Rice Image Dataset

Egor Bykov

June 6, 2025

Rice Image Dataset description

The dataset contains color images of five different rice grain varieties:

- Arborio,
- Basmati,
- Ipsala,
- Jasmine,
- Karacadag.

The dataset includes 75 000 images: 15 000 from each class.

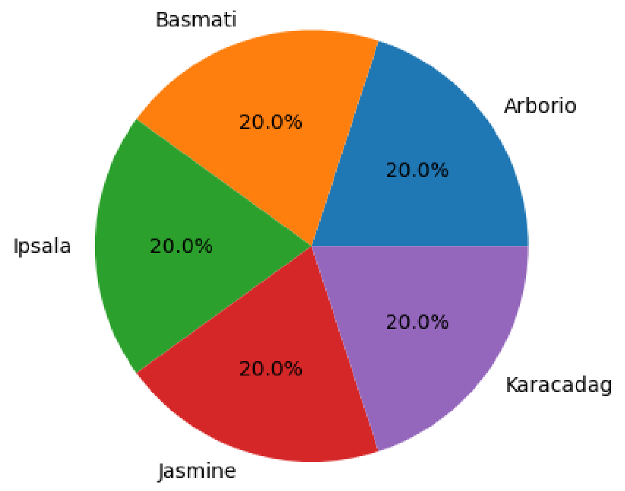


Figure 1: Rice class distribution.

Rice Image Dataset description

- The images are originally in JPEG format and have resolution 250x250 pixels.
- Some rice varieties look similar to each other, while other can be easily distinguished. For example, Basmati is similar to Jasmine variety because of its streched shape, while Arborio and Karacagad are almost round (Figure 2).

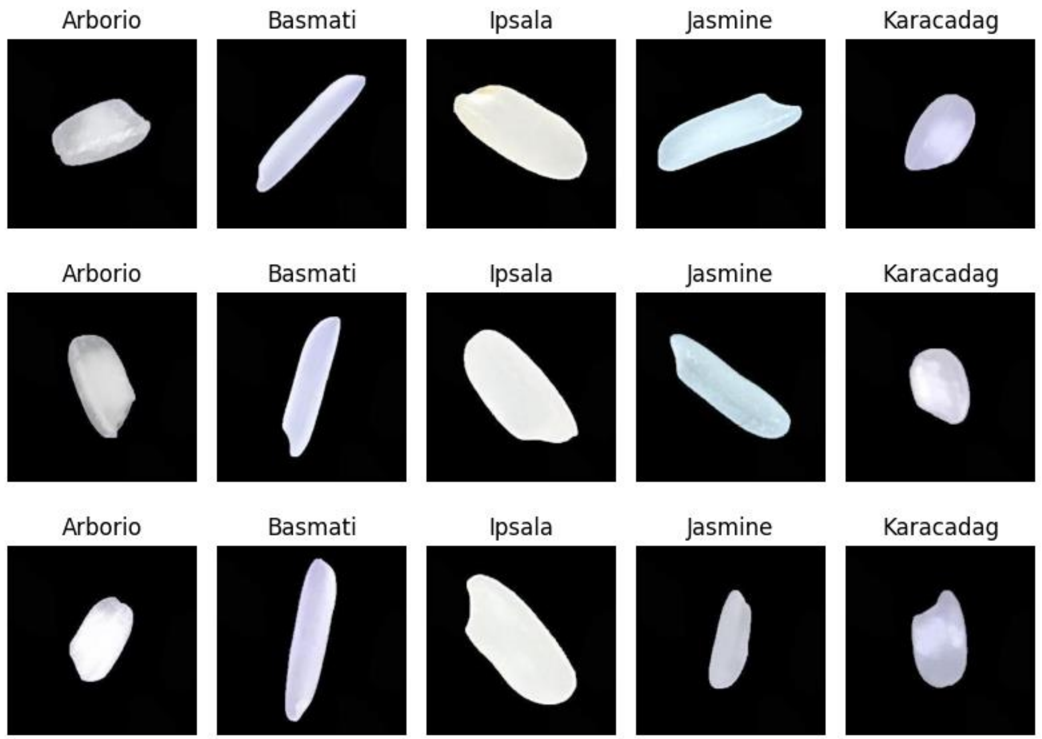


Figure 2: Samples from the dataset.

Rice Image Dataset description

- The images were obtained by a high-resolution camera and lightning equipment.

- The rice varieties differ from each other by such features as texture, shape and color.

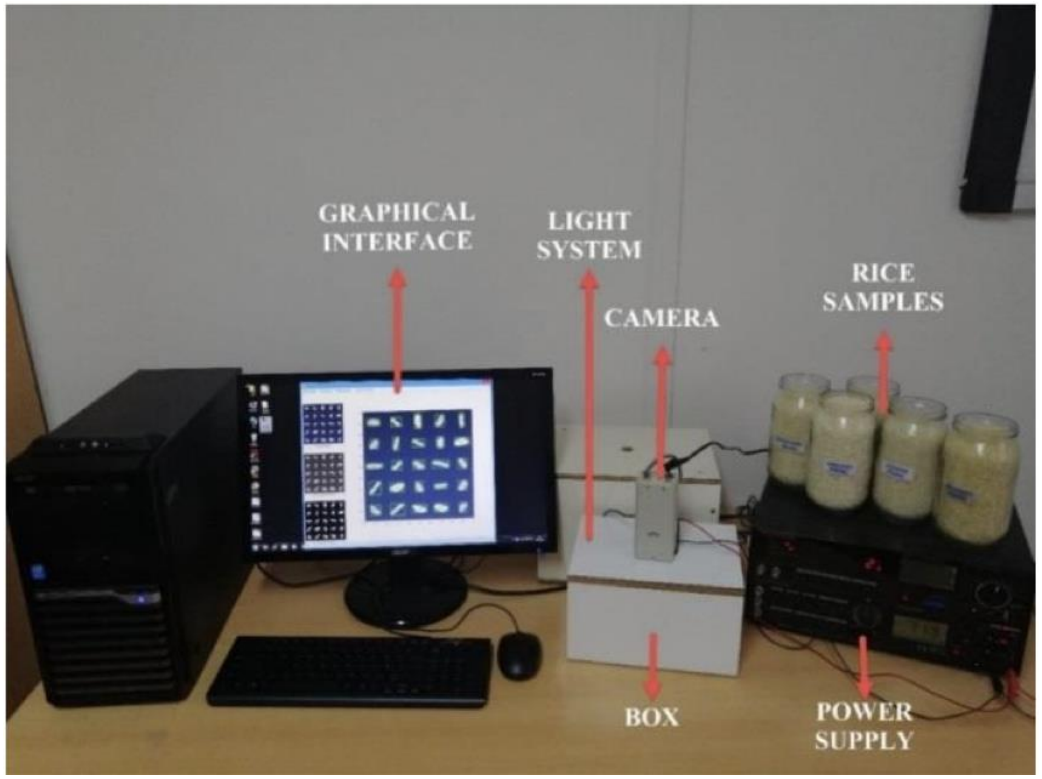


Figure 3: Equipment used to obtain images [1].

[1] Çınar, İlkay, and Murat Köklü. "Determination of effective and specific physical features of rice varieties by computer vision in exterior quality inspection." *Selcuk Journal of Agriculture and Food Sciences* 35.3 (2021): 229-243.

Preprocessing

1) Each image is reshaped to 32x32 RGB format, yielding an input tensor of shape $(N, 32,32,3)$, where N is the total number of images. Then each image is flattened into a vector of length $32 * 32 * 3 = 3072$.
2) The dataset is split into training and test sets using 80-20 ratio. It was done by *train_test_split()* method, with default parameter value *stratify=None*. So after the split there might be a slight class implalance in train and test datasets.

Preprocessing

3) Each input feature was standardized using $StandardScaler()$:

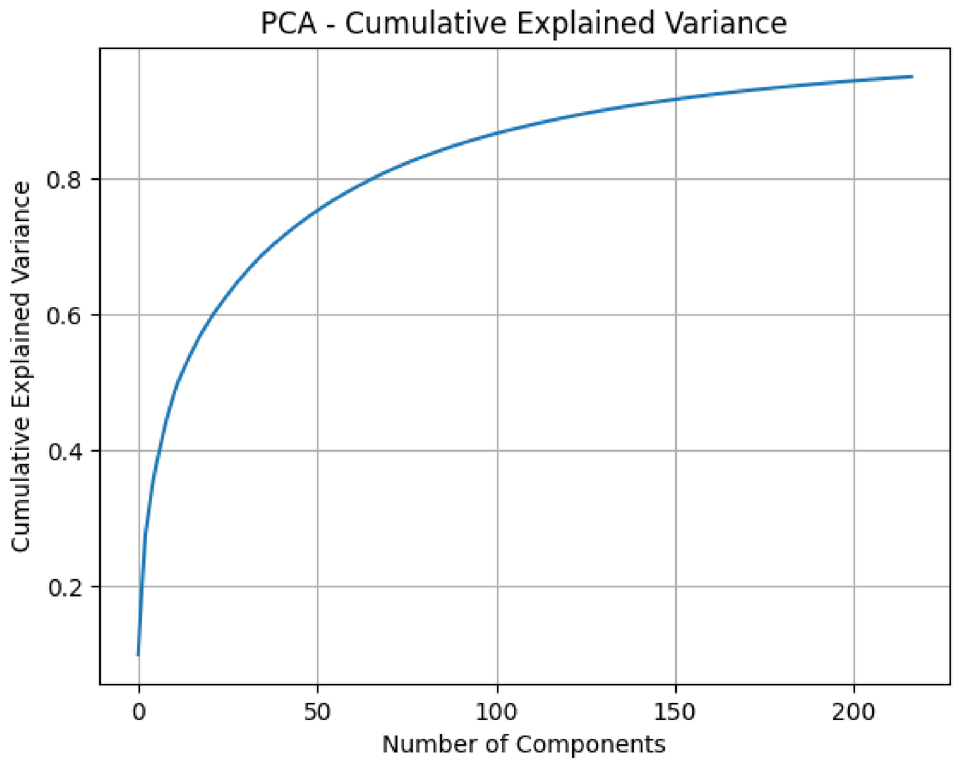
$$z = \frac{x - \mu}{\sigma},$$

Where μ and σ are the mean and standard deviation computed from the training set.

Standardization ensures that each pixel intensity contributes equally to the PCA and classification algorithms.

Preprocessing

4) To reduce the dimensionality of the datasets, *`PCA(n_components=0.95)`* method was applied. It decreased the number of features from 3072 to 217.



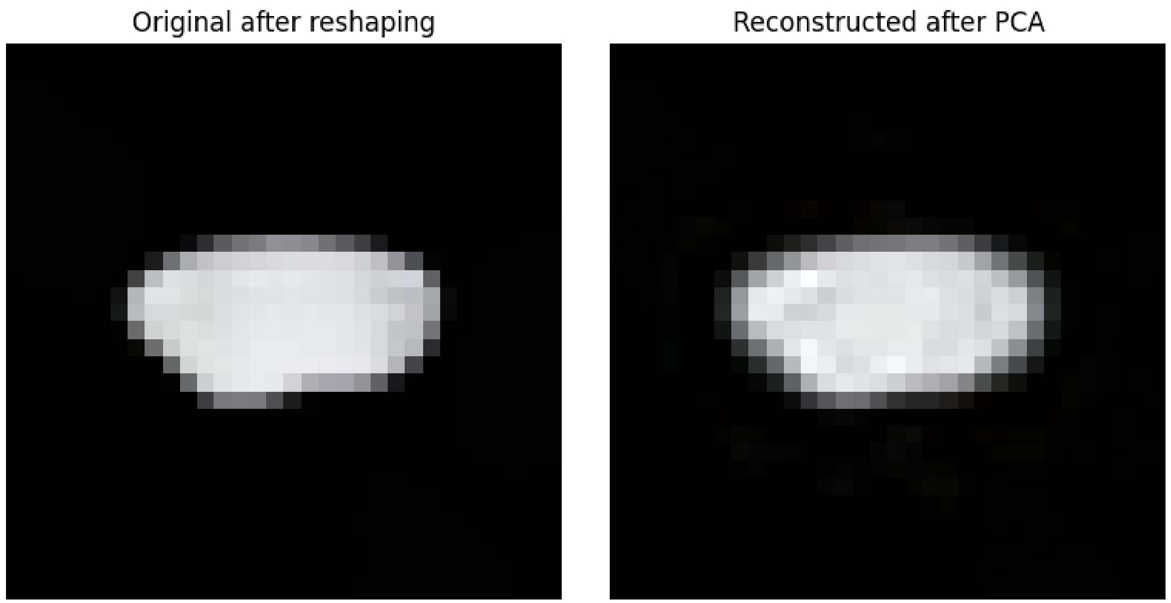


Figure 4: Comparison of rice image after resizing and after PCA

ML models – Random Forest

Each tree in the Random Forest was trained on a bootstrap sample with feature randomness. Final predictions were made by majority vote.

The following hyperparameters were used in the modelling:

- *n_estimators=100* (number of trees)
- *max_features=”sqrt”* (square root of total number of features)

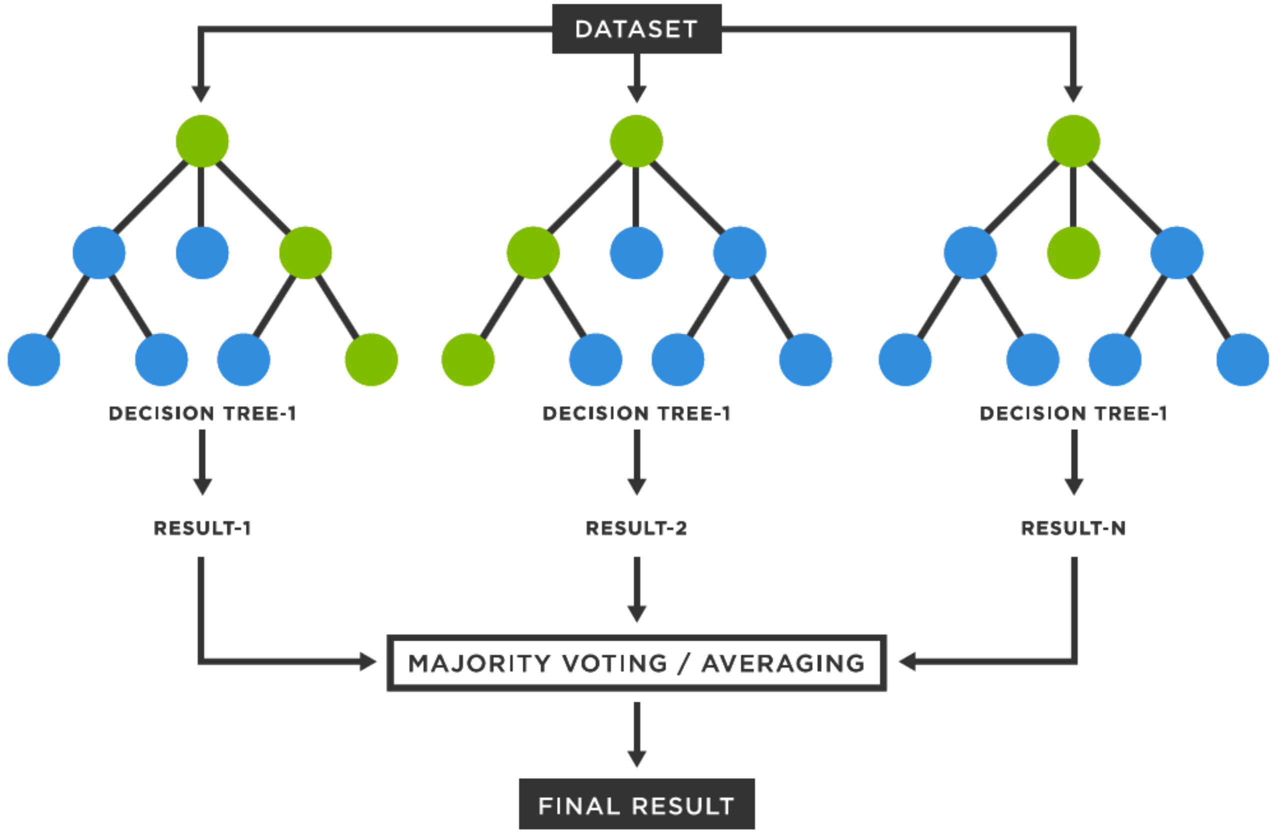


Figure 5: Random Forest illustration

ML models – Random Forest

- The splitting was made using Gini Impurity measure:

$$G(t) = 1 - \sum_{i=1}^{C} p_i^2 ,$$

where p_i is the proportion of class i instances at node t and C is the number of classes.

- Random Forest builds each tree on a different bootstrap sample $D_b \subseteq D$, where D is the full training dataset. Each D_b is generated by sampling n times from D with replacement.
- The splitting feature at each node was selected from a random $\sqrt{p}$ number of features, where p is the total number of features. At each node, the feature and corresponding split that minimizes the weighted average Gini Impurity of the child nodes was selected.

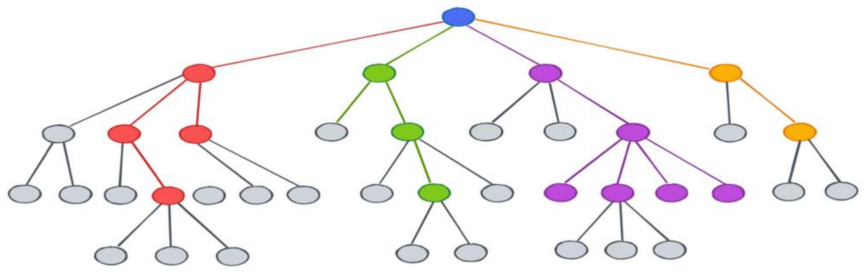
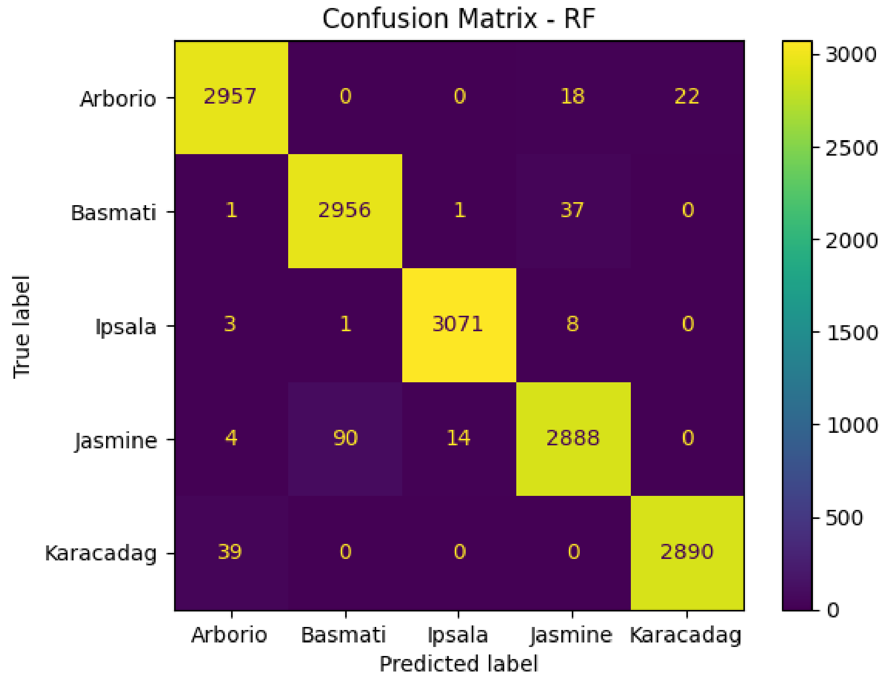


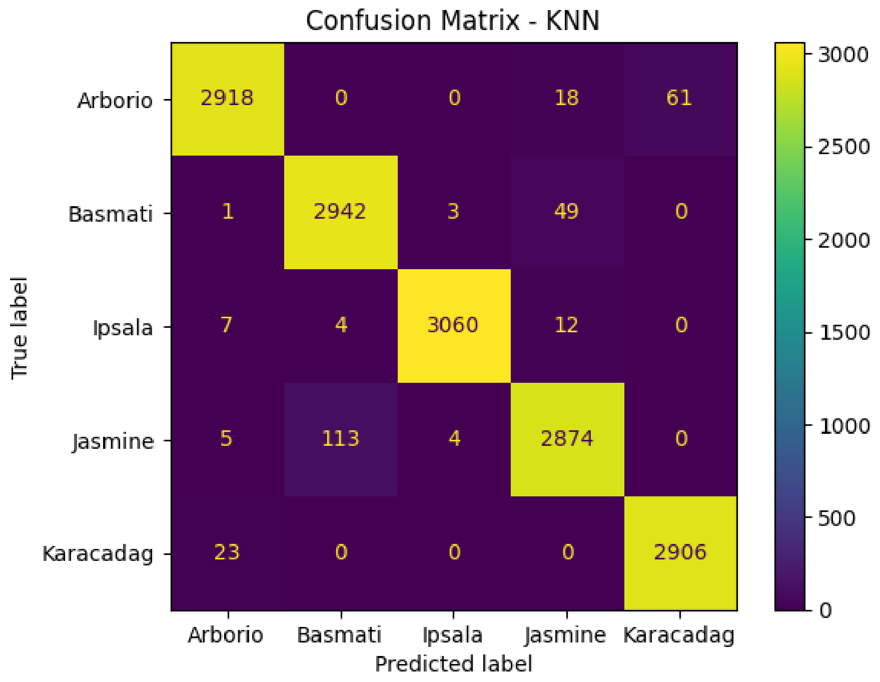
Figure 6: Decision tree illustration.



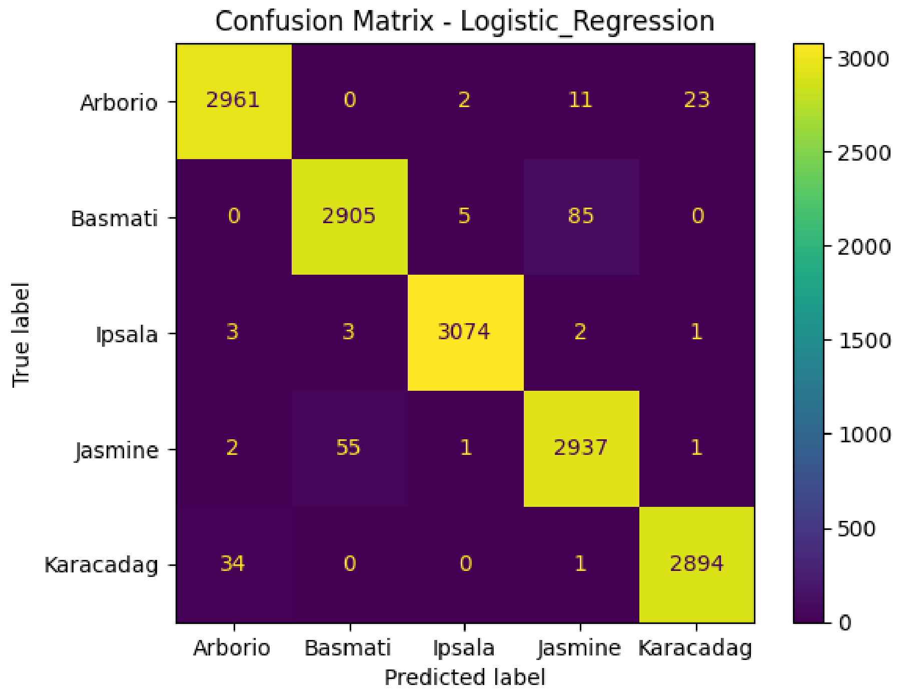
ML models – K-Nearest Neighbors (KNN) and Logistic Regression

Both KNN and Logistic Regression were implemented with default parameters:

- The KNN classifiers was performed with the Euclidean distance parameter and fixed number of neighbors $k = 5$.
- Logistic Regression was implemented with L2 regularization.



True label \ Predicted label	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	2918	0	0	18	61
Basmati	1	2942	3	49	0
Ipsala	7	4	3060	12	0
Jasmine	5	113	4	2874	0
Karacadag	23	0	0	0	2906



True label \ Predicted label	Arborio	Basmati	Ipsala	Jasmine	Karacadag
Arborio	2961	0	2	11	23
Basmati	0	2905	5	85	0
Ipsala	3	3	3074	2	1
Jasmine	2	55	1	2937	1
Karacadag	34	0	0	1	2894

Model evaluation metrics

- Accuracy: Propotion of correct predictions.
- Mean average precision (mAP): Calculated by taking the mean of average precision values over all classes

$$mAP = \frac{1}{C} \sum_{k}^{C} AP_k,$$

where C is the number of classes and AP_k is the average precision of class k.

Results

All classifiers showed good results due to high quality pictures. Logistic regression scored the best Accuracy and Random Forest scored the best mAP.

Metric	Random Forest	KNN	Logistic Regression
Accuracy	0.984	0.980	**0.985**
mAP	**0.998**	0.987	0.991

Code implementation

To speed up computations, the preprocessed image data and trained models were saved as files on Google Drive using Python's `pickle` module. This allowed quick reloading without the need to repeat image preprocessing and model training.

Code implementation

The following classes were created:

- **GoogleDataDownloader**
  - Downloads the ZIP archive using *gdown*.
  - Extracts it to a designed directory.
  - Deletes the ZIP file to save space.
- **ImageLoader**
  - Reads images from each subfolder, resizes them to 32 × 32 pixels, and stores them as NumPy arrays.
  - Assigns integer labels based on class names.
  - Stores the loaded data pickle files, or loads it from disk if already available.
- **Preprocessor**
  - Reshapes the image arrays into 2D arrays.
  - Splits into training and testing sets.
  - Standardizes using *StandardScaler()*.
  - Reduces dimensionality with PCA, retaining 95% of the variance.
  - Saves\loads the trained scaler and PCA objects using *pickle*.

Code implementation

- **ModelEvaluator**
  - Trains and saves the models or loads the models if they are already on the disk.
  - Measures the accuracy and the mean average precision (mAP).
  - Displays the confusion matrix.

Thank you for your attention!