**ANALISI SPETTRALE E CLASSIFICAZIONE DELLE SPECIE ARBOREE CON TECNICHE DI IMAGE PROCESSING E MACHINE LEARNING SU IMMAGINI SATELLITARI**

**Studenti:** Saponaro Claudio, Mattiace Alessio

**INTRODUZIONE**

Negli ultimi anni, l'utilizzo delle immagini satellitari per l'analisi ambientale ha registrato un notevole incremento grazie ai progressi tecnologici nel campo del telerilevamento e della loro elaborazione; il nostro progetto si propone di approfondirne i concetti chiave ed applicarli ad uno specifico caso: quello della classificazione degli ulivi sul territorio Pugliese.

Il nostro studio, partendo proprio da immagini satellitari ad alta risoluzione, tenta di integrare delle tecniche di elaborazione con modelli di machine learning, offrendo una soluzione automatizzata per la classificazione delle specie arboree; tutti i codici e le funzioni utili ai nostri scopi, verranno scritti nel linguaggio di programmazione Matlab.

Gli obiettivi principali sono:

* In primis, creare un database sufficientemente informativo e completo che, con delle features mirate, permetta all’algoritmo di discriminare al meglio tra le varie specie di ulivi.
* Successivamente utilizzare diversi modelli di Machine Learning in grado di effettuare la vera e propria classificazione delle specie di alberi partendo dal dataset fornito.
* Infine, confrontare l’efficacia dei sopracitati modelli, valutando le performance in termini di accuratezza, precisione e robustezza.

I risultati ottenuti potrebbero avere interessanti implicazioni per il monitoraggio ecologico e la gestione sostenibile delle risorse forestali, supportandone le decisioni nella pianificazione; inoltre, migliorerebbe la nostra capacità di monitorare la biodiversità, rilevare cambiamenti ecologici e identificare specie a rischio.

**STATO DELL’ARTE**

Come già accennato nell’introduzione, negli ultimi anni abbiamo assistito a notevoli progressi nel mondo del telerilevamento, con l’introduzione di nuove tecniche e tecnologie che hanno migliorato l’accuratezza e l’affidabilità con la quale effettuare stime sulla vegetazione.

Nell’analisi della letteratura scientifica abbiamo individuato diversi strumenti e componenti hardware adatti allo scatto di immagini multispettrali; ad esempio, in [1] vengono usate immagini multispettrali satellitari QUICKBIRD e IKONOS (tramite satellite IKONOS-2).

Un’alternativa potrebbe essere quella esaminata in [2] e [3] dove il detecting delle chiome non viene supportato da immagini ottenute da satellite, bensì da un UAV Italdron 4HSE EVO (drone multi-rotore) da un’altezza di circa 70 metri su cui sono montate diverse fotocamere:

* Fotocamera Multispettrale a cinque bande MicaSense RedEdge-M.
* Fotocamera Termica FLIR Vue Pro 640 (per la cattura di immagini termiche ad alta risoluzione).
* Fotocamera Visibile ad Alta Risoluzione Sony α7r.

Questo tipo di strumentazione è particolarmente efficace nel bilanciare la qualità delle immagini (risoluzione) e l'efficienza del volo (resa dei rilievi aerei), considerando le caratteristiche specifiche degli oliveti.

Per quanto riguarda invece la segmentazione delle chiome, diverse ricerche come [4], propongono di eseguire una trasformata di Hough circolare per via della loro forma simile ad una circonferenza, oppure l’algoritmo di K-Means (dove le chiome saranno i clusters); Da osservare che nel nostro caso non è stato necessario effettuare questo ulteriore step poiché ci sono state già fornite immagini con la segmentazione delle chiome da usare come maschera binaria sulle immagini satellitari.

Infine, [5] e [6] mostrano come utilizzare (e con quali risultati), gli algoritmi di Machine Learning utili alla classificazione delle specie di ulivi; di seguito una lista dei più citati in letteratura:

* Logistic Regression
* Support Vector Machines (SVM)
* Random Forest
* Linear Discriminant Analysis (LDA)
* Neural Networks

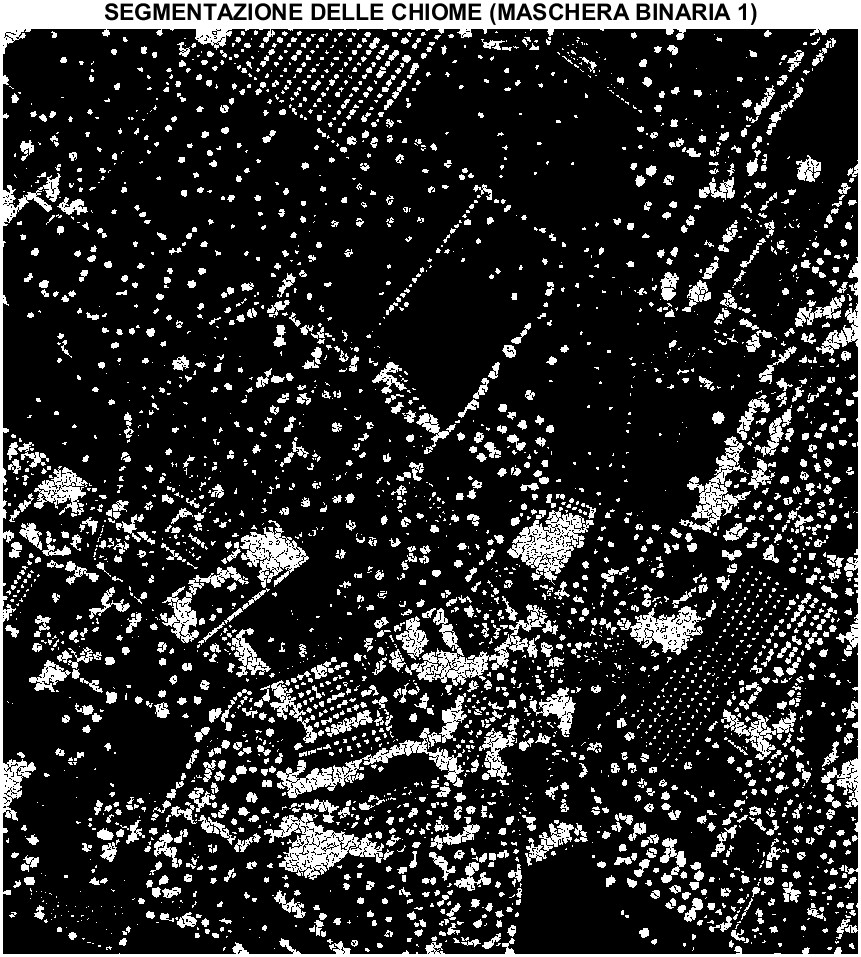
**SEZIONE 1 – CREAZIONE DEL DATABASE**

Il primo step è quello di ottenere un database da usare per training e test; esso dovrà contenere, per ogni chioma, informazioni sul valore medio dei suoi pixel, per ognuna delle 47 bande; questo vuol dire avere uno spettro medio per ogni albero e poterlo usare in seguito, per la classificazione.

**Materiale fornito**

Per raggiungere il nostro scopo, a cui abbiamo accennato nell’introduzione, possiamo contare sul seguente materiale:

* due immagini satellitari multispettrali ad alta risoluzione con 47 differenti bande in formato .TIF di campagne Pugliesi contenenti ciascuna migliaia di chiome.
* Le rispettive maschere binarie contenenti tutte le chiome arboree già segmentate (da utilizzare per isolare l’ulivo dallo sfondo).



* I rispettivi database in formato Excel che includono alcune centinaia di alberi con la loro posizione in coordinate geografiche, già classificati con alcune delle tipologie di ulivo presenti sul territorio Pugliese (come “Leccino”, “Ogliarola Barese”, ecc.); i dati sono raccolti da operatori direttamente sul campo.



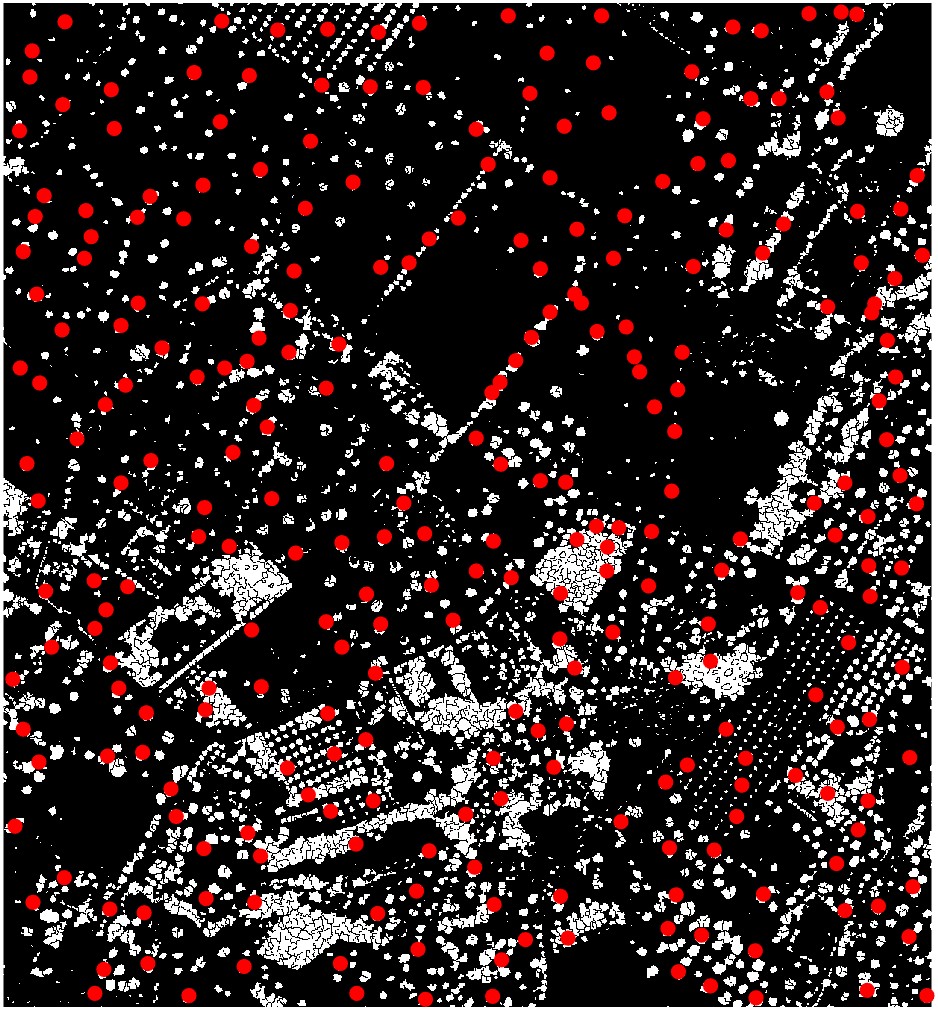
**Importazione del Database**

Per prima cosa utilizzo il comando “readtable” di Matlab per leggere il database in formato Excel già fornito ed elimino le righe (o samples) che contengono dati mancanti, successivamente assegno a delle variabili i valori di latitudine, longitudine e coltivazione.

**Trasformazione in coordinate intrinseche**

Per mappare le coordinate fornite dal database in punti dell’immagine in cui vi è presente una chioma, trasformiamo le coordinate geografiche in coordinate intrinseche dell’immagine georeferenziata: viene fatto effettuando una proiezione tramite il comando “worldToIntrinsic”.

Ora abbiamo una matrice “points” contenente le coordinate delle chiome presenti e classificate nel nostro database; di seguito una visualizzazione della loro distribuzione rispetto al totale degli alberi presenti in un crop:



OSSERVAZIONE – Essendo un database costruito tramite dati ottenuti in modo manuale da operatori sul campo, le coordinate di alcuni alberi risultano non coincidenti con una specifica chioma nell’immagine, bensì sono sullo sfondo: in questi casi il data sample andrà perduto e la chioma non considerata nel training e nel test.

**ID chioma univoco**

Per trattare ogni chioma della nostra immagine in modo univoco, necessitiamo di un ID da conferire ad ognuna di esse; quindi, dopo aver importato la maschera binaria andiamo ad usare il comando Matlab “bwlabel” che va ad etichettare ogni cluster dell’immagine segmentata ottenendo una matrice “L” con valore ‘0’ per le celle dello sfondo e un ID numerico per le celle dove sono presenti ulivi; inoltre, in “num” avremo il numero totale di essi che è nell’ordine delle migliaia.

Successivamente eseguo un semplice ciclo for per tener conto, in un vettore chiamato “id\_chiome\_db”, degli ID dei soli ulivi georeferenziati nel Database (e quindi utili alla classificazione).

**Calcolo spettro medio**

A questo punto, il nostro intento è quello di creare una matrice con 47 colonne e tante righe quanti alberi, in modo da contenere, per ogni riga (corrispondente alla singola chioma), lo spettro medio; per prima cosa, quindi, importiamo l’immagine multispettrale a 47 bande su cui effettuare le varie elaborazioni.

Successivamente creiamo un ciclo for che scorra tutte le chiome del nostro database in base al loro ID e, per ognuna delle 47 bande, andiamo per prima cosa a ‘mascherare’ l’immagine multispettrale eliminando lo sfondo; di seguito una visualizzazione del risultato di questo processo.

|  |
| --- |
| Immagine che contiene schermata, stella, costellazione, spazio  Descrizione generata automaticamente |

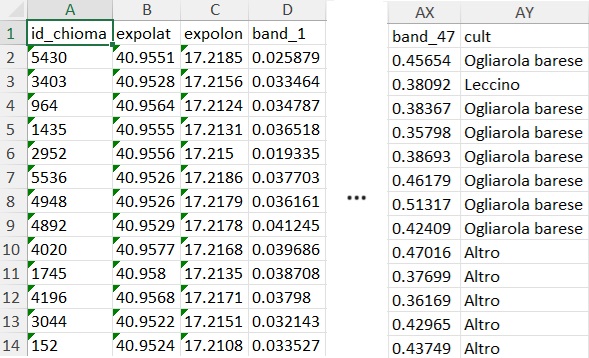
Poi, andiamo a calcolare l’array “Values” che contiene tutti i valori dei pixel di una singola chioma per una specifica banda: su questo andremo a ricavare il valore medio eliminando gli outliers.

OSSERVAZIONE – Consideriamo outliers tutti i valori che si allontanano dalla media più di una certa soglia (nel nostro caso la soglia è due volte la deviazione standard)

A questo punto utilizzeremo questo valore medio per riempire la cella (i, banda) della matrice “Firma\_spettrale\_media” dove “i” è la specifica chioma.

**Aggiornamento del Database**

Per concludere il nostro lavoro di creazione del database da utilizzare per il successivo step di classificazione delle specie arboree, vogliamo esportare (utilizzando Matlab) un file Excel che contenga, oltre le informazioni di latitudine, longitudine e coltivazione, anche lo spettro medio delle chiome e un ID univoco; quello che vogliamo ottenere, quindi, è un file strutturato in questo modo:

****

Lo facciamo concatenando orizzontalmente i vari vettori contenenti le informazioni che ci servono per costruirlo (rispettivamente ID, latitudine, longitudine, banda 1, banda 2, …, banda 47, coltivazione) e associando i dati alle rispettive labels con “array2table”; infine usiamo il comando “writetable” per esportare la tabella creata in un file Excel nella directory corrente.

**SEZIONE 2 – MODELLI DI CLASSIFICAZIONE**

Dopo aver concluso la sezione relativa alla creazione della tabella partendo dalle coordinate di latitudine e longitudine degli ulivi risulta necessario classificarli utilizzando appositi algoritmi di apprendimento automatico.

A tal proposito sono stati utilizzati tre algoritmi (in diverse configurazioni) per effettuare la classificazione:

1. SVM (Support Vector Machine)

2. RF (Random Forest)

3. LDA (Linear Discriminant Analysis)

Prima di spiegare in modo dettagliato il funzionamento e la logica di ognuno di questi algoritmi e, quindi, passare all’applicazione degli stessi sono stati effettuati dei passaggi preliminari di preprocessing del dataset a disposizione:

* suddividere il dataset in input (X) e output (Y).
* normalizzare i dati in X usando la z-score normalization.
* partizionare il dataset usando la strategia di ‘Holdout’ che prevede di dividerlo rispettivamente in training set e test set; in particolare la divisione è stata fatta considerando l’80% dei data points come training set e il restante 20% come test set (preservando la proporzione tra le varie classi usando la stratificazione).

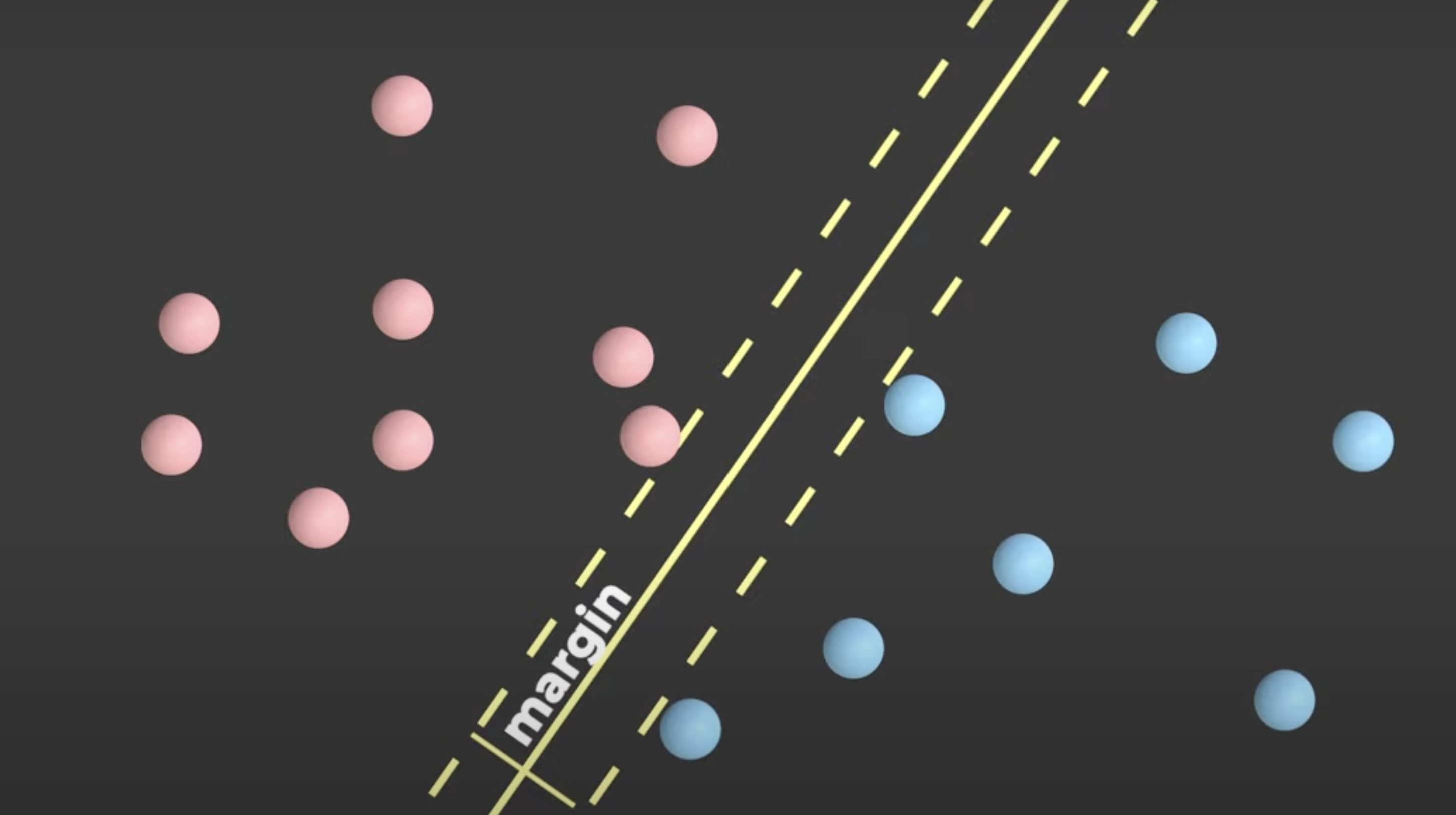
Di seguito viene fornita una spiegazione del funzionamento dei tre modelli in questione:

**SVM**

Tale algoritmo è descritto nel dettaglio nel paper [7].

In estrema sintesi l’idea proposta nel paper è quella di rappresentare i dati in uno spazio di dimensione pari al numero delle features e provare a classificarli trovando un iperpiano separatore che massimizzi la distanza tra le varie classi di dati; questa distanza è delimitata dai margini: linee parallele all’iperpiano.

Se i dati sono linearmente separabili, siamo in grado di trovare un iperpiano che non permette ai samples di essere mal classificati, in caso contrario potremmo usare dei soft margins oppure il Kernel Trick (dove si proiettano i dati in uno spazio a dimensione più alta prima di essere ‘separati’).



Tale figura mostra l’idea dell’algoritmo, possiamo visualizzare i vari dati appartenenti a due diverse classi (pallini rossi e blu), l’iperpiano separatore (linea gialla continua), i margini (linee gialle tratteggiate) e i vettori di supporto (pallino rosso e pallino blu giacenti sui margini).

Tale algoritmo è stato implementato in due varianti utilizzando la libreria offerta da Matlab: quella tradizionale, e l’SVM che sfrutta il kernel trick (per la precisione il kernel gaussiano).

**Random Forest**

Il secondo modello utilizzato è stato il Random Forest che rappresenta un’estensione del classification tree, superandone le limitazioni prestazionali.

Nell’algoritmo random forest vengono generati più alberi di classificazione per evitare l’overfitting (ovvero l’incapacità di generalizzare la previsione su nuovi dati) provocato dall’uso di un solo albero di classificazione: al tal proposito prende il nome di ‘forest’.

Il termine ‘random’ deriva dal fatto che i ‘bootstrap’ datasets (ovvero i dataset costruiti a partire dal dataset originale) su cui sono addestrati gli altri alberi di classificazione ottenuti casualmente effettuando un random sampling con reinserimento dal dataset originale.

I dati non presenti nel bootstrap dataset vengono utilizzati come validation dall’albero di classificazione addestrato sul bootstrap corrente.

Tale algoritmo viene spiegato nel dettaglio in [8].

Immagine che contiene schermata, Policromia, tabellone

Descrizione generata automaticamente

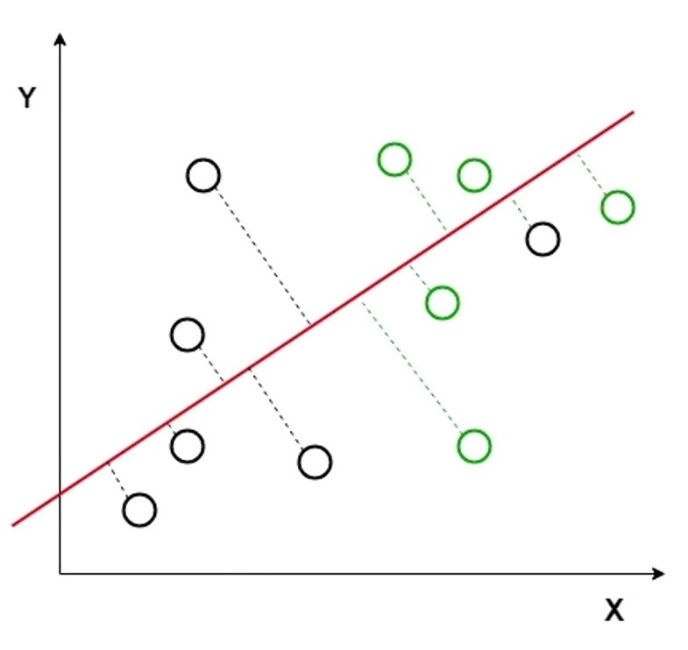
Tale figura chiarisce il processo di creazione dei bootstrap datasets su cui sono addestrati i vari alberi di classificazione.

Il numero di alberi di classificazione da utilizzare per creare la foresta casuale viene scelta a priori dall’utente mediante cross-validation per evidenziare la configurazione ottimale della classificazione finale.

**LDA**

Tale algoritmo è stato teorizzato ed è spiegato nel dettaglio in [9].

A differenza dei precedenti algoritmi LDA si concentra sul trovare un nuovo piano dimensionalmente inferiore rispetto a quello iniziale, in cui risulta più semplice effettuare la classificazione; il termine ‘linear’ fa riferimento al tipo di classificatore che separa le classi (esso è lineare esattamente come nel SVM senza l’uso del kernel trick).



Il termine ‘discriminant’ fa proprio riferimento alla funzione del discriminante per la risoluzione dell’algoritmo, che è la modalità con cui esso cerca di trovare una combinazione lineare delle caratteristiche che massimizzi la separazione tra le classi; questo viene fatto massimizzando la varianza interclasse e minimizzando la varianza intraclasse.

Noi abbiamo utilizzato tre varianti:

* discriminante lineare (metodo standard che assume che la variabilità dei dati all’interno di ciascuna classe sia uguale in ogni direzione).
* discriminante diagonale (in cui viene assunto che la matrice di covarianza delle classi sia diagonale).
* discriminante pseudolineare che si colloca come compromesso degli approcci precedenti.

**RISULTATI**

Per concludere il nostro lavoro, abbiamo valutato l’output (o predizioni) dei modelli precedentemente descritti con alcune metriche per la classificazione:

**Accuracy:** proporzione tra previsioni corrette sul totale delle previsioni.

**Precision:** proporzione tra veri positivi su tutti i casi che il modello ritiene positivi - ci suggerisce quanto il modello sia in grado di non etichettare positive le classi negative.

**Recall:** proporzione tra i dati classificati correttamente positivi e tutti i dati realmente positivi - ci suggerisce quanto il modello sia in grado di classificare le istanze positive.

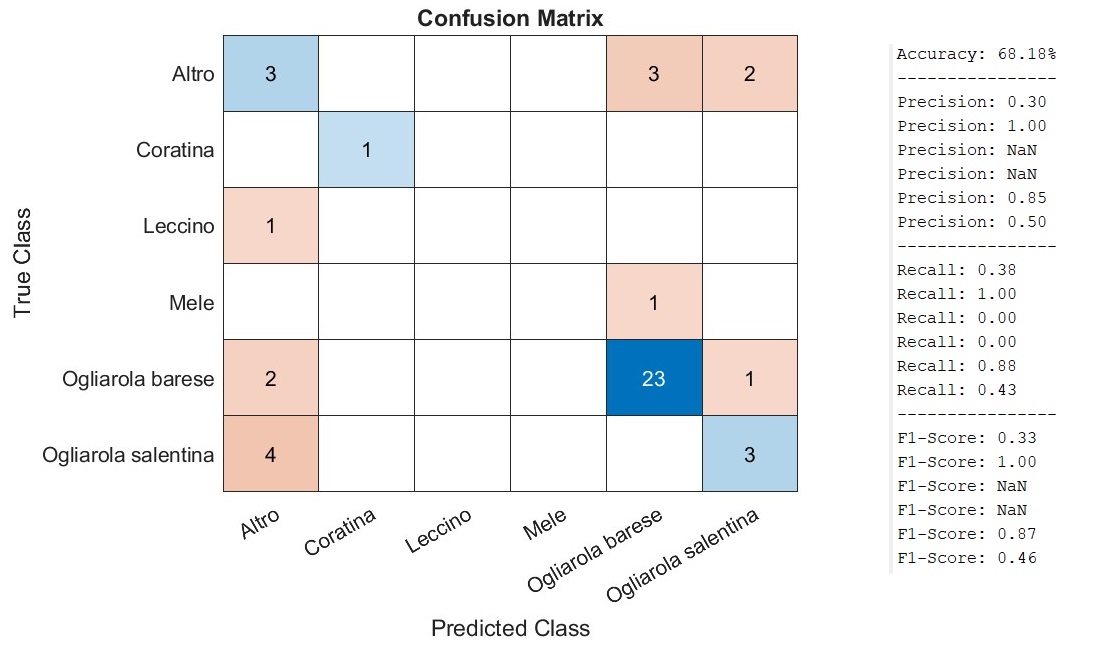
**F1-Score:** formula che tiene in considerazione sia recall che precision in una sorta di media armonica.

Di seguito vi sono riportati i risultati per ogni modello:

OSSERVAZIONE - i valori NaN, all’interno delle metriche, corrispondono ad un’assenza della specifica classe tra i data samples nel test set utilizzato, di conseguenza il modello non riesce ad usarle per la classificazione.

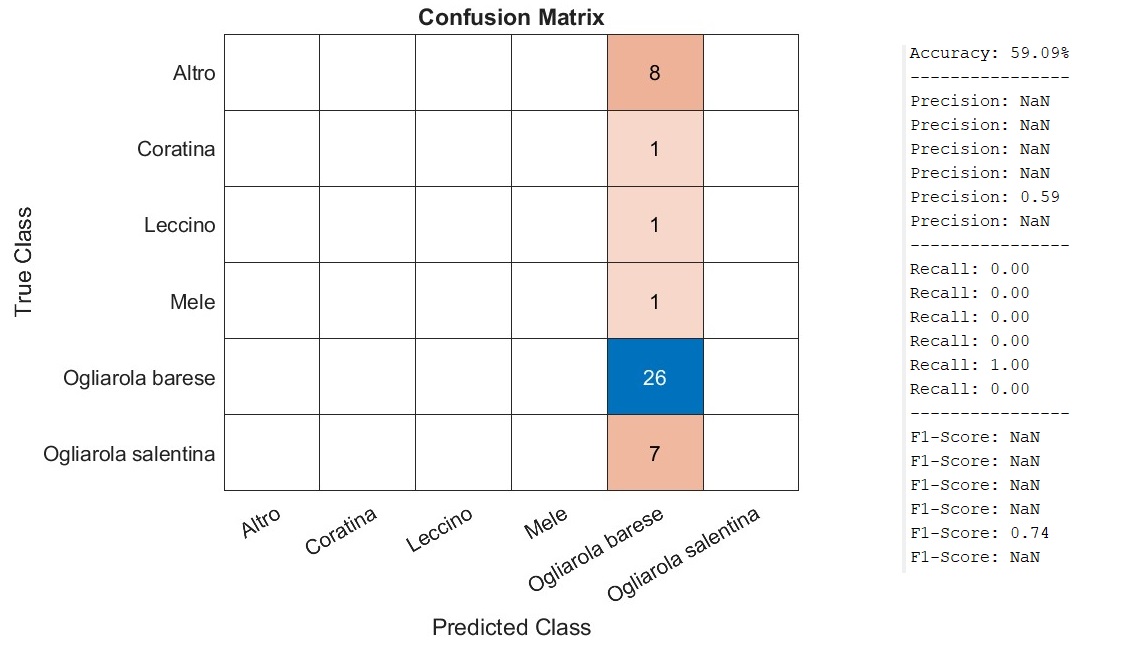
OSSERVAZIONE – ogni classe avrà un proprio valore di precision, recall ed F1-Score, mentre l’accuracy è un unico valore.

**SVM con Kernel lineare**

****

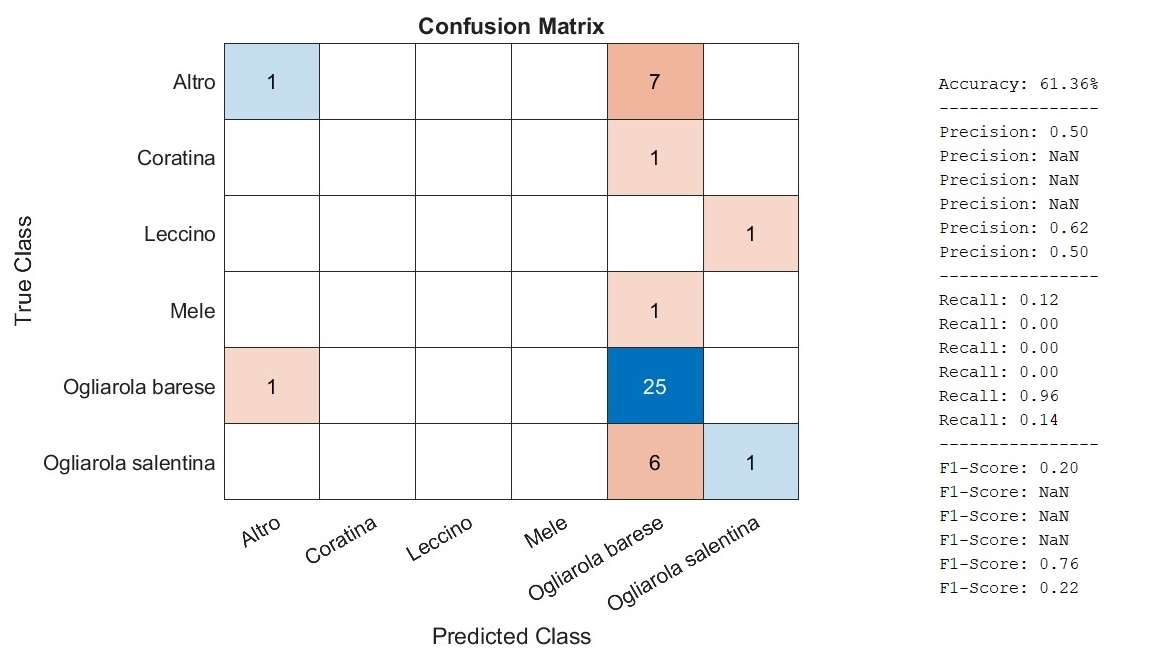
---------------------------------------------------------------------------------------------------------------

**SVM con Kernel gaussiano**

****

---------------------------------------------------------------------------------------------------------------

**Random Forest**

****

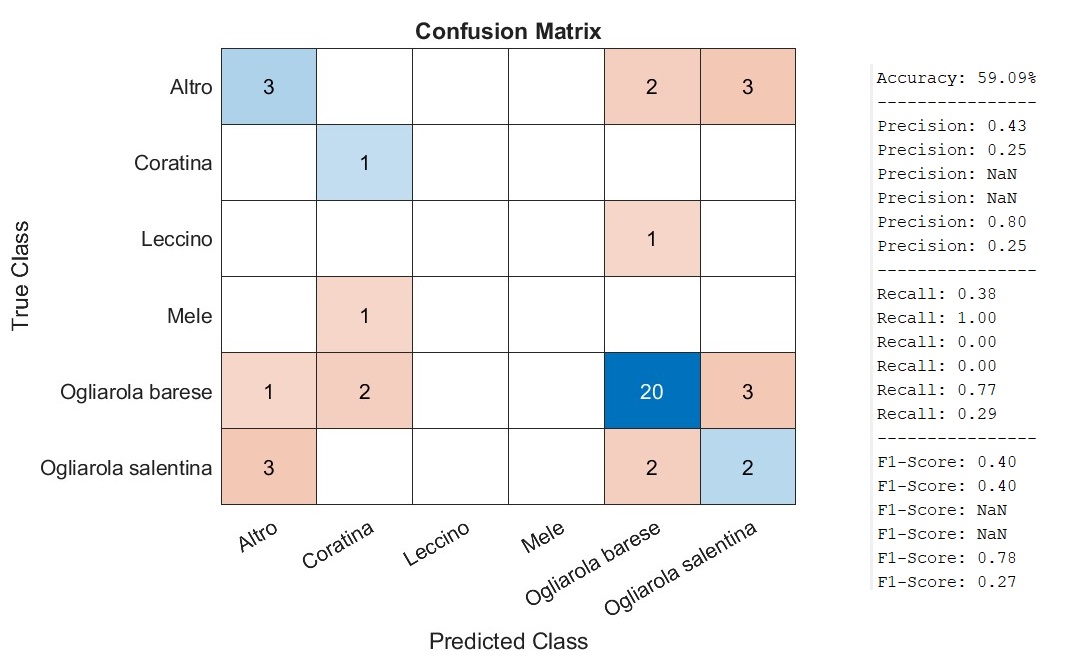
---------------------------------------------------------------------------------------------------------------

**LDA con Discriminant Type linear**

****

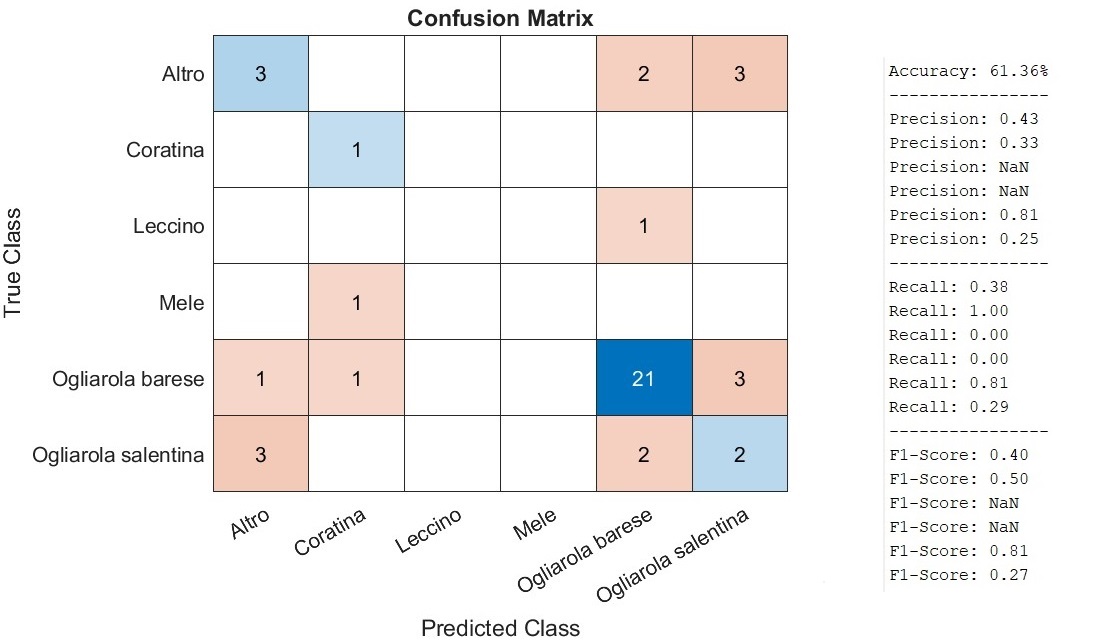
---------------------------------------------------------------------------------------------------------------

**LDA con Discriminant Type diaglinear**

****

---------------------------------------------------------------------------------------------------------------

**LDA con Discriminant Type pseudolinear**

****

Notiamo che i modelli migliori sono l’SVM con Kernel lineare e l’LDA con discriminant type lineare che raggiungono un’accuracy intorno al 70%, tutti gli altri da noi testati invece, come il Random forest, performano peggio con un’accuracy intorno al 60%; il peggiore risulta essere l’SVM con kernel gaussiano poiché non è in grado di classificare correttamente nessuno degli ulivi all’infuori dell’Ogliarola barese.

**RIFERIMENTI**

[1] M. Waleed, T. -W. Um, A. Khan and Z. Ahmad, "An Automated Method for Detection and Enumeration of Olive Trees Through Remote Sensing," in IEEE Access, vol. 8, pp. 108592-108601, 2020, doi: 10.1109/ACCESS.2020.2999078.

[2] «Sensors | Free Full-Text | Fast Detection of Olive Trees Affected by Xylella Fastidiosa from UAVs Using Multispectral Imaging». Consultato: 8 marzo 2024. [Online]. Disponibile su: <https://www.mdpi.com/1424-8220/20/17/4915>

[3] F. Adamo, F. Attivissimo, A. Di Nisio, M. A. Ragolia and M. Scarpetta, "A New Processing Method to Segment Olive Trees and Detect Xylella Fastidiosa in UAVs Multispectral Images," 2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Glasgow, United Kingdom, 2021, pp. 1-6, doi: 10.1109/I2MTC50364.2021.9459835.

[4] A. Khan et al., "Remote Sensing: An Automated Methodology for Olive Tree Detection and Counting in Satellite Images," in IEEE Access, vol. 6, pp. 77816-77828, 2018, doi: 10.1109/ACCESS.2018.2884199.

[5] M. S. Mandava, D. Jadhav and R. R. Naik, "Fault classification using SVM," 2015 IEEE International Circuits and Systems Symposium (ICSyS), Langkawi, Malaysia, 2015, pp. 17-21, doi: 10.1109/CircuitsAndSystems.2015.7394056.

[6] G. Ramat et al, "Mapping of olivetrees using Sentinel-2 and Sentinel-1 images: an assessation of pixel-based analyses," 2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor), Pisa, Italia, 2023, pp. 263-267, doi: 10.1109/MetroAgriFor58484.2023.10424313.

[7] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt and B. Scholkopf, "Support vector machines," in IEEE Intelligent Systems and their Applications, vol. 13, no. 4, pp. 18-28, July-Aug. 1998, doi: 10.1109/5254.708428.

[8] Cutler, Adele & Cutler, David & Stevens, John. (2011). Random Forests. 10.1007/978-1-4419-9326-7\_5.

[9] Fisher, R.A. (1936) The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics, 7, 179-188. http://dx.doi.org/10.1111/j.1469-1809.1936.tb02137.x