Multiclass Classification of Tomato Plant Diseases from Leaf Images

Marco Caruso – `Student ID: 65836`
Silvia Giannetti – `Student ID: 54978`
Giacomo Alberto Napolitano – `Student ID: 51578`
Federico Silvi – `Student ID: 64257`

Academic Year 2024-2025

1 Introduction

This project report has been translated in English thanks to ChatGPT and has been reviewed by the authors.

The wole project files are available at Github link.

1.1 Project Objectives

The objective of the project is to build a multiclass classifier to recognize various tomato leaf diseases from images.

1.2 Dataset

The dataset chosen for the project is the *PlantVillage Dataset*, which is one of the most widely used in the agricultural domain and is publicly available on Kaggle (Kaggle link).

It contains dozens of diseases associated with various plant species. For each disease, there are 3 types of images:

- Grayscale: grayscale images.
- Color: color images.
- Segmented: images with already segmented leaves.

Each disease has hundreds of images (resolution 256×256), mostly varying in background and lighting.

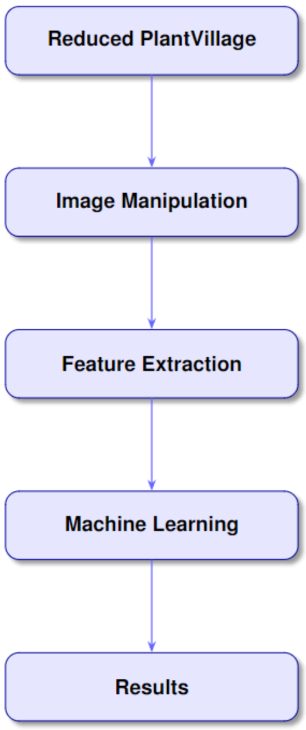
For the project, the tomato plant was selected, for which the dataset includes 9 disease classes + 1 healthy class.

For practical reasons related to lighting quality and disease coloration, the "Early blight" and "Late Blight" classes were discarded. Thus, the dataset actually used includes 7 disease classes + 1 healthy class.

For each of these diseases, 160 images from the 'Color' section were selected to ensure inter-class balance.

1.3 Operational Procedure

The project follows the pipeline below:



2 Image Process & Analysis

The Image Process & Analysis section aims to segment the leaves within the images to extract relevant features.

2.1 Segmentation

2.1.1 Initial Attempts

Various techniques were tested during the project to segment the leaves, but without success, including:

- K-Means ($k = 3$) on the H and S channels in the HSV color space. The chosen k aimed to separate the image into:
  - Leaf
  - Leaf shadow
  - Background

  The goal was to select the cluster corresponding to the leaf. However, this separation did not occur effectively due to image lighting variability.

- Region growing on the H and S channels in HSV. The idea was to define growth rules, but these varied significantly across intra-class and inter-class scenarios.
- Watershed on the H and S channels in HSV. The idea was similar to K-Means, but deep shadows along leaf veins made this method ineffective, often splitting the leaf internally rather than along its border.
- Linear combinations of the H and S channels in HSV. The goal was to find a combination highlighting only the leaf, making segmentation simple. This proved difficult even for similar images (intra-class).

Pre-processing was also attempted to reduce differences between images and improve intra-class results. Edge-preserving filters were used:

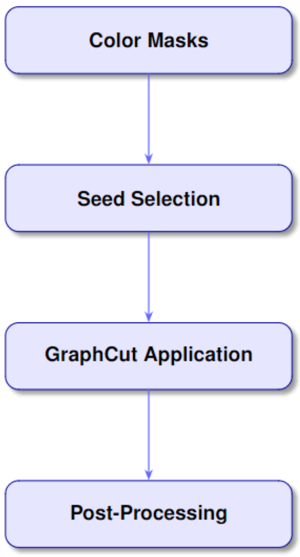
- Bilateral filtering
- Non-local means

Nonetheless, the difficulties remained.

2.1.2 Technique Used

The technique used[1] in the project is graph-based. Initially, GrabCut was tested, assuming the leaf is centered in the image, using a rectangle slightly smaller than the image to avoid algorithm crashes. While this worked for most images, it failed when leaves were near the border.

Thus, a graph-based method, GraphCut, was used, which requires foreground and background seeds. The following pipeline was adopted:



where:

- Color Masks: depending on the class, an HSV color range was selected to reasonably separate the leaf from its shadow and background (not perfectly).
- Seed Selection: the resulting mask underwent:
  - Opening: removing small holes in the foreground (leaf).
  - Area filtering: removing small connected components.
  - Inversion: obtaining background mask.
  - Erosion: ensuring masks contain strictly foreground/background pixels.
- GraphCut Application: the two masks, converted to point lists, were input to GraphCut for segmentation.
- Post-Processing: the GraphCut result was cleaned of artifacts via area filtering of connected components.

  Additionally, the ”Spider Mites” and ”Yellow Curl Virus” classes underwent further processing to remove shadows:

  - Binarization: Otsu thresholding ignored the shadow as it was close to the (now black) background.
  - Closing: ensured any binarization gaps were filled.

Due to the poor quality of the already segmented images in the dataset, it wasn't possible to compute a segmentation quality metric (e.g., Dice Index). Refer to the '`IPA/segmentation contours`' folder for manual evaluation of segmentation quality.

The segmented images are available in the '`IPA/segmented`' folder.

2.2 Feature Extraction

Given the classification task, the following types of texture-based features were selected:

- LBP
- GLCM
- Gabor

The datasets used in the Machine Learning phase were created incrementally by concatenating the features above.

2.2.1 LBP

For Local Binary Patterns, features were extracted from the H and S channels in HSV. These evolved as follows:

- Initially, a radius of 1 and 8 neighbors were used, resulting in dataset D0.
- Then, various radii and neighbor counts were tested to build dataset D1.

Radii	Number of Neighbors	Dataset Name	Dataset Size
1	8	D0	20
(1,2,3)	(8,12,16)	D1	84

2.2.2 GLCM

For Gray Level Co-occurrence Matrices, the following parameters defined the positional operator Q:

[1]For the segmentation technique, refer to the code in `IPA/segmentation.cpp`

- Distances: $[1, 2, 3]$
- Angles: $\left[0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right]$

These extracted GLCMs formed dataset D2, with 102 features[2].

2.2.3 Gabor

For Gabor feature extraction on RGB images, two parallel methods were used:

- Extraction via AlexNet: A pretrained AlexNet was instantiated, and the first 48 convolutional filters were used to extract 96 features by computing the mean and variance for each[3].
- Manual Extraction: The following parameters were used to extract 80 Gabor features manually[4]:
  - Frequencies: [0.05,0.08,0.12,0.18,0.25, 0.35,0.4,0.5,0.6,0.7]
  - Angles: $\left[0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right]$

  For each, mean and variance were computed.

Adding these features resulted in:

- Dataset D3 using Gabor features via AlexNet, with 198 features.
- Dataset D4[5] using manual Gabor features, with 182 features[6].

Since LBP and GLCM features can be noise-sensitive, they were extracted from images filtered using Non-Local Means ($\sigma = 10$). This led to sub-types:

- X_1: features from unfiltered images.
- X_2: features from filtered images.

3 Machine Learning

The Machine Learning section aims to classify leaf diseases based on the features extracted and saved in the datasets described above.

3.1 Models used

The main approach adopted for the task was supervised, considering the following models:

- KNN
- Random Forest
- SVM
- XGBoost

[2]For LBP and GLCM extraction refer to `Feature extraction/LBP&GLCM.py`
[3]For AlexNet-based Gabor extraction refer to `Feature extraction/alexnet.py`
[4]For manual Gabor extraction refer to `Feature extraction/gabor_80.py`
[5]In source files, dataset names may differ slightly from this naming convention
[6]For dataset combination refer to `Feature extraction/combine.py`

In order to compare different paradigms, an unsupervised approach was also tested, using the following models:

- K-Means (n_clusters=8)
- GMM (n_components=8)

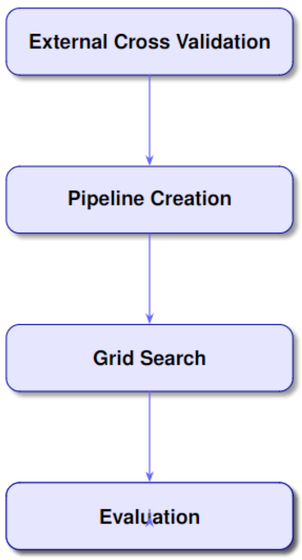
3.2 Datasets used

The datasets[7] used are:

Name	Features	Size
D0_1	Basic LBP	20
D0_2	Basic LBP	20
D1_1	Multiple LBP	84
D1_2	Multiple LBP	84
D2_1	Multiple LBP GLCM	102
D2_2	Multiple LBP GLCM	102
D3_1	Multiple LBP GLCM Gabor (AlexNet)	198
D3_2	Multiple LBP GLCM Gabor (AlexNet)	198
D4_1	Multiple LBP GLCM Gabor (Manual)	183
D4_2	Multiple LBP GLCM Gabor (Manual)	183

3.3 Main pipeline

The logic[8] underlying the code[9] used for training and evaluating the supervised models follows this structure:



where:

- Splitting of the dataset into training and test sets using stratified cross-validation (5 folds), aiming to perform training and evaluation in a statistically robust manner.
- Creation of a pipeline including:

[7]Refer to the folder `Feature extraction/dataset/Segmentation dataset` for the datasets
[8]This was applied for each model on each dataset, resulting in a total of 40 experiments
[9]Code available in `ML/Pipeline.py`

   - Normalization (None, MinMax, Standard Scaler).
   - Dimensionality reduction (None, PCA) focusing on the number of components.
   - Feature selection (None, KBest) focusing on the number of features.
   - Model hyperparameters.

  with the aim of performing a grid search on the training set.

  Specifically, the grid search implements internal cross-validation on the training set (5 folds), in order to obtain the best configuration for the pipeline components, using accuracy as the metric on the validation set (balanced dataset).

- Selection of the best configuration (via majority voting) and subsequent training of the defined model on the training set.
- Evaluation of the model's performance on the test set.

For the unsupervised approach[10] the same logic was followed, with the following differences:

- The metric used was the silhouette score instead of accuracy.
- Cross-validation and grid search were implemented manually.

3.4 Final ensemble

After completing these 40 experiments, an ensemble method (majority and soft voting[11]) was implemented to strengthen the classification.

4 Results

The results presented in this section refer to the best 5 configurations, which were then used in the final ensemble[12].

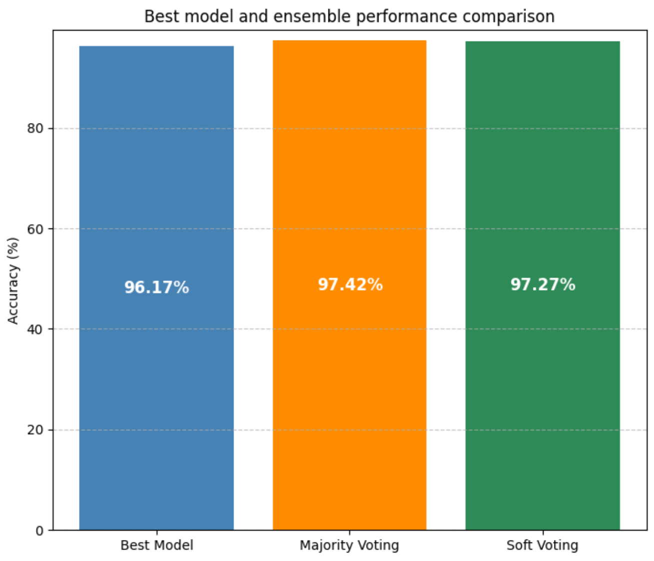
4.1 Leaf segmentation

Dataset	Model	PCA	Feature Selection	Norm.	Acc.
D3_1	XGBoost	180	None	None	96.17%
D3_1	SVM	None	180	Standard Scaler	95.31%
D4_2	SVM	None	160	Standard Scaler	93.05%
D3_2	SVM	None	180	Standard Scaler	91.80%
D4_2	XGBoost	100	None	None	91.72%

The ensemble methods achieved the following:

- Majority: 97.42%
- Soft: 97.27%

[10]Code available in `ML/Pipeline_un.py`
[11]Code in `ML/Best_voting.py` and `ML/Best_soft.py`, respectively
[12]All experiment results can be found in the folders `ML/Supervised` and `ML/Unsupervised`, with a summary available in the Excel file `ML/Results.xlsx`



4.2 No segmentation

The same configurations were considered in the case where the images were not segmented, obtaining:

Dataset	Model	PCA	Feature Selection	Norm.	Acc.
D3_1	XGBoost	180	None	None	94.84%
D3_1	SVM	None	180	Standard Scaler	95.23%
D4_2	SVM	None	160	Standard Scaler	90.39%
D3_2	SVM	None	180	Standard Scaler	95.47%
D4_2	XGBoost	100	None	None	91.80%

The ensemble methods achieved the following:

- Majority: 96.09%
- Soft: 96.72%

4.3 Background segmentation

The same configurations were considered in the case where only the background was segmented, obtaining:

Dataset	Model	PCA	Feature Selection	Norm.	Acc.
D3_1	XGBoost	180	None	None	89.53%
D3_1	SVM	None	180	Standard Scaler	91.64%
D4_2	SVM	None	160	Standard Scaler	87.81%
D3_2	SVM	None	180	Standard Scaler	91.64%
D4_2	XGBoost	100	None	None	85.86%

The ensemble methods achieved the following:

- Majority: 92.73%
- Soft: 93.05%

4.4 Unsupervised

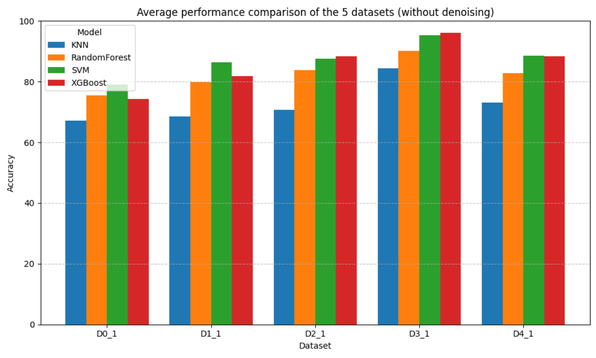
Below are the results of the unsupervised methods:

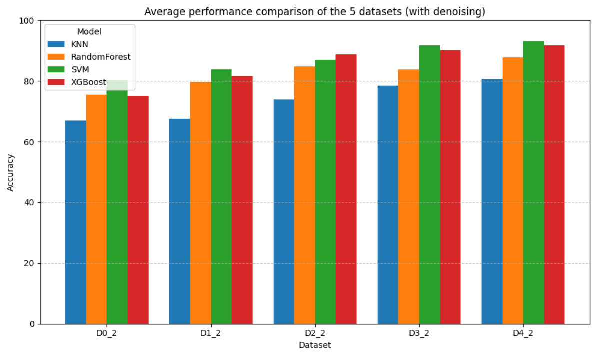
Dataset	Model	PCA	Norm.	Silhouette Score
D2_1	K-Means	30	None	0.47
D2_2	K-Means	30	None	0.47
D3_1	K-Means	80	None	0.47
D4_1	K-Means	80	None	0.47
D4_2	K-Means	80	None	0.47

5 Discussion

5.1 Classification

An initial analysis of all results was carried out by comparing the performance of the models across different datasets:



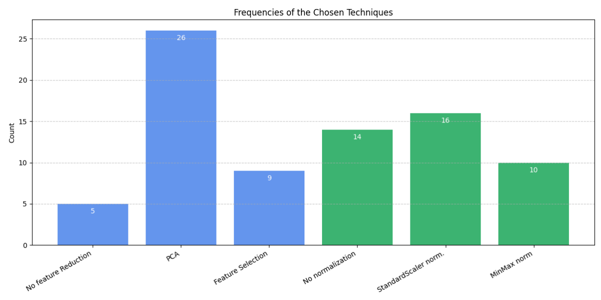


Noting that

- In both dataset types (with and without noise), the best models are consistently SVM or XGBoost.
- The best-performing datasets are those containing all types of extracted features[13].

.

Furthermore, the frequency of the selected components of the pipeline was analyzed:

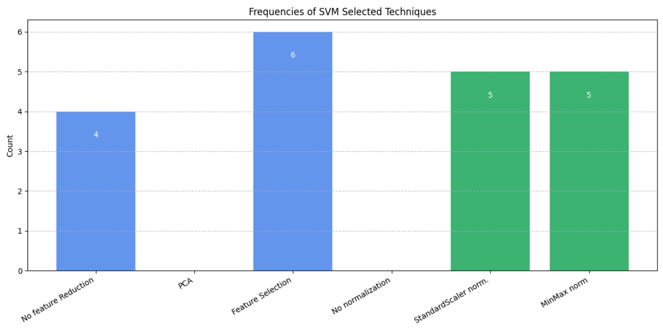


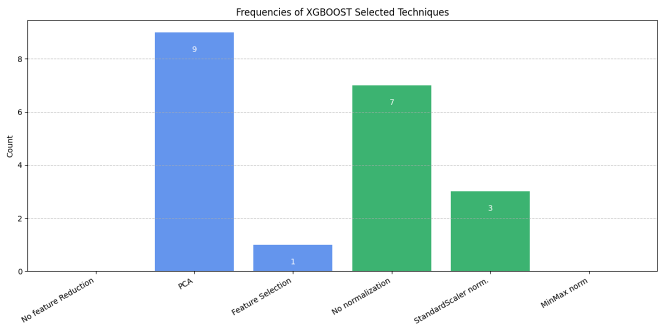
[13]The top 2 configurations use Gabor features extracted via AlexNet

Observing that

- Normalization: the choice is fairly balanced.
- For feature reduction, the use of PCA is much more frequent than both feature selection and applying no technique at all.

By separating these data for the two best-performing models, the following graphs were obtained:

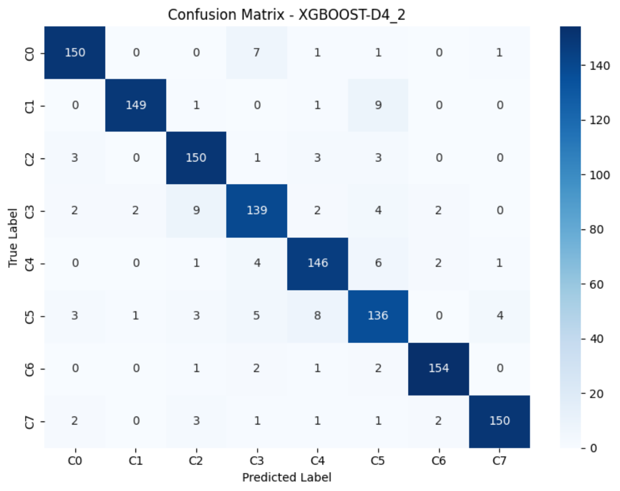




From which it is clear that:

- SVM generally benefits from normalization (no strong preference) and PCA is never applied.
- XGBoost generally prefers the absence of normalization and the application of PCA.

Next, the confusion matrices of the 5 best configurations were analyzed:



True \ Predicted	C0	C1	C2	C3	C4	C5	C6	C7
C0	150	0	0	7	1	1	0	1
C1	0	149	1	0	1	9	0	0
C2	3	0	150	1	3	3	0	0
C3	2	2	9	139	2	4	2	0
C4	0	0	1	4	146	6	2	1
C5	3	1	3	5	8	136	0	4
C6	0	0	1	2	1	2	154	0
C7	2	0	3	1	1	1	2	150

(Example of a confusion matrix)

Noting, across all 5 matrices, that:

- The values on the diagonal are close to the total number of images per class (160).

- The model errors are not symmetric, thus rendering a hierarchical approach to training ineffective.

5.2 Observations on segmentation

To verify the effectiveness of segmentation, the best configurations were applied to datasets generated from unsegmented images.

The results reported in 4.2 may suggest that segmentation is not essential for machine learning algorithms in classification tasks. To further assess this, the best configurations were also applied to datasets generated from images with only the background segmented.

The results reported in 4.3 instead indicate that machine learning can successfully perform classification based solely on the background. This is caused, as discovered through post-hoc image analysis, by intra-class homogeneity of the background.

It was therefore concluded that the excellent results obtained even without segmentation may be amplified by the improper use of features extracted from the background.

5.3 Observations on unsupervised learning

Regarding the results obtained with unsupervised methods (4.4), the following observations were made:

- Although K-Means was the best 9 times out of 10, the performance of the two models is comparable.
- PCA is always preferred, while normalization is never applied.
- The performance stabilizes around 0.47 (Silhouette Score) from dataset type D2 onward.

6 Conclusions

It was thus concluded that the segmentation implemented from the Image Process & Analysis side is necessary in order to obtain an algorithm capable of generalizing based solely on the information contained in the leaf.

From the Machine Learning perspective, it was concluded that supervised methods are significantly superior, as expected, to unsupervised ones.

The best performance obtained with the ensemble method reached 97.42%, making it comparable to the results achieved in the reference literature[14].

[14]Reference link: the performances are obtained on the full original dataset