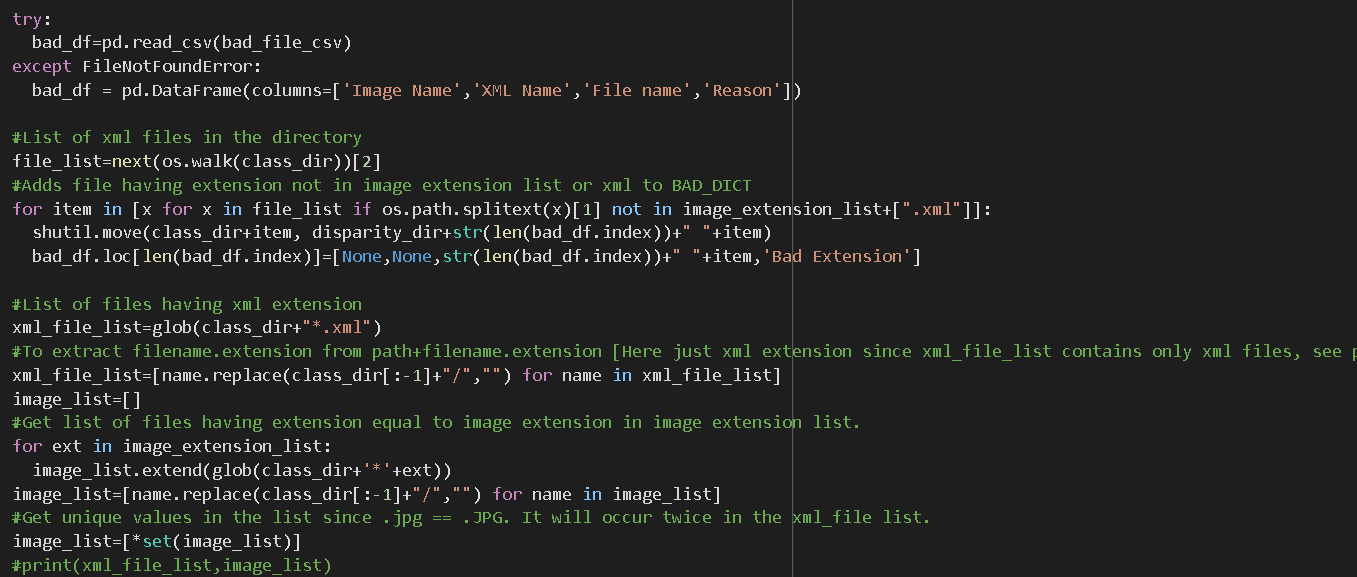
****

Here we are making disparity and destination directory if they don’t exist.

We read classes\_list.csv which contains class names.

If image metadata already exists, we read else we create the dataframe.

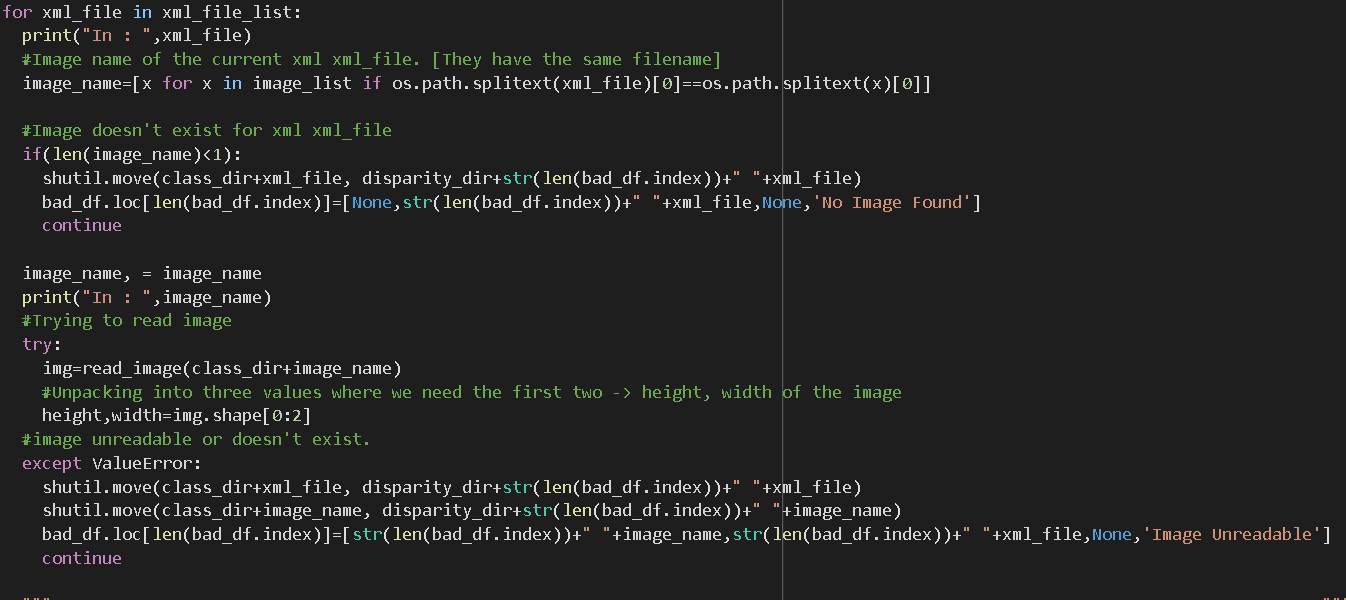
Start\_index, end\_index to account for how many records are added.



Bad file csv read else created.

We go through the file list and remove files with extensions that we don’t accept.

We get the xml file list as well as the image list

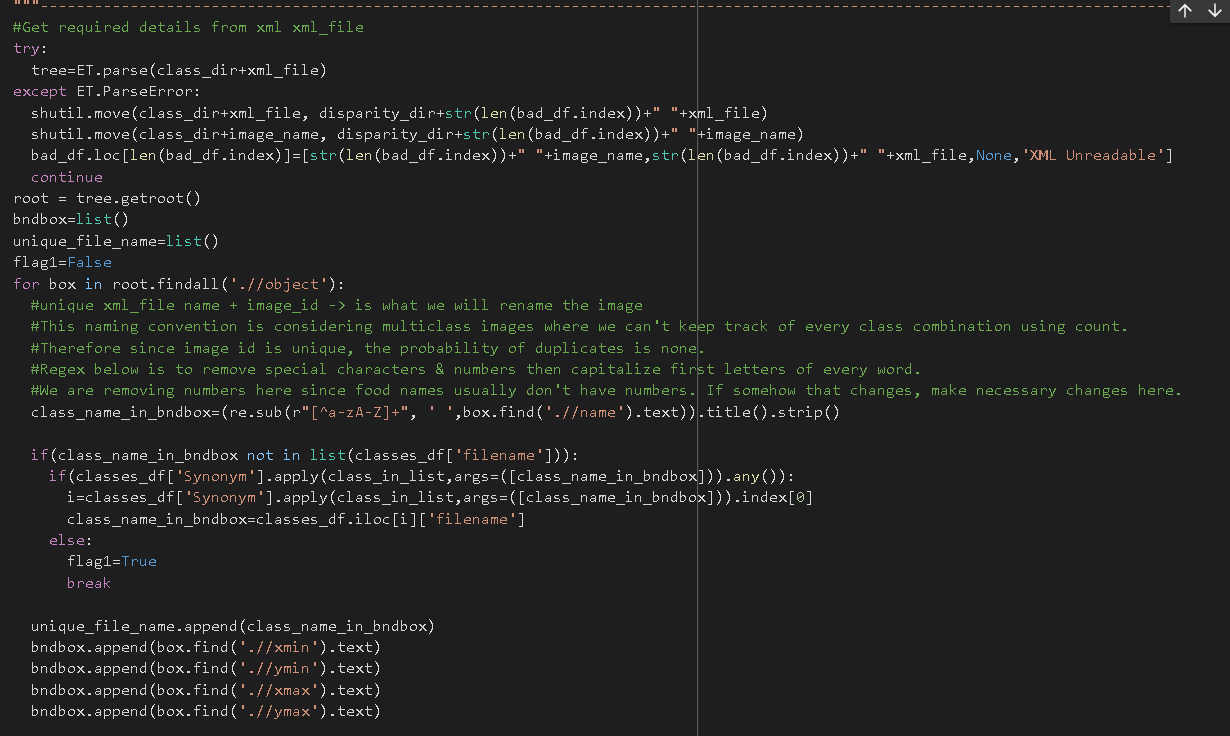


We go through the xml files, get the image name for that particular xml file.

If image doesn’t exist, we remove the xml file.

If the image doesn’t get read, we remove both the image, xml file.

Here ValueError is the exception that is raised when the image doesn’t get read.



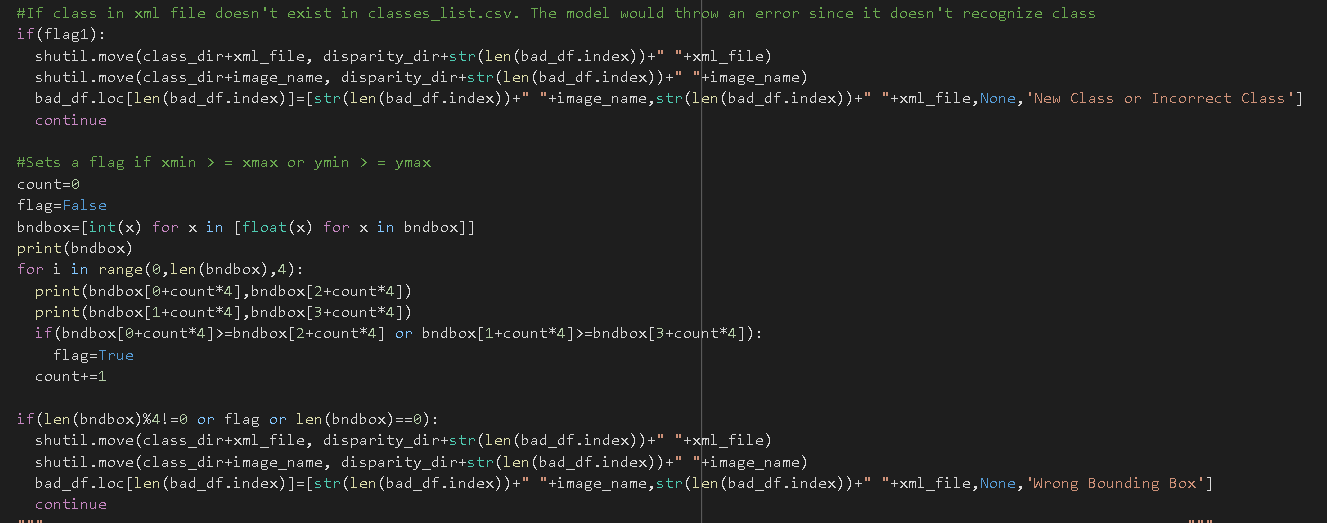
We read the xml file contents. If we can’t, we remove it as well.

Root.findall(“//object”) is the list of every bounding box in the image

We traverse through every bounding box, rename the class in it by removing special characters and numbers.

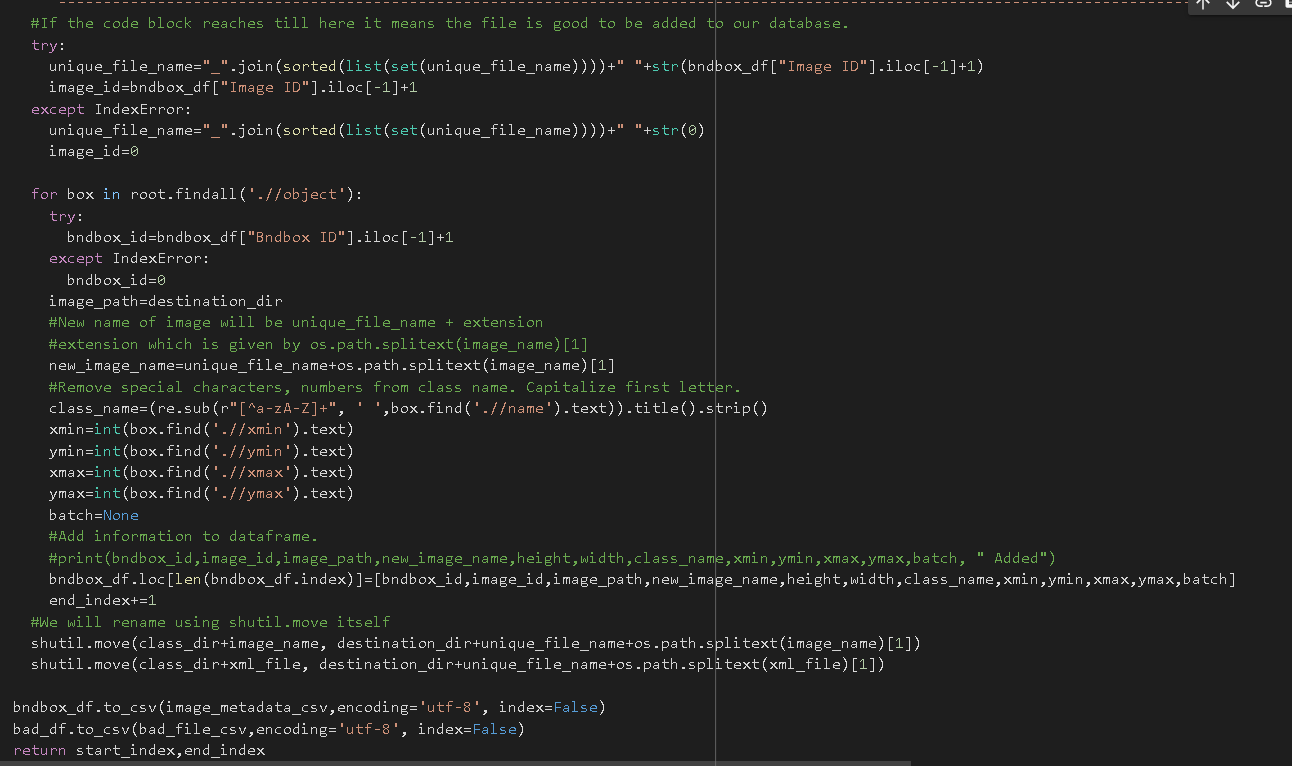
If the class isn’t in classes list, it’s a new class which will cause problems for the model. We set a flag for this.

We add bounding box coordinates to bndbox



If flag is true, we move it to disparity folder.

We check for bounding box coordinates next. If xmin>xmax or ymin>ymax, we remove those images along with images without bounding boxes or with an incorrect number of coordinates.



Unique file name is a combination of class names found in the image. Imagine the image has Chicken Curry and Orange Juice in it. The unique file name will be Chicken Curry\_Orange Juice image id.jpg where image id is dependent on the csv file.

Then we add all extracted details onto the csv file.