**APPROACH FOR SOLVING THE COMPUTER VISION ASSIGNMENT**

The main Idea behind solving the objectives(Counting rice grains) is to make the provided image in the best possible format.  
If there would be clarity in the image, and rice grains are well separated from the background image then there would be ease in counting them.

Then, Solve the Corner cases and hence build the solution.

There are a total 3 major and challenging parts in building solutions:

1. Image Pre-processing
2. Solving Grains touching problem
3. Counting broken rice

**Image Pre-processing**

Image pre-processing is one of the vital parts of the solution, as, on this itself, Whole Ideology relies.

If the image is perfectly tuned as per the needs, then it becomes easy to work further with the approaches.

The crucial part of Image pre-processing is tuning the Image, it takes a lot of trial and error to fix the parameter to the required value.

As part of the Solution, I have used:

* Conversion of BGR Image to Grayscale Image
* Image Thresholding
* And removing noise from the thresholded image using morphologyEx (Opening)

After all the processes, the clear Image was ready for further use.

**Solving Grains touching Problem**

After successfully Pre-processing the image, there comes the challenging part of the problem statement.

"Counting the rice grains that are touching each other"

It is not even easy for a human eye in the manual inspection process to count the rice grains. The small size and white color create the illusion.

Well for Machines,  
Counting Rice grains is quite easy if they are well separated. The reason, is there are lots of algorithms out there and various techniques that can come in helpful.

But when there is an object touching or overlapping problem then there needs a lot of effort in grains classification.

In our case, it becomes more difficult as the rice grains are quite small in size.

Because of its small size, We can't apply processes like erosion to get the touching corner part separated.

So, For Solving this problem, I have applied the Watershed Algorithm:

Watershed Algorithm is based on extracting sure background and foreground and then using markers will make the watershed algorithm run and detect the exact boundaries.

It is like, filling the valleys and then separating hills out of that.

**Using ML-Model to predict the number of grains with category**

After using the Watershed algorithm, We will get the total number of rice, and the area of each rice grain present in the image, but counting broken rice grains is one of the typical tasks.

The above process is taken as a reference from open-source code, the link is shared below

<https://github.com/mramanindia/Counting_rice_grain>

**Difficulties Faced :**

There are in total 4 ML models Logistic Regression, KNN, Decision tree, and Random Forest models trained to predict the rice grains. Out of these 4 Logistic regression is giving the best F1 score but was not able to predict the False positives correctly. So Random forest is chosen as the model to predict rice grains.

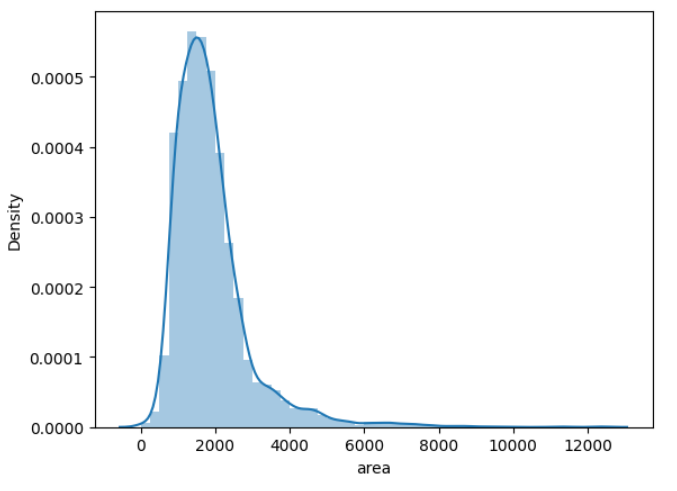
Now using the areas of full and broken grains we shall create labeled data w.r.t the rice type and will train a Machine Learning model using Random Forest algorithms.

The model with the best result is chosen to predict.

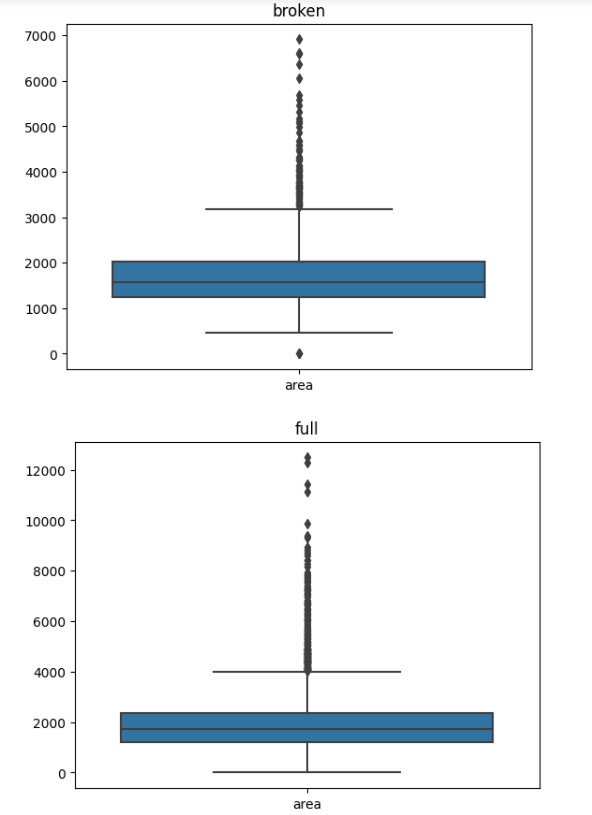
**Limitations & Possible improvements :**

* The watershed algorithm is not able to work well for mixed rice grain data, so it is not used for analysis model training.
* So, the alternative approach is to break the image into smaller chunks so that it can work on mixed rice grain too as they will be less entropy in chunks compared to the whole image
* This above step can be taken as an improvement for this code with given time and resources.

**Visualizations:**



The areas of all the individual rice grains are distributed normally with a mean of 1887.27 and with a standard deviation of 1070



When analyzing the distributions of rice grains of type full and broken it is identified they have outliers.

So, the areas of the rice grains are clipped to the interquartile range respectively for both broken and full grain. The sample code is shared below,

q3 = df1["area"].quantile(0.75)

q1 = df1["area"].quantile(0.25)

iqr = q3-q1

ut = q3+(1.5\*iqr)

lt = q1-(1.5\*iqr)

df1["area"] = df1["area"].clip(lt,ut)