## Documentation for the Waste Image Classification Hackathon 2025

### Repository Setup

\* GitHub repository named "Waste-Image-Classification-Hackathon-2025"

\* README.md included

### README.md File

1.\*\* Project Overview:  
 The AI Image Classification Hackathon 2025 is a challenge focused on building an image classification model to identify and categorize waste. Participants will go through the entire process of developing and deploying a machine learning solution, including data loading and preprocessing, model training and evaluation, deployment on cloud platforms, API integration, and UI integration.

The project aims to develop a solution for smart waste management using AI.

\* Dataset used:  
 Dataset provided was downloaded from the link provided on Waste Image Classification Dataset from https://prod-dcd datasets-cache-zipfiles.s3.eu-west-1.amazonaws.com/n3gtgm9jxj-2.zip

\* Project Objective  
 The AI Image classification hackathon is a hands-on challenge that will engage participants to build an image classification model that identifies and categorizes waste. Participants will work through data loading, image preprocessing, model training and evaluation, deployment on cloud platforms, API integration, and finally UI integration.

2. \*\*Data Preprocessing:\*\*

\* Data cleaning

**Clean image files:** This involves removing corrupt images and handling missing data.

**Standardize image sizes:** All images should be resized to a fixed size, such as 256x256 pixels.

**Normalize pixel values:** Pixel values should be normalized to a range of [0, 1].

\* Image standardization process  
This process involves resizing all images in the dataset to have the same dimensions. For example, if the dataset has images of varying sizes, such as 100x100, 200x300, and 500x500, the standardization process would resize all of them to a uniform size, such as 256x256 pixels, as suggested in the document.

Image standardization is a crucial preprocessing step because it ensures that all input images have the same dimensions, which is often required by machine learning models.

\* normalization techniques

In machine learning applications involving image data, effective preprocessing of image arrays is essential for optimal model performance, encompassing normalization, data augmentation, and quality enhancement.

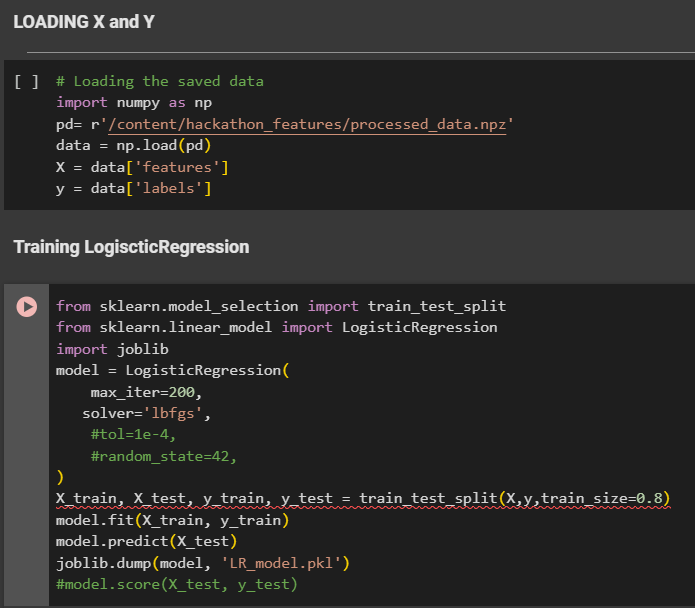
\* Challenges faced during preprocessing  
There were a good amount of challenges that arose during the preprocessing stage, though they were manageable and easy o work through. Some issues included  
 **Variability:** Lighting, perspective, scale, and noise.

 **Data Issues:** Insufficient data, imbalance, mislabeling, and bias.

 **Computational Cost:** Large image sizes and complex pipelines.

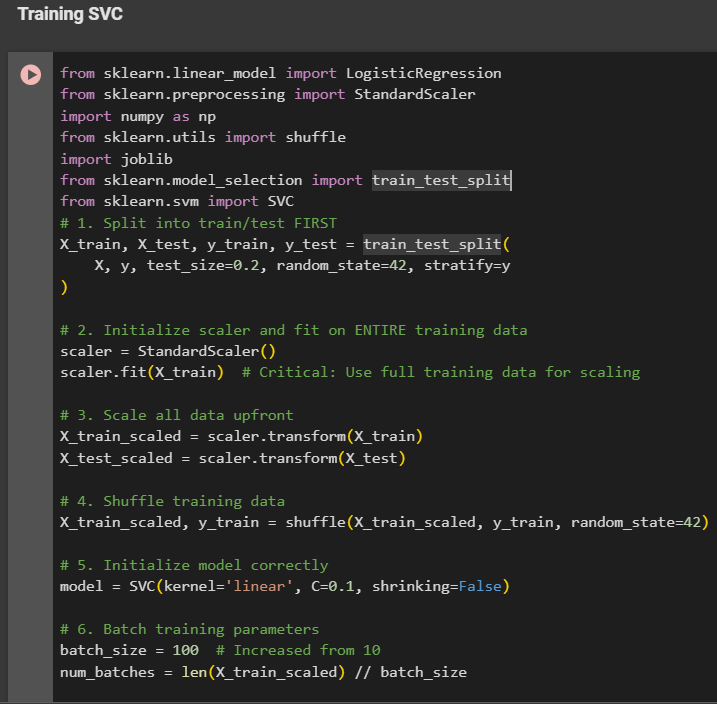
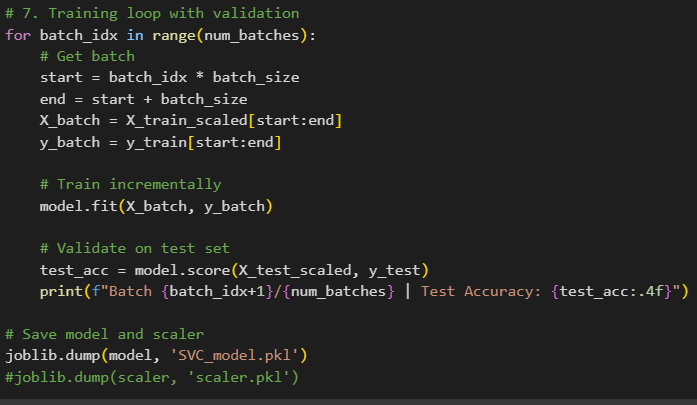
 **Feature Preservation:** Avoiding information loss during processing.

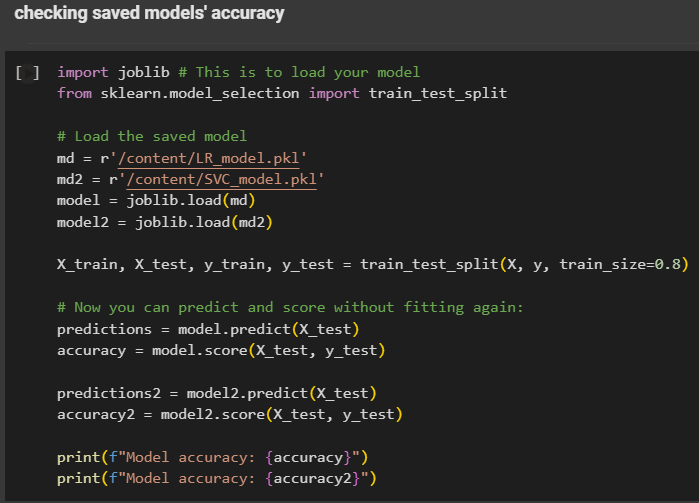
 **Real-time Constraints:** Latency and resource limitations.

3. \*\*Model Training:\*\*

\* Code snippets for model training

The code demonstrates a basic workflow for image classification using a logistic regression model. It loads data, splits it into training and test sets, trains a logistic regression model, makes predictions on the test data, and evaluates the model's performance.





\* Hyperparameters

- Kernel=’linear’  
-C=0.1  
-shrinking=False

4. \*\*Model Evaluation:\*\*

\*Screenshots of the performance metrics (accuracy, precision, recall, F1-score, confusion matrix)[



**SVM Model Performance:**

* **Accuracy:** 67% (moderate)
* **Organic:** Precision 70%, Recall 72%
* **Recyclable:** Precision 63%, Recall 60% (lower performance)
* **Class Imbalance:** More "organic" samples.
* **Confusion:** Errors in both classes, more issues with "recyclable."

5. \*Deployment & API:\*

\* Instructions for deploying the model

**Register:** Store your model in Azure Machine Learning workspace.

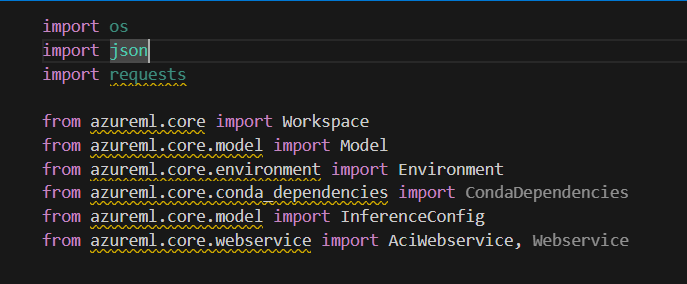
**Entry Script:** Create a script (init(), run()) to load and execute the model.

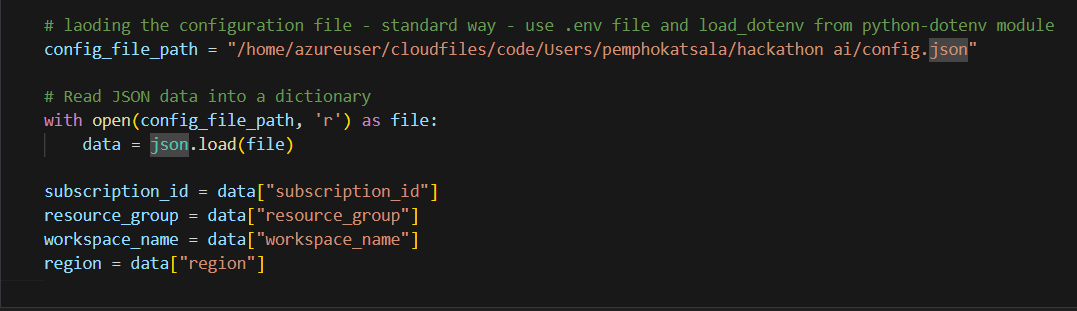
**Inference Config:** Define the environment (dependencies) using Conda or Docker.

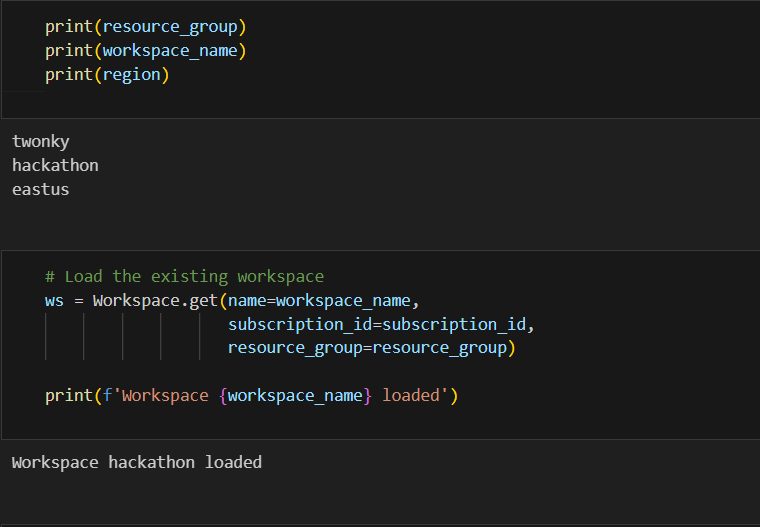
**Compute Target:** Choose AKS (production), ACI (testing), or managed online endpoints.

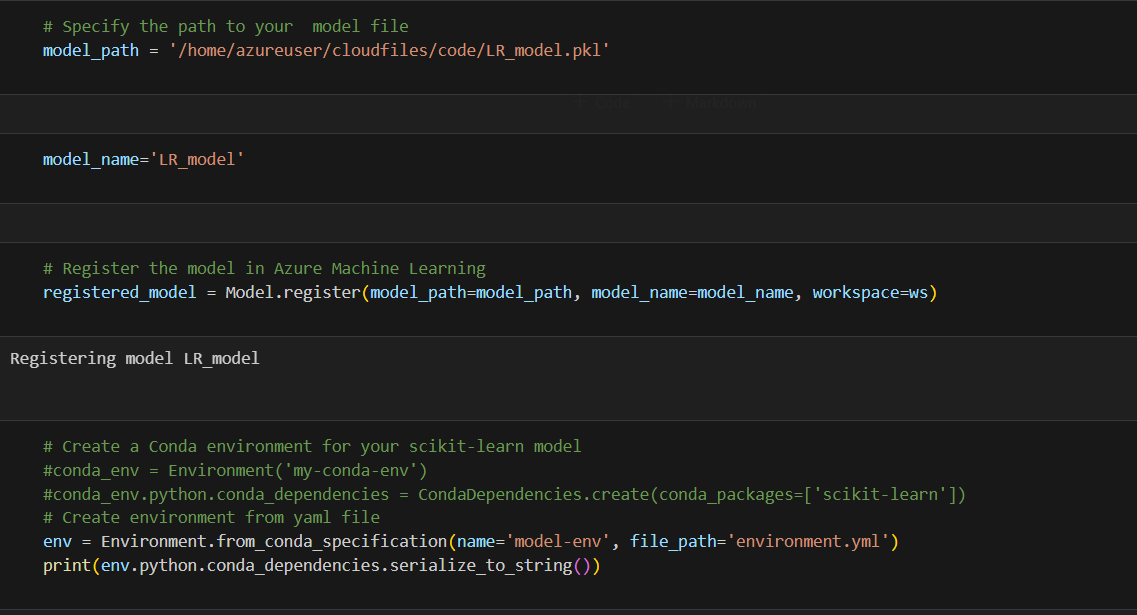
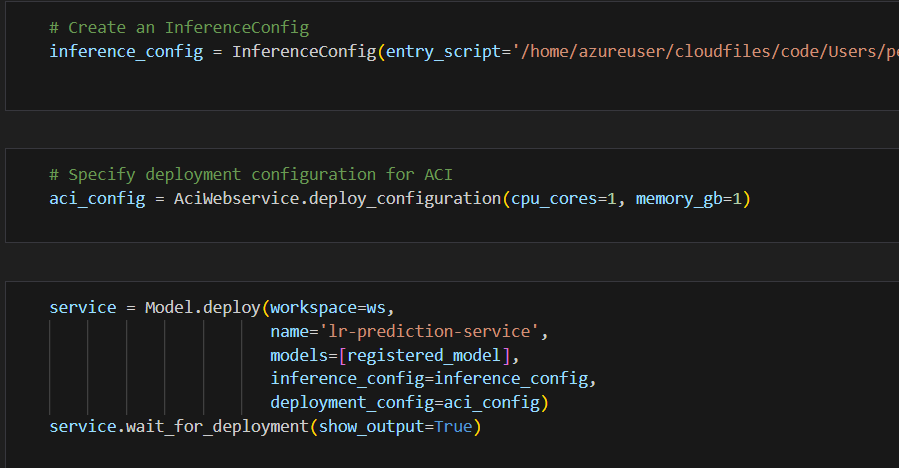
**Deploy:** Deploy the model, script, and config to the target, creating a web service endpoint.

6. \*\*Deployment\*\*

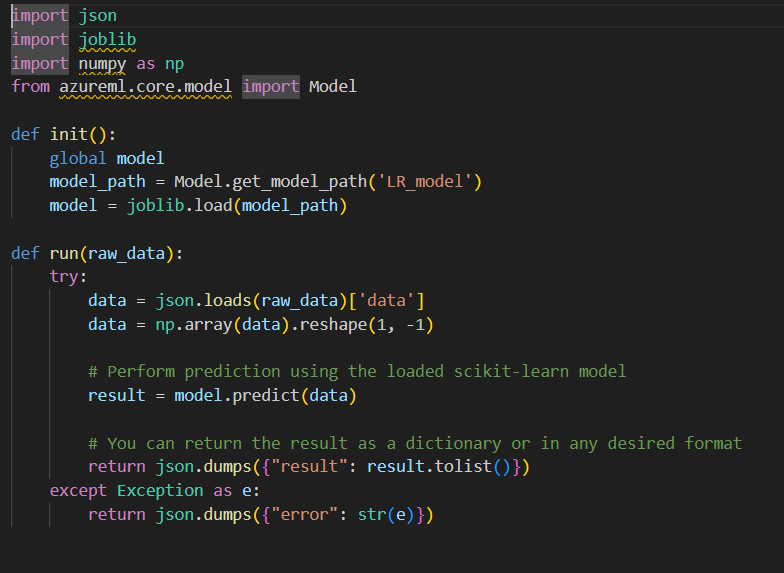








\*\*Score.py



- Loads a pre-trained scikit-learn linear regression model from the Azure ML model registry.

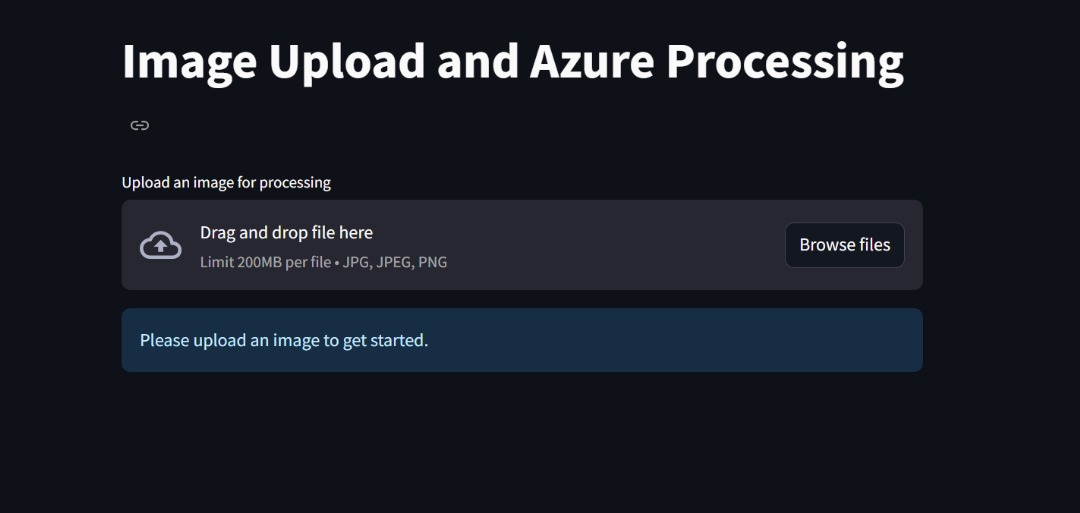
\* Receives input data in JSON format. \* Performs predictions using the loaded model.

\* Returns the prediction results in JSON format.

\* Handles exceptions and returns error messages.

\* It assumes that the deployed model is named 'LR\_model'.

\* It assumes that the input json contains a key named 'data' that holds the array of numerical values used as input for the model.

7. \*\*UI Integration\*\*