1. Introduction

This report outlines the methodology used to extract features from mango images using a custom Python implementation (MangoFeatureExtractor class). The process involves extracting color, shape, and edge features from mango images stored in the train/ and val/ directories. The extracted features are saved as CSV files for further analysis.

To improve computational efficiency, traditional image processing techniques from scikit-image are combined with GPU-accelerated operations using PyTorch.

---

2. Feature Extraction Summary

A wrapper class is designed to extract the following four types of features:

- LAB Color Features
- HSV Color Features
- Shape Features
- Edge Features

From a total of 2,526 mango images (2,019 from training and 507 from validation), a set of 19 features is extracted per image.

---

3. Outlier Detection and Cleaning

To enhance the quality of clustering, the dataset underwent outlier detection using the z-score method. As a result:

- 252 training samples and 50 validation samples were removed as outliers.
- The cleaned dataset now contains 2,224 samples (1,767 train + 457 val).

---

4. Detailed Feature Extraction Process

4.1 LAB Color Features

- Process:
  - Convert RGB to LAB using skimage.color.rgb2lab.
  - Extract L, A, B channels and compute mean and standard deviation using PyTorch tensors (on GPU if available).
- Features Extracted (6 total):
  - LAB_L_mean, LAB_L_std
  - LAB_A_mean, LAB_A_std
  - LAB_B_mean, LAB_B_std

4.2 HSV Color Features

- Process:
  - Convert RGB to HSV using skimage.color.rgb2hsv.
  - Extract H, S, V channels and compute mean and standard deviation using GPU if available.
- Features Extracted (6 total):
  - HSV_H_mean, HSV_H_std
  - HSV_S_mean, HSV_S_std
  - HSV_V_mean, HSV_V_std

4.3 Shape Features

- Process:
  - Convert image to grayscale using rgb2gray.
  - Apply Otsu's thresholding for binary segmentation.
  - Label connected regions and select the largest region (assumed mango).
  - Compute region properties using skimage.measure.regionprops.
- Features Extracted (4 total):
  - area: Number of pixels
  - perimeter: Boundary length
  - aspect_ratio: Ratio of major/minor axis (0 if undefined)
  - eccentricity: Measure of roundness

4.4 Edge Features

- Process:
  - Convert image to grayscale.
  - Apply Canny edge detector (skimage.feature.canny) with sigma=1.0.
  - Calculate:
    - Edge density: Ratio of edge pixels to total pixels
    - Edge intensity (mean & std) using img_as_ubyte and PyTorch
- Features Extracted (3 total):
  - edge_density
  - edge_intensity_mean
  - edge_intensity_std

---

5. Image Processing Pipeline

- Image Loading:
  - Load images using skimage.io.imread.
  - Convert RGBA images to RGB.
- Feature Extraction:
  - The extract_features() method processes images in train/ and val/.
  - Each image is passed through all four feature extractors.
  - Extracted features, image name, and labels are stored in dictionaries.
- Storage:

- Features are appended to CSV files (in append mode) to save memory and support resumption.

---

6. Efficiency and Optimization

- GPU Acceleration:
  - Color and edge statistical computations use PyTorch GPU tensors to accelerate large dataset processing.
- Dynamic CSV Writing:
  - Writing after each image reduces memory usage and increases fault tolerance.

---

7. Dataset Balancing Strategy

The original dataset contains extracted features for mangoes belonging to five varieties, but suffers from class imbalance:

Variety	Samples
Badami	250
Banganapalli	160
Raspuri	392
Mallika	840
Malgova	124

To address this, a hybrid resampling strategy was used:

- SMOTE (Synthetic Minority Over-sampling Technique) for underrepresented classes:
  - Applied to: Badami, Banganpalli, Raspuri, Malgova
- Random Under-sampling for the overrepresented class:
  - Applied to: Mallika

Post-balancing Result

- Each class has exactly 404 samples
- Total samples = 404 × 5 = 2,020
- Slight distribution disturbances may occur due to synthetic generation and random removal.

2. Balancing Strategy

To address class imbalance in the dataset, a two-fold strategy was employed:

- Oversampling with SMOTE: Synthetic Minority Over-sampling Technique (SMOTE) was used to generate synthetic samples for underrepresented classes.
- Undersampling: Random undersampling was applied to reduce the size of the majority class.

A target of 404 samples per class was chosen to strike a balance—minimizing data loss from the majority class while avoiding excessive synthetic data for minority classes.

---

3. Methodology

3.1. Data Preparation

- Features (X): All columns except the image name and the target label were used.
- Target (y): The target column contained the mango variety labels and served as the class variable.

3.2. SMOTE Oversampling

Objective: To increase the number of samples in the minority classes up to 404.

Classes Augmented:

- Malgova: from 124 → 404
- Banganapalli: from 160 → 404
- Badami: from 250 → 404
- Raspuri: from 392 → 404

Technique:
SMOTE generates new samples by interpolating between existing samples in the minority class. This helps preserve the original distribution and characteristics of the dataset.

3.3. Random Undersampling

Objective: To reduce the majority class to 404 samples.

Class Reduced:

- Mallika: from 840 → 404

Technique:
Random undersampling selects a random subset of samples from the majority class, reducing its size without generating synthetic data.

3.4. Pipeline Execution

A preprocessing pipeline was implemented using the imblearn library, which applied:

1. SMOTE for oversampling the minority classes.
2. Random undersampling for the majority class.

This sequence ensured that minority classes were enhanced before reducing the majority class, keeping the balancing process consistent.

3.5. Final Balanced Dataset

After balancing, the processed features and labels were combined into a new dataset, ensuring uniform representation across all classes.

3.6. Verification

The final class distribution was verified using the Counter utility. Each class had exactly 404 samples, confirming successful balancing.

---

4. Rationale for Using SMOTE

SMOTE was chosen based on the nature of the dataset and the need for effective class balancing.

4.1. Preserving Data Integrity

- SMOTE creates synthetic samples based on the original feature space, making it well-suited for maintaining the integrity of numerical features derived from images.

4.2. Reducing Overfitting Risk

- Unlike simple duplication, SMOTE generates unique synthetic examples, reducing the risk of overfitting on repeated data.

4.3. Handling Multiple Minority Classes

- With four minority classes having varying sample sizes, SMOTE allowed precise and tailored oversampling for each, bringing them all to 404 samples.

4.4. Complementing Undersampling

- While undersampling effectively reduces the size of the dominant Mallika class, it can lead to information loss. SMOTE balances this by boosting minority classes without over-relying on synthetic data, providing a more comprehensive solution.

4.5. Compatibility with Feature-Based Data

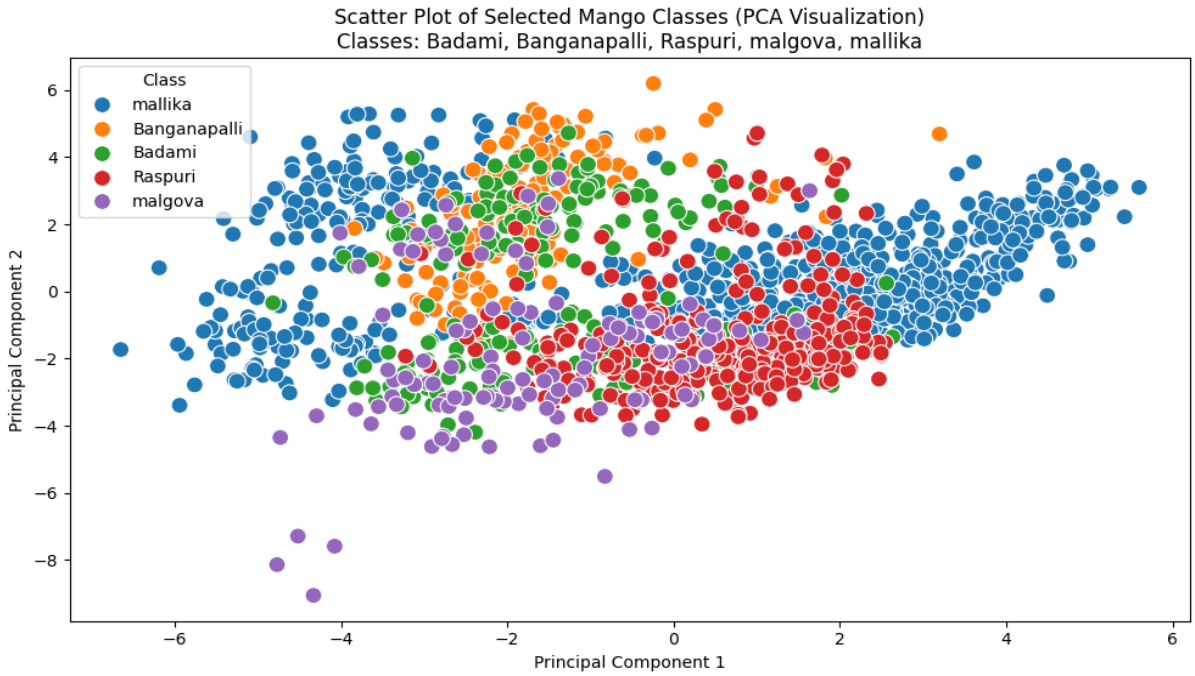
- The dataset consists of continuous, image-derived numerical features. SMOTE's interpolation method works particularly well in such feature spaces.

Target Sample Size Justification:
Choosing 404 samples per class offered a practical compromise—ensuring enough data per class without overloading the model with synthetic data or discarding too much original data.

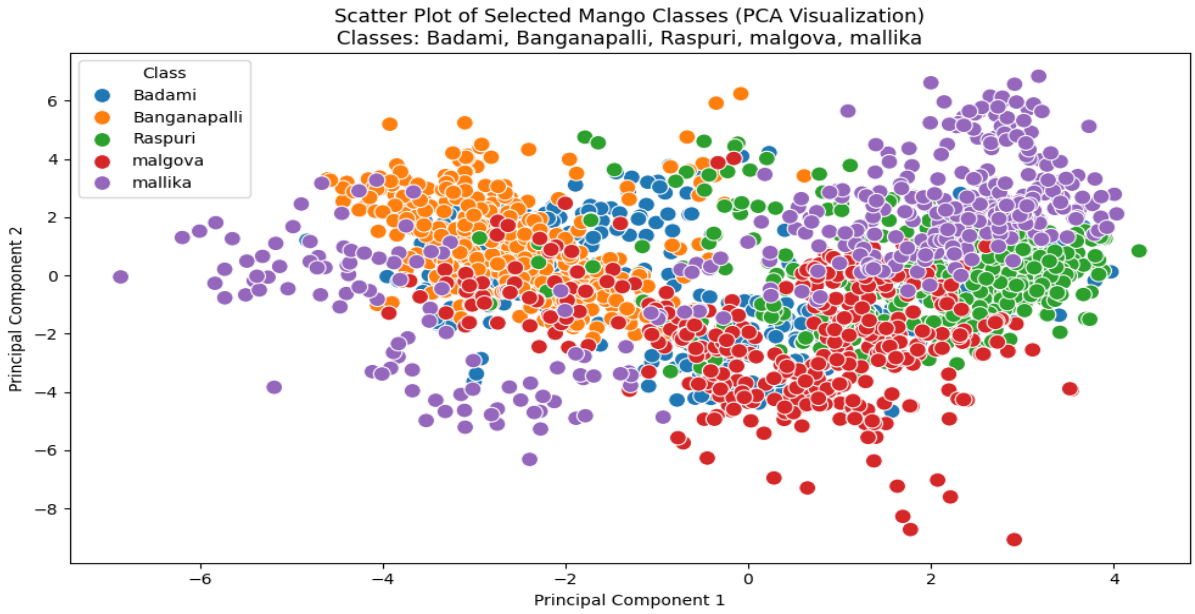
6. Conclusion

The methodology successfully balanced the mango variety dataset by applying SMOTE to oversample minority classes and Random under-sampling to reduce the majority class, resulting in 404 samples per class. This approach enhances model fairness and performance. SMOTE was chosen for its ability to preserve data integrity, handle multiple minority classes, and complement under-sampling, making it an optimal solution given the observed distribution.

- Scatter plot of imbalanced data.



- Scatter plot of balanced data.


Scatter Plot of Selected Mango Classes (PCA Visualization)
Classes: Badami, Banganapalli, Raspuri, malgova, mallika
Class
Badami
Banganapalli
Raspuri
malgova
mallika
Principal Component 2
Principal Component 1
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4