Classificazione multiclasse di malattie della pianta di pomodoro da immagini di foglie

Marco Caruso – Matricola: 65836
Silvia Giannetti – Matricola: 54978
Giacomo Alberto Napolitano – Matricola: 51578
Federico Silvi – Matricola: 64257

A.A. 2024-2025

1 Introduzione

Tutti i file di progetto sono disponibili su Github (Github link).

1.1 Obiettivi del progetto

L'obiettivo del progetto è quello di andare a creare un classificatore multi-classe per riconoscere diverse malattie delle foglie di pomodoro a partire dalle loro immagini.

1.2 Dataset

Il dataset scelto per il progetto è il *PlantVillage Dataset*, il quale è uno dei più utilizzati nell'ambito agricolo ed è disponibile pubblicamente su Kaggle (Kaggle link).

Esso contiene decine di malattie associate a diverse specie di piante. Per ogni malattia si hanno 3 tipi di immagini:

- Grayscale: immagini in scala di grigi.
- Color: immagini a colori.
- Segmented: immagini con foglie già segmentate.

Ogni malattia dispone di centinaia di immagini (risoluzione 256×256) per la maggior parte differenti per sfondo e illuminazione.

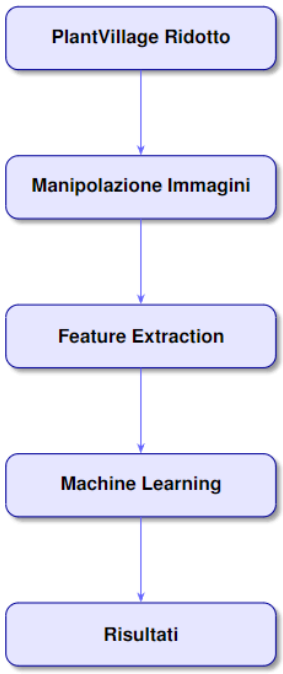
Per il progetto si è scelta la pianta di pomodoro, per la quale il dataset contiene 9 classi di malattie + 1 classe di foglie sane.

Per questioni pratiche legate alla qualità d'illuminazione delle foto abbinata alla colorazione delle malattie, si è scelto di scartare le classi "Early blight" e "Late Blight". Dunque il dataset effettivamente utilizzato contiene 7 classi di malattie + 1 classe di foglie sane.

Per ognuna di queste malattie si sono scelte 160 immagini della sezione 'Color', in modo da avere un bilanciamento inter-classe.

1.3 Procedura operativa

Il progetto segue la seguente pipeline



2 Image Process & Analysis

La parte di Image Process & Analysis mira alla segmentazione delle foglie all'interno delle immagini, in maniera tale da poter estrarre le feature a loro inerenti.

2.1 Segmentazione

2.1.1 Primi tentativi

Nel corso del progetto sono state provate varie tecniche per ottenere la segmentazione delle foglie, senza però successo, tra cui:

- K-Means ($k = 3$) sui canali H e S all'interno dello spazio dei colori HSV. Il valore di k scelto è stato dettato dall'idea di dividere l'immagine in 3 cluster:
  - Foglia
  - Ombra della foglia
  - Background

per poi selezionare il cluster associato alla foglia. Si è osservato che la separazione ipotizzata non veniva effettuata in maniera ottimale a causa dell'eterogeneità in termini di illuminazione delle immagini.

- Region growing sui canali H e S all'interno dello spazio dei colori HSV. L'idea consisteva nell'individuazione di regole di crescita per le immagini, le quali però si sono rivelate fin da subito molto differenti sia scenario intra-classe e inter-classe.

- Watershed sui canali H e S all'interno dello spazio dei colori HSV. L'idea è analoga a quella del K-Means, ma in questo caso la tecnica non ha avuto successo a cause della presenza, nella maggior parte delle immagini, di ombre profonde in corrispondenza delle venature delle foglie. In particolare questo rendeva il Watershed inefficace anche nei casi più "semplici", tagliando la foglia internamente e non lungo il bordo.

- Combinazioni lineari dei canali H e S all'interno dello spazio dei colori HSV. L'idea consisteva nell'individuare una combinazione che portasse a visualizzare esclusivamente la foglia, rendendo quindi molto semplice la sua segmentazione. Ciò non è accaduto a causa della forte difficoltà nel trovare empiricamente una combinazione per immagini anche molto simili tra di loro (intra-classe).

Si è anche tentato di applicare un pre-processing in modo da attenuare le differenze tra le immagini e quindi ottenere un risultato discreto almeno nello scenario intra-classe. In particolare sono stati applicati i seguenti filtri edge-preserving:

- Bilateral filtering

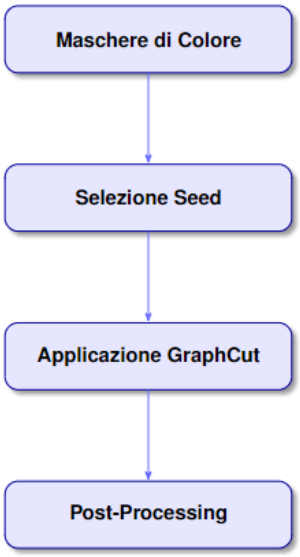
- Non local means

Ciononostante le difficoltà permanevano.

2.1.2 Tecnica utilizzata

La tecnica utilizzata[1] effettivamente nel progetto si basa su un approccio graph-based. In particolare si è dapprima tentato di utilizzare direttamente il GrabCut, volendo sfruttare l'ipotesi che la foglia è sempre al centro dell'immagine, fornendo quindi come riquadro che contenesse la foglia un rettangolo poco più piccolo dell'immagine stessa (per evitare il crash dell'algoritmo). Ciò, sebbene funzionasse bene per la maggioranza delle immagini, falliva per quelle in cui le foglie erano molto vicine al bordo.

Di conseguenza, volendo utilizzare sempre un metodo graph-based, si è optato per il GraphCut, che necessita però di seed di foreground e di background. Si è seguita quindi la seguente pipeline:



in cui:

- Maschere di Colore: a seconda della classe analizzata, si è individuato un range all'interno dello spazio dei colori HSV, per il quale la foglia veniva separata discretamente sia dalla sua ombra che dal background (non perfettamente).

- Selezione Seed: la maschera risultante dal passaggio precedente è stata soggetta alle seguenti modifiche:
  - Opening: eliminazione di piccoli fori all'interno del foreground (inteso come foglia).
  - Filtraggio per area: eliminazione di connected components minori.
  - Inversione: ottenimento di maschera di background.
  - Erosion: garantisce che le maschere abbiano pixel strettamente appartenenti al foreground e al background.

- Applicazione GraphCut: le due maschere, una volta convertite in una lista di punti, vengono fornite al GraphCut per la segmentazione della foglia.

- Post-Processing: il risultato del Graphcut viene ripulito da possibili artefatti tramite un filtraggio per area delle connceted components.

  Inoltre le classi "Spider Mites" e "Yellow Curl Virus" sono soggette ad un ulteriore processo per l'eliminazione dell'ombra. In particolare:
  - Binarizzazione: applicando Otsu si ignora l'ombra in quanto molto vicina al background (adesso nero).
  - Closing: garantisce che eventuali buchi nella binarizzazione vengano colmati.

Data la qualità scadente delle foto già segmentate all'interno del dataset, non è stato possibile calcolare una metrica per quantificare la bontà della segmentazione (Dice Index). Si faccia quindi riferimento alla cartella '`IPA/segmentation contours`' per verificare manualmente la qualità del processo di segmentazione.

[1] Per la tecnica di segementazione fare riferimento al codice in `IPA/segmentation.cpp`

Le immagini risultanti dal processo di segmentazione sono disponibili all'interno della cartella '`IPA/segmented`'.

2.2 Feature extraction

Dato il task preposto, si sono scelte le seguenti tipologie di feature

- LBP
- GLCM
- Gabor

correlate alla descrizione delle texture.

I dataset utilizzati nella parte di Machine Learning sono stati creati in maniera incrementali, concatenando le feature riportate sopra.

2.2.1 LBP

Per i Local Binary Pattern si sono estratte le feature nei canali H e S nello spazio dei colori HSV. Queste feature hanno seguito la seguente evoluzione:

- Dapprima si è considerato il raggio unitario e un numero di vicini pari a 8, andando a creare il dataset denominato D0.
- Succesivamente si sono considerati vari raggi e numero di vicini andando a costruire il dataset denominato D1.

Quindi

Raggi	Numero vicini	Nome Dataset	Dimensione Dataset
1	8	D0	20
(1,2,3)	(8,12,16)	D1	84

2.2.2 GLCM

Per le Gray Level Co-ocurence Matrix si sono considerati i seguenti parametri per la definizione dell'operatore di posizione Q:

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

Aggiungendo le GLCM estratte nella maniera descritta si è costruito il dataset denominato D2, con dimensioni pari a 102[2].

2.2.3 Gabor

Per l'estrazione delle feature di Gabor sulle immagini nei canali RGB, si sono seguite due strade parallele:

- Estrazione tramite AlexNet: si è istanziata Alexnet pre-addestrata e si sono estratti i primi 48 filtri convoluzionali e si sono estratte 96 feature calcolando per ciascun media e varianza[3].
- Estrazione manuale: si sono definiti i seguenti parametri per l'estrazione di 80 feature di Gabor[4]:
  - Frequenze: $[0.05,0.08,0.12,0.18,0.25,\ 0.35,0.4,0.5,0.6,0.7]$
  - Angoli: $\left[0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}\right]$

  Per ognuna si considera media e varianza.

Aggiungendo queste feature si ottengono:

- Con le feature di Gabor tramite AlexNet si ottiene il dataset denominato D3, con un numero di feature pari a 198.
- Con le feature di Gabor manuali si ottiene il dataset denominato D4[5], con un numero di feature pari a 182[6].

Poiché le feature LBP e GLCM possono essere sensibili al rumore si è scelto di estrarre queste feature partendo da immagini filtrate tramite Non-Local Means ($\sigma = 10$). Per questo si sono divise le tipologie in sotto-tipologie, in particolare:

- X_1: feature estratte da immagini non filtrate.
- X_2: feature estratte da immagini filtrate.

3 Machine Learning

La parte di Machine Learning mira alla classificazione delle malattie delle foglie a partire dalle feature estratte e salvate nei dataset sopra descritti.

3.1 Modelli utilizzati

L'approccio principale utilizzato per il task è stato supervised, considerando i seguenti modelli:

- KNN
- Random Forest
- SVM
- XGBoost

Volendo poi apportare un confronto tra differenti paradigmi si è anche provato un approccio unsupervised, con i seguenti modelli:

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

3.2 Dataset utilizzati

I dataset[7] utilizzati sono:

[2]Per l'estrazione degli LBP e GLCM fare riferimento al codice `Feature extraction/LBP&GLCM.py`

[3]Per l'estrazione delle feature di Gabor tramite AlexNet fare riferimento al codice `Feature extraction/alexnet.py`

[4]Per l'estrazione manuale delle feature di Gabor fare riferimento al codice `Feature extraction/gabor_80.py`

[5]Nei file sorgente, i dataset hanno nomi estesi rispetto alla nomenclatura qui utilizzata

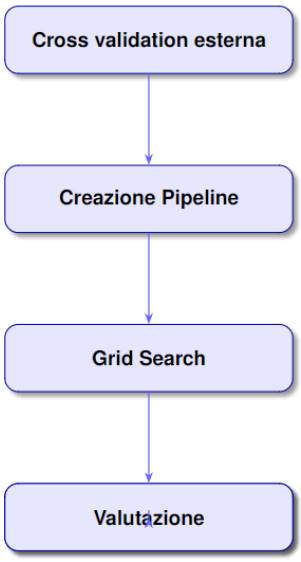
[6]Per la combinazione di dataset fare riferimento al codice `Feature extraction/combine.py`

[7]Per i dataset fare riferimento alla cartella `Feature extraction/dataset/Segmentation dataset`

Nome	Feature	Dimensione
D0_1	LBP base	20
D0_2	LBP base	20
D1_1	LBP multipli	84
D1_2	LBP multipli	84
D2_1	LBP multipli GLCM	102
D2_2	LBP multipli GLCM	102
D3_1	LBP multipli GLCM Gabor (AlexNet)	198
D3_2	LBP multipli GLCM Gabor (AlexNet)	198
D4_1	LBP multipli GLCM Gabor (Manuali)	183
D4_2	LBP multipli GLCM Gabor (Manuali)	183

3.3 Pipeline principale

La logica[8] sottostante il codice utilizzato[9] per l'addestramento e la valutazione dei modelli supervised segue questa struttura:



dove:

- Divisione del dataset in test set e train set attraverso una cross-validation stratificata (5 fold), con l'obiettivo di effettuare addestramento e valutazione in maniera statisticamente robusta.
- Creazione di una pipeline circa
  - Normalizzazione (None, MinMax, Standard Scaler).
  - Riduzione dimensionale (None, PCA) con focus sul numero dei componenti.
  - Feature selection (None, KBest) con focus sul numero di feature.
  - Iper-parametri del modello.

  con l'obiettivo di effettuare una grid search sul train set precedentemente ottenuto.

  In particolare la grid search implementa una cross-validation interna al train set (5 fold), in modo da ottenere la migliore configurazione per le voci della pipeline, usando come metrica sul validation set l'accuracy (dataset bilanciati).
- Scelta della migliore configurazione (tramite majority voting) e successivo addestramento del modello così definito sul train set.
- Valutazione della performance del modello sul test set.

Per l'approccio unsupervised[10] si è seguita la stessa logica, con le seguenti differenze:

- La metrica utilizzata, invece dell'accuracy, è il silhouette score.
- La cross-validation e la grid search sono state implementate a mano.

3.4 Ensamble finale

Dopo aver eseguito questi 40 esperimenti, si è scelto di implementare un ensamble method (majority e soft voting[11]) in modo da irrobustire la classificazione.

4 Risultati

I risultati riportati in questa sezione fanno riferimento alle migliori 5 configurazioni, utilizzate poi per l'ensamble finale[12].

4.1 Segmentazione foglia

Dataset	Modello	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%

Gli ensamble method ottengono rispettivamente:

- Majority: 97.42%
- Soft: 97.27%

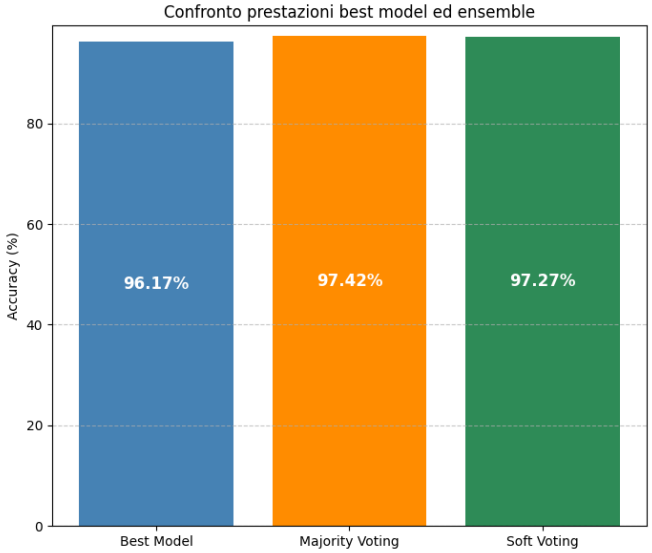
[8] Questa è stata applicata per ogni modello su ogni dataset, portando ad un totale di 40 esperimenti

[9] Codice all'interno di `ML/Pipeline.py`

[10] Codice all'interno di `ML/Pipeline_un.py`

[11] Codice rispettivamente in `ML/Best_voting.py` e `ML/Best_soft.py`

[12] Tutti i risultati degli esperimenti sono consultabili nelle cartelle `ML/Supervised` e `ML/Unspurvised`, con una sintesi disponibile nel file Excel `ML/Risultati.xlsx`



4.2 No segmentazione

Le stesse configurazioni sono state considerate nel caso in cui le immagini non siano segmentate, ottenendo

Dataset	Modello	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%

Gli ensamble method ottengono rispettivamente:

- Majority: 96.09%
- Soft: 96.72%

4.3 Segmentazione sfondo

Le stesse configurazioni sono state considerate nel caso in cui sia stato segmentato solo lo sfondo, ottenendo:

Dataset	Modello	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%

Gli ensamble method ottengono rispettivamente:

- Majority: 92.73%
- Soft: 93.05%

4.4 Unsupervised

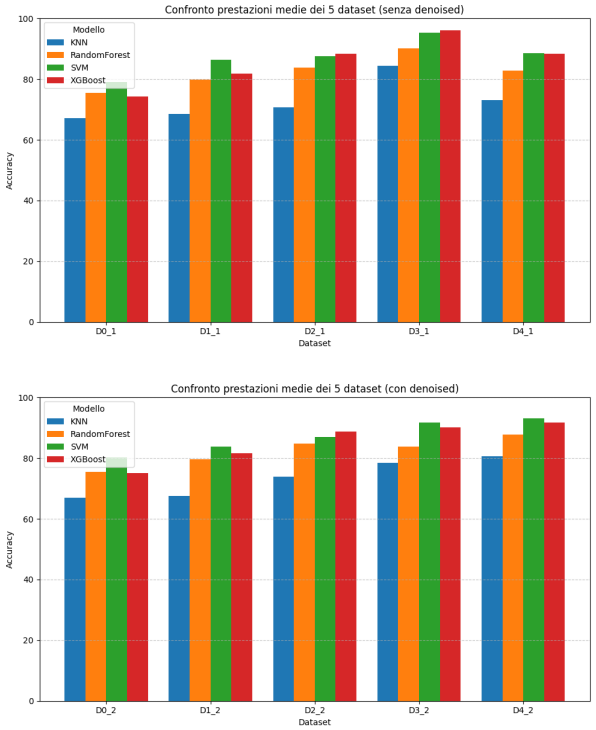
Di seguito i risultati dei metodi unsupervised:

Dataset	Modello	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 Discussione

5.1 Classificazione

Una prima analisi su tutti risultati è stata compiuta sul confronto delle prestazioni tra modelli nei diversi dataset:

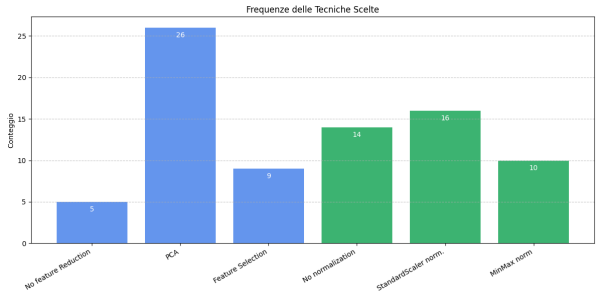


Notando che

- In entrambe le tipologie di dataset (con e senza rumore), i migliori modello sono sempre SVM o XGBoost.
- I dataset migliori sono quelli contenenti tutte le tipologie di feature estratte[13].

[13] Le configurazioni top 2 utilizzano le feature di Gabor estratte da AlexNet

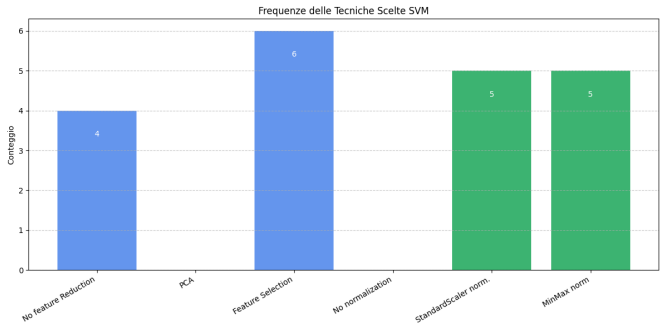
Inoltre si è analizzato la frequenza delle scelte dei singoli componenti della pipeline:

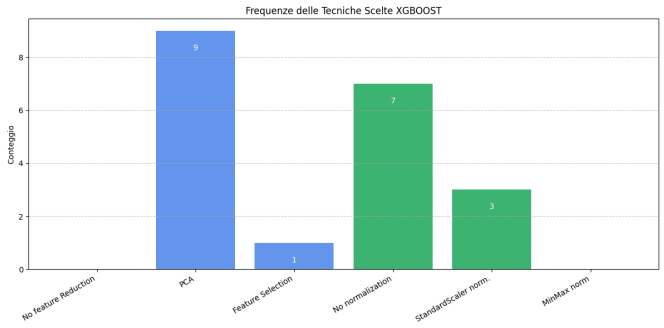


Osservando che

- Normalizzazione: la scelta è abbastanza bilanciata.
- Per la feature reduction è molto più frequente l'utilizzo della PCA rispetto sia alla feature selection che al non applicare nessuna tecnica.

Separando questi dati per i due modelli più performanti, si sono ottenuti i seguenti grafici:

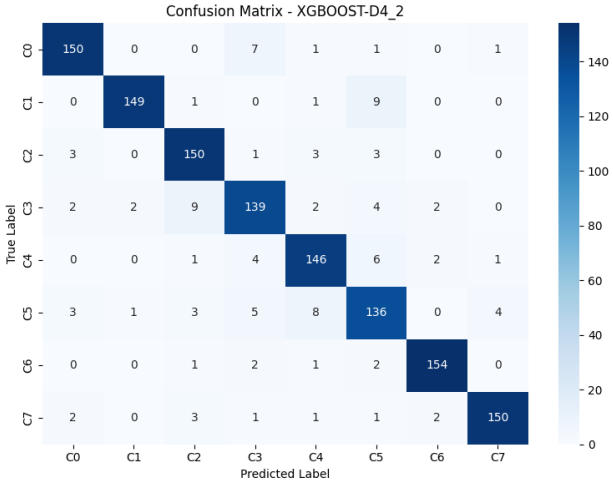




Da cui si evince che:

- SVM beneficia in generale della normalizzazione (nessuna preferenza) e che la PCA non viene mai applicata.
- XGBoost preferisce in genere l'assenza di normalizzazione e l'applicazione della PCA.

Successivamente si sono analizzate le matrici di confusione delle 5 migliori configurazioni:



(Esempio di matrice di confusione)

Notando, per tutte le 5 matrici, che

- I valori sulla diagonale sono prossimi al totale delle immagini per classe (160).
- Gli errori dei modelli non sono simmetrici, rendendo quindi inutile un approccio di tipo gerarchico nell'addestramento.

5.2 Osservazioni sulla segmentazione

Per verificare l'efficacia della segmentazione, si sono applicate le migliori configurazioni su dataset generati da immagini non segmentate.

I risultati riportati in 4.2 potrebbero far pensare che la segmentazione non sia fondamentale agli algoritmi di Machine Learning ai fini della classificazione. Per verficare ciò, si sono applicate le migliori configurazioni anche su dataset generati da immagini con solamente lo sfondo segmentato.

I risultati riportati in 4.3 indicano invece che il Machine Learning riesce nel task di classificazione basandosi esclusivamente sullo sfondo. Questo è causato, a posteriore di un'analisi delle immagini, all'omogeneità intra-classe dello sfondo.

Si è concluso, dunque, che gli ottimi risultati anche in assenza di segmentazione, potrebbero essere galvanizzati dall'utilizzo improprio delle feature estratte dallo sfondo.

5.3 Osservazioni sull'unsupervised

Per quanto riguarda i risultati ottenuti con metodi unsupervised (4.4), si sono fatte le seguenti osservazioni:

- Sebbene il K-Means sia risultato il migliore 9 volte su 10, le performance dei due modelli sono equiparabili.
- La PCA è sempre preferita, mentre la normalizzazione non viene mai applicata.
- Le performance si assestano attorno al 0.47 (Silhouette Score) dalla tipologia di dataset D2 in poi.

6 Conclusioni

Si è concluso quindi che la segmentazione implementata dal lato di Image Process & Analysis è necessaria per ottenere un algoritmo in grado di generalizzare sulla base

delle informazioni contenute esclusivamente nella foglia.

Dal lato Machine Learning si è concluso che i metodi supervised sono nettamente superiori, come da previsione, rispetto a quelli unsupervised.

La migliore prestazione ottenuta con il metodo ensamble si attesta al 97.42%, risultando quindi equiparabile alle prestazioni ottenute nella letteratura di riferimento[14].

[14] Reference link: le prestazioni sono ottenute sull'intero dataset di partenza