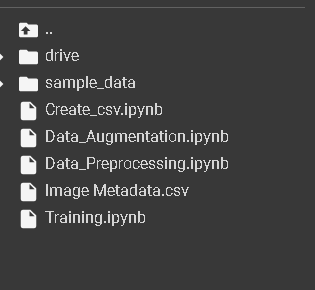
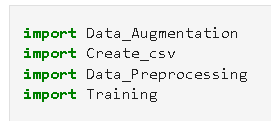
[**Pipeline**](https://github.com/DeveloperVivek9/Calorie-Tracker/blob/main/Deployment/Pipeline.ipynb)

****

Copy the required ipynb files to the current directory. The ipynb files mentioned here are the same ones in the deployment folder in the GitHub repository.

We then import the ipynb files.



STEP 1: [Data\_Preprocessing](https://github.com/DeveloperVivek9/Calorie-Tracker/blob/main/Deployment/Data_Preprocessing.ipynb)

We run data entry function of data pre-processing which creates the image metadata csv file. [Read Preprocessing docx before moving to next step]

STEP 2: [Data\_Augmentation](https://github.com/DeveloperVivek9/Calorie-Tracker/blob/main/Deployment/Data_Augmentation.ipynb)

The function ‘Augmentation\_function’ augments the images in image metadata csv file and saves the images in the annotated folder. [Read Augmentation docx]

STEP 3: [Initial\_Training](https://github.com/DeveloperVivek9/Calorie-Tracker/blob/main/Deployment/Training.ipynb)

We generate train and test csv files for images that has not been trained before. We classify which images have been trained or not by the batch value. If batch value = NULL then it has not been trained before. We then set batch = 0 for the trained images.

STEP 4: [Online\_Training](https://github.com/DeveloperVivek9/Calorie-Tracker/blob/main/Deployment/Training.ipynb)

After the initial training, we set a batch size. If the batch size has been reached, we start training the model again. The batch size is defined by the number in the if statement checking for unique values image id. Here it is 1000.

Again, we take in images that has not been trained, we split them in to train and test then respectively train them. The difference is we have to load the model here since it already exists.

Thus, the entire lifecycle is completed.