Title: Rice grain counting and analysis

PROBLEM STATEMENT:

Given an image, we aim to create a model that can count the total number of rice grains, the number of broken grains and the number of full grains.

DATA COLLECTION:

The training data for the model consists of 3 broken rice images and 5 full grain images.

IMAGE PROCESSING:

This code uses a Python script for object detection in images, implementing two methods: contour detection and watershed segmentation. Both methods follow similar steps of pre-processing the image (mean shift filtering, gray scaling, thresholding, and noise removal) to obtain a binary image. The contour detection method uses the cv2.findContours and cv2.minEnclosingCircle functions from OpenCV to detect contours and draw circles around them in the original image. The watershed segmentation method uses the cv2.distanceTransform (with a threshold of 20) and cv2.watershed functions to perform the watershed transformation on the binary image and mark the detected objects in the original image.

The output of both methods is an image with the detected objects marked and the information about the total number of objects, average object area, median object area, and the area of each object. The output is returned as a dictionary and converted to a dataframe, which is stored as a .csv file.

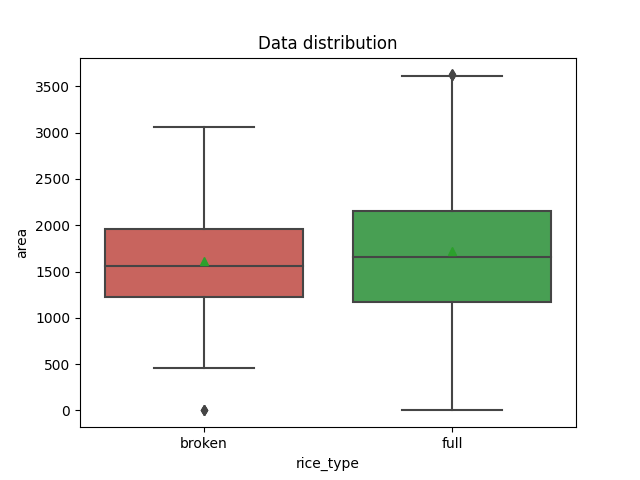
Each image is area is stored in ‘processed\_data/area\_data’ which will be later used to broken and full grains using machine learning model

FEATURE ENGINEERING:

We transformed the extracted rice grain areas into features by creating a dataframe containing the area and rice\_type. The areas extracted from the broken and full grain images were labelled (rice\_type) as broken or full grain and exported as a .csv file named label\_data.csv. This .csv file served as the training data for the model building.

MODEL TRAINING:

We performed EDA on the training data and observed that the distributions and medians were similar, with only minor differences. This indicated that we would need other features to differentiate broken and full grains (to be discussed in the "Future Work" section).



We removed outliers as these extreme values might be due to two or more rice joining each other or some rice being cut in more than half, split the data using random sampling, and encoded the target variable using the sklearn.preprocessing LabelEncoder class. We considered three models for training: logistic regression, decision tree, and random forest. We hyper tuned the decision tree and random forest models using the RandomizedSearchCV method. The hyper tuned parameters included max\_depth, min\_samples\_split, and min\_samples\_leaf for the decision tree and n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf, max\_features, criterion, bootstrap, oob\_score, verbose, random\_state, and n\_jobs for the random forest. The AUC-ROC score for the random forest model was 0.54±2, which was not better than random guessing, indicating the poor ability of the feature (area) to differentiate between broken and full grains.

MODEL SELECTION:

As the random forest model performed slightly better than the other models, it was selected and exported using the pickle module as finalized\_model.pkl.

OUTPUT:

We tested the model on new images to get the count of total rice grains and the count of broken and full grains by using the area data and the final model to categorize the output as broken or full.

CONCLUSION:

We successfully counted the total number of rice grains in an image and classified them as broken or full. However, the differentiation between broken and full grains was not accurate as the area feature was not able to differentiate between the two

FUTURE WORK:

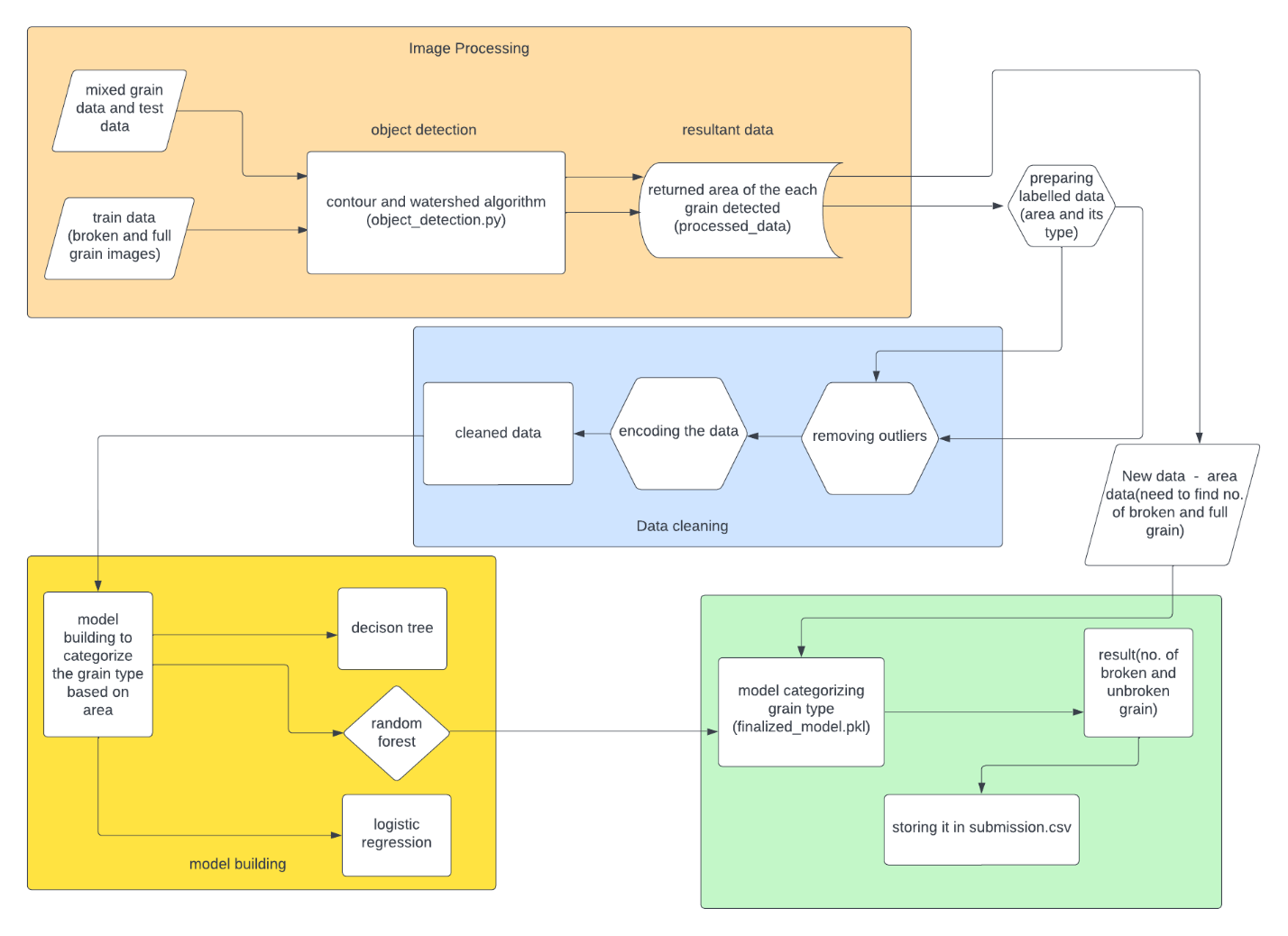
Improving grain differentiation:

a. Limitations: Area as a feature is not effective in differentiating between broken and full grain.

b. Solutions:

i. Consider additional features: The images of each grain are taken from different distances, leading to differences in size between the grain at the boundary and centre of the image. Taking these measurements as input features can improve grain differentiation.

ii. Exploring alternative image segmentation algorithms: Other algorithms such as YOLO or somel edge detection algorithms, others could also be explored to improve grain differentiation.



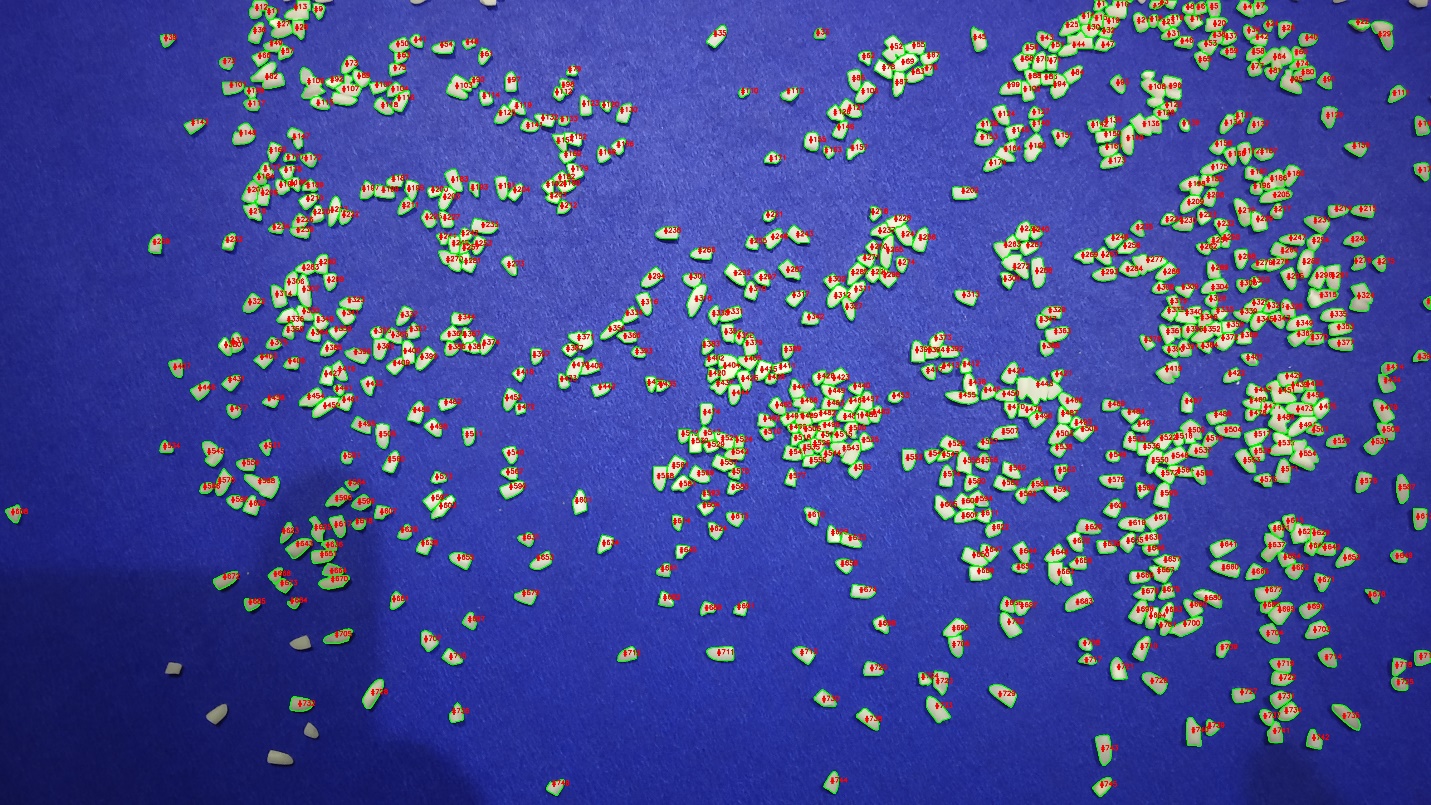
THE PIPELINE

TRAIN DATA:

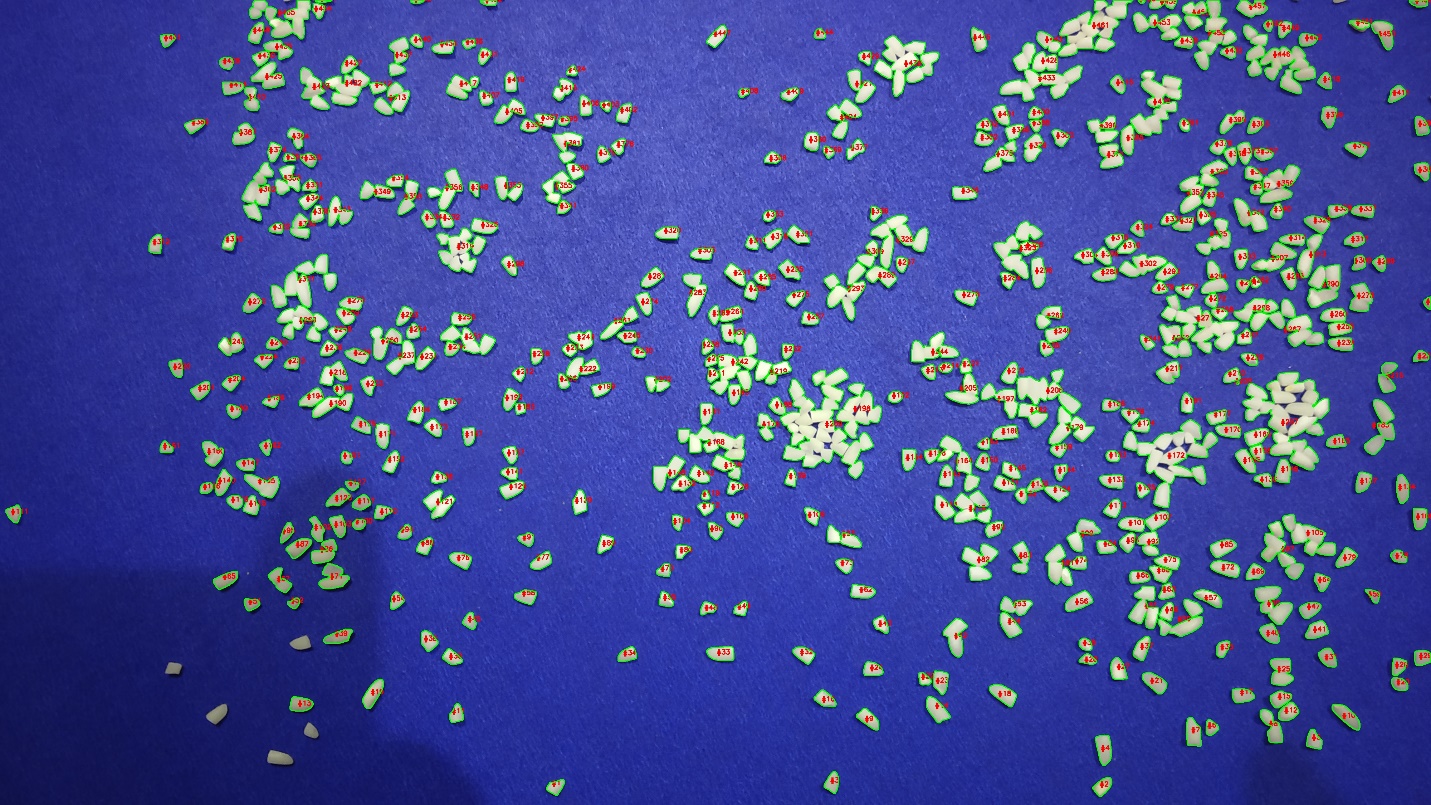
SAMPLE INPUT IMAGE:



OUPUT IMAGE BY WATERSHED ALGORITHM:



OUTPUT BY CONTOUR ALGORITHM:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | method | Total count | Broken rice count | Full grain count |
| broken\_grain\_1.jpg | watershed | 746 | 181 | 565 |

References

<https://medium.com/analytics-vidhya/computer-vision-watershed-algorithm-ca16bd00485>

<https://docs.opencv.org/4.x/d3/db4/tutorial_py_watershed.html>

<https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/#imwrite-function-in-opencv>

<https://pythonprogramming.net/canny-edge-detection-gradients-python-opencv-tutorial/>

<https://github.com/Mushahid2521/Rice-Grain-Purity-Analysis-Using-Deep-Learning>