**Report**

**Loading Images:**

Load an input image from disk

Determine the image’s width, height, and number of channels

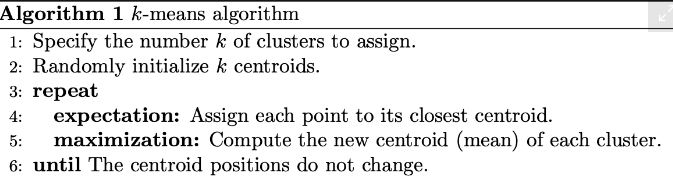
Display the loaded image to our screen , resize it

**K mean Clustering:**

The k-means clustering method is an unsupervised machine learning technique used to identify clusters of data objects in a dataset. There are many different types of clustering methods, but k-means is one of the oldest and most approachable. These traits make implementing k-means clustering in Python reasonably straightforward, even for novice programmers and data scientists.

The first step is to randomly select k centroids, where k is equal to the number of clusters you choose. Centroids are data points representing the center of a cluster.

Main Working of K mean Clustering is:

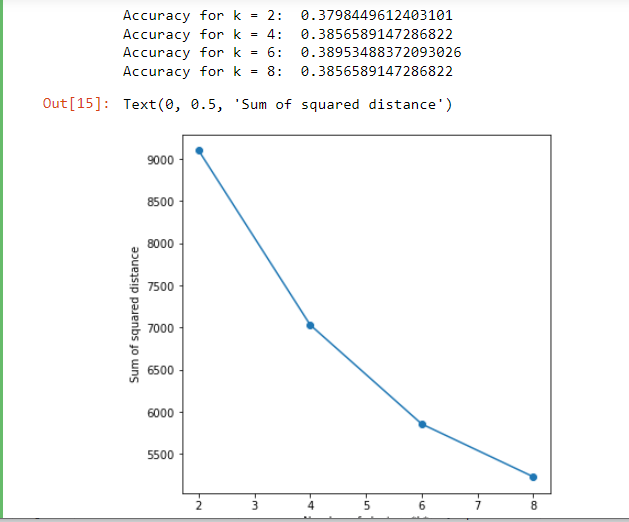


**Elbow Method :**

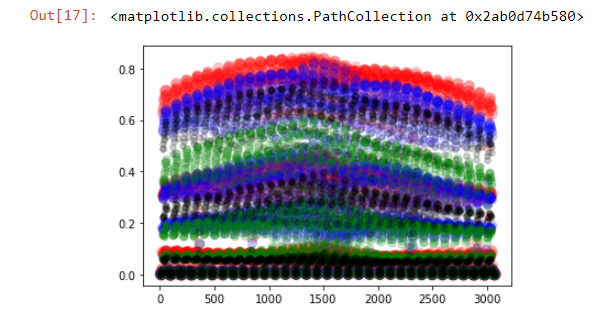
The Elbow Method is one of the most popular methods to determine this optimal value of k.

We now demonstrate the given method using the K-Means clustering technique using the Sklearn library of python.

We get,

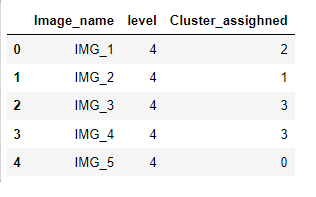


Plot Clusters in different Colors to see there pattern.



As the figure shows, there is mixture of some colors that represent mixing of values in a cluster that's why our accuracy is very low abut 30 percent.

Then Place the clusters against each image in a data set like,



Then ,Deduct 30 percent data from each cluster and create a final prepossessed Data set.

**Images Segmentation:**

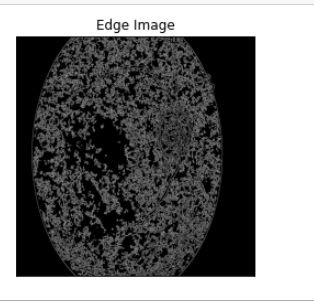
Image segmentation is the process of dividing images to segment based on their characteristic of pixels. It helps us to analyze and understand images more meaningfully.

First we Blur the image , then apply canny edge detector to extract features from the image and next step is to apply K mean Clustering .

When we applying k-means clustering algorithm to an image, it takes each pixel as vector point and building k-clusters of pixels. Let’s go through the Pseudo code algorithm.

1. Choose the number of clusters(K) and obtain pixels
2. Initialize K-means with random pixels
3. Repeat steps 4 and 5 until convergence or until the end of a fixed number of iterations
4. for each pixels Pi:
5. Find the nearest Centroid
6. 2 . Assign the pixels to that cluster
7. for each cluster Ci
8. 1. centroid = mean of all points assigned to that cluster
9. End

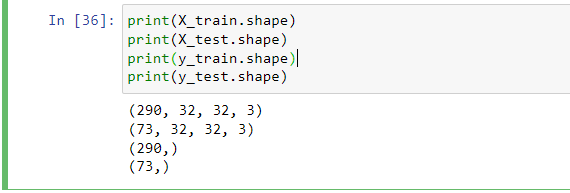
After Segmentation the images like to be:



**Splitting Data Set:**

Split both data sets, after and before , in train and test with random percentage of 10.

The results seems to be:

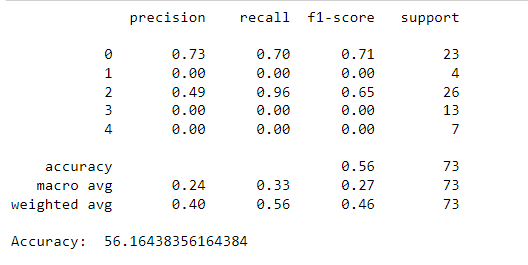


Then use standardScaler to transform Train and Test data set

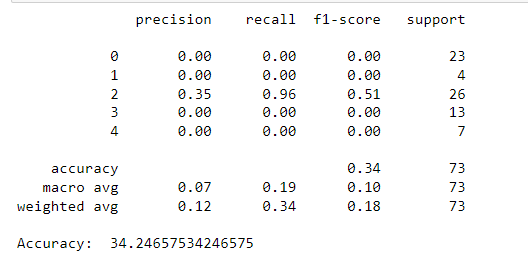
**Apply ML Model:**

# **Stochastic Gradient Descent**

Before Segmentation:

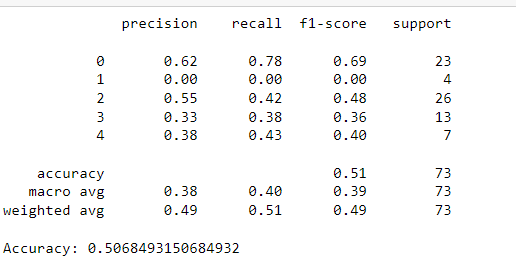


After Segmentation:

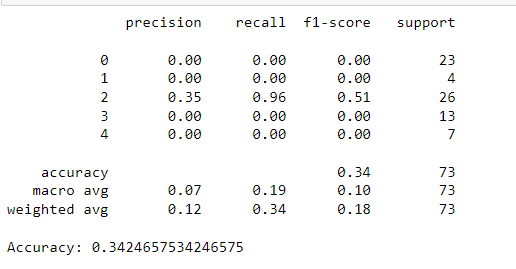


1. **Logistic Regression:**

Before Segmentation:



After Segmentation:



**Apply CNN:**

Load Data set:

PyTorch provides ImageFolder class which makes the task easy for us to prepare the dataset. We simply have to pass the directory of our data to it and it provides the dataset which we can use to train the model.

Then we resize the image for (150\*150) shape and then transforms them into tensors.

So our first image in the dataset has a shape (3,150,150) which means the image has 3 channels (RGB), height 150, and width 150.

so we split the dataset into batches and instead of training the model on whole in a single phase.

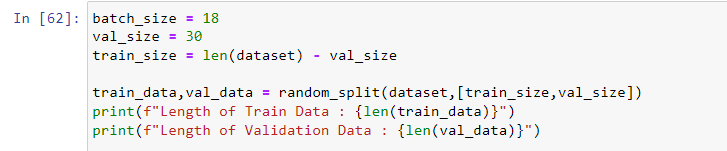
we prepare a base class that extends the functionality of torch.nn.Module (base class used to develop all neural networks. We add various functionalities to the base to train the model, validate the model, and get the result for each epoch.Then we apply CNN

In this CNN model, there are 3 CNN blocks, and each block consists of 2 convolution layers and 1 max-pooling layer. Relu activation function is used to remove negative values from the feature map because there can not be negative values for any pixel value. Stride(1,1) used and padding is also 1.

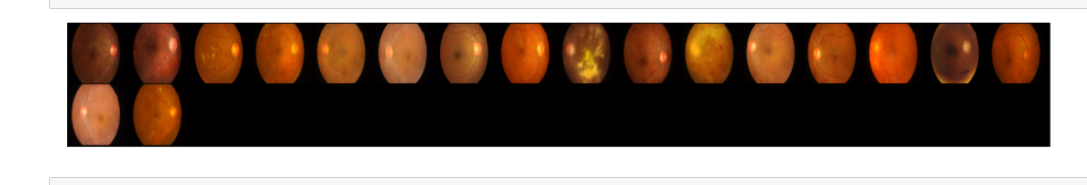
After applying convolution and extract features from the image, a flatten layer is used to flat the tensor which has 3 dimensions. The flatten layer converts the tensor to one-dimensional. Then 3 linear added to reduce the size of the tensor and learn the features.

**Before Segmentation:**

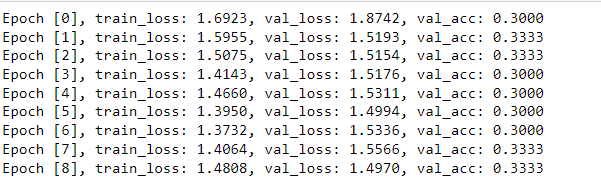
Create Batches of images of size 18 :

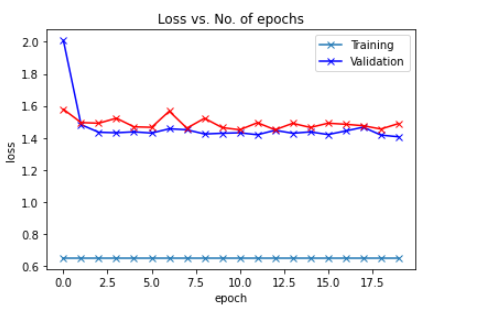


And Plot First Batch :



After running on 20 Epochs The, results are

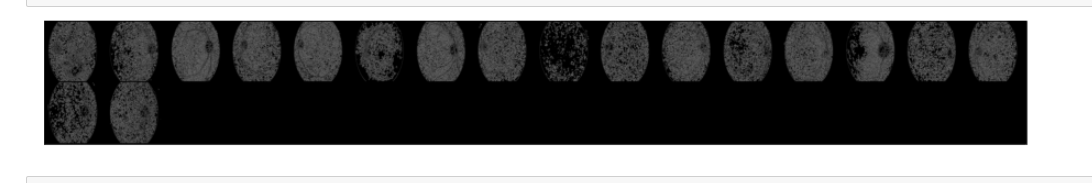




The lost is decreasing and average is increasing on average after 4 epochs ..

**After Segmentation:**

First Batch:



After Running 20 epochs Results seem to be:

