Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica

Automatic classification of healthy / diseased plants using multispectral images

Relatore
Prof. Maurizio Morisio

Candidato
Stefano Pavone

Correlatore
Prof. Luca Ardito

Anno Accademico 2021 - 2022

Alla mia famiglia, a mia nonna, ai miei amici e a me.

Indice

Introduzione

In agricoltura, al crescere dell'estensione delle piantagioni, aumenta la complessità nel valutare lo stato di salute della singola pianta.
Questo avviene perché attualmente la valutazione dello stato fisiopatologico di esse si basa su controlli visivi diretti, seguiti eventualmente da indagini di laboratorio, caratterizzati da processi dispendiosi, non solo in termini di tempo, ma anche in termini economici.
Questi lunghi processi possono comportare, oltre agli elevati costi delle analisi strumentali, una mancanza in termini di reattività decisionale, che nel caso ad esempio, di malattie trasmissibili, può portare alla perdita di più esemplari.
Tutti questi processi risultano di difficile applicazione, specie quando una coltivazione è composta da moltissimi esemplari.

La tesi in questione porta avanti il progetto DroNuts, che si pone come obiettivo quello di costruire un'infrastruttura sia hardware che software, in grado di fornire tramite servizi di telerilevamento, una classificazione automatizzata dello stato di salute di ogni singola pianta.
Sulla base di immagini multispettrali, raccolte in una precedente fase tramite l'uso di un apposito drone, si andranno ad approfondire ed analizzare diversi approcci per la realizzazione di un software in grado, tramite l'analisi e l'elaborazione delle riprese, di classificare autonomamente lo stato di salute di una pianta di nocciolo.
Si andrà inizialmente a fare un rapida discussione di cosa sono le immagini multispettrali, di come esse vengano acquisite e quale tipologia di informazioni apportano in ambito agricolo.
Insieme ad esse si andranno ad accennare brevemente anche i sensori iperspettrali. Anche se essi non sono stati impiegati in questa fase progettuale, si reputa interessante la loro conoscenza ed importante un eventuale loro futuro utilizzo.

Successivamente, nel *Capitolo 2* si andranno ad analizzare gli Indici Vegetazionali, strumento indispensabile per la realizzazione di questo progetto e per lo studio delle condizioni di salute di qualsiasi coltivazione.
Si darà una loro generica descrizione ed in seguito si approfondirà quali indici sono stati utilizzati in particolare e come.
Vedremo che da essi sarà possibile estrarre alcune metriche, che descrivono numericamente le caratteristiche della pianta su cui è stato calcolato il particolare indice.
Sulle le suddette metriche, si costruiranno ed addestreranno degli algoritmi di machine learning che permetteranno una classificazione della pianta in sana o malata.
Ovviamente prima di ciò, nel *Capitolo 3,* si approfondiranno le caratteristiche del dataset, ed in particolare delle foto a disposizione, andandone ad evidenziare i difetti, le criticità e cercando degli approcci che possano porre rimedio.
In seguito nel *Capitolo 6,* si discuterà degli algoritmi applicati, del loro addestramento, ed ovviamente delle performance da essi raggiunte.
Infine, nel *Capitolo 7*, si esplorerà un diverso approccio, non più basato sugli indici vegetazionali, ma bensì su reti neurali ed in particolare sulle *Convolutional Neural Networks* con in ingresso le immagini RGB.
Si confronterà quindi quest'ultimo approccio, con quello facente uso degli indici vegetazionali, ed in ultimo si mostreranno vantaggi e svantaggi di ognuno.
Nell'ultimo capitolo si cercherà di riassumere le fasi percorse, dare delle considerazioni finali e cercare di elencare un insieme di strategie volte al perfezionamento dei processi visti.

Capitolo 1

1.1 Immagini Multispettrali

Le immagini multispettrali vengono spesso utilizzate nell'ambito del *remote sensing* o del Telerilevamento.
Ambito tecnico-scientifico che si pone come obiettivo quello di ricavare informazioni su determinati oggetti posti a distanza da un sensore, mediante la misura della radiazione elettromagnetica emessa, riflessa o trasmessa, dalle loro superfici.
Nell'ambito del telerilevamento sono molti i mezzi che possono essere utilizzati per effettuare le riprese: aerei, satelliti, droni o addirittura sonde spaziali.
Generalmente, lo studio di una superficie effettuato con tecniche di telerilevamento avviene in tre fasi distinte:

1. Acquisizione dei dati mediante riprese effettuate con la necessaria strumentazione;
2. Elaborazione dei dati e successiva creazione delle immagini digitali;
3. Interpretazione e analisi dei dati acquisiti.

Nel nostro caso il mezzo tramite il quale sono state effettuate le rilevazioni è un drone.
La fase precedente del progetto ha infatti visto, come uno degli obiettivi più importanti, la scelta del drone e dei relativi sensori multispettrali.
In particolare, per la raccolta immagini è stato utilizzato un drone modello DJI P4 Multispectral.
Esso è dotato di sei sensori, aventi le seguenti caratteristiche:

Sensori	Sei sensori CMOS 1/2.9", incluso un sensore RGB per le immagini su luce visibile e cinque sensori monocromatici per l'acquisizione di immagini multispettrali. Ciascun sensore: Pixel effettivi 2,08 MP (2,12 MP in totale)
Filtri:	Blu (B): 450 nm ± 16 nm, verde (G): 560 nm ± 16 nm, rosso (R): 650 nm ± 16 nm, Red-Edge (RE): 730 nm ± 16 nm, vicino infrarosso (NIR): 840 nm ± 26 nm
Obiettivi	FOV (campo visivo): 62,7° Lunghezza focale: 5,74 mm (formato equivalente 35 mm: 40 mm), autofocus a ∞ Apertura: f/2.2
Intervallo ISO sensore RGB	200 – 800
Guadagno del sensore monocromatico	1 – 8x
Otturatore elettronico globale	1/100 – 1/20000 s (luce visibile); 1/100 – 1/10000 s (multispettrale)
Dimensione massima dell'immagine	1600×1300 (4:3.25)

Figura 1-1: Caratteristiche tecniche sensori

In particolare il sensore in dotazione è in grado di scattare foto RGB (*quindi spettro del visibile*), Red-Edge(RE) caratterizzato da una lunghezza d'onda di 730 nm ± 16nm e NIR (Near Infrared) ad una lunghezza d'onda di 840 nm ± 26nm.

Essi permettono quindi una misura, alle varie lunghezze d'onda elencate, della radiazione elettromagnetica riflessa dagli oggetti ripresi.

Ogni lunghezza d'onda apporterà tipologie di informazione differenti, e come vedremo a breve, la loro combinazione è ciò su cui si basano gli Indici Vegetazionali.

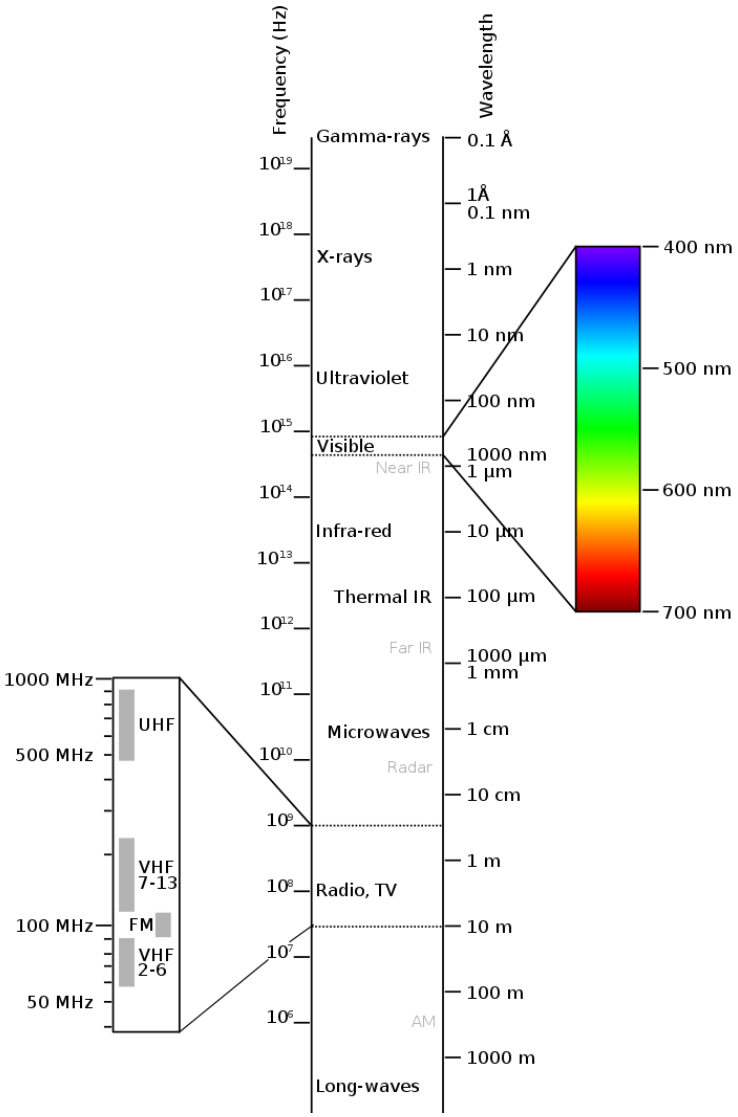


Figura 1-2: Spettro Elettromagnetico

1.2 Riflettanza spettrale vegetativa

Come detto in precedenza, oltre allo spettro del visibile, i sensori a nostra disposizione sono in grado di effettuare misurazioni anche ad altre due diverse lunghezze d'onda, denominate rispettivamente Red-Edge e NIR.

Il NIR (*Near Infrared)* è una regione dello spettro elettromagnetico che spazia da una lunghezza d'onda di 780 nm a 2500 nm.

Esso appartiene alla banda dell'infrarosso, ma allo stesso tempo è la regione più vicina, in termini di lunghezza d'onda, allo spettro del visibile.

Il NIR viene ampiamente utilizzato nell'analisi spettroscopica, che tramite l'osservazione della radiazione assorbita e riflessa da una superficie, permette di determinare le proprietà senza alcuna alterazione di essa.
Date le sue proprietà, vede ampio uso non solo nell'ambito agrario, ma anche nell'ambito della diagnostica medica e fisiologica, nello studio delle caratteristiche atmosferiche e nell'astronomia.
Il Red-Edge invece si riferisce ad un range dello spettro elettromagnetico tra il Rosso e il NIR, in cui è possibile osservare un rapido cambiamento nella proprietà di riflettanza della vegetazione.
Infatti, la clorofilla contenuta nella vegetazione assorbe la maggior parte della luce nello spettro del visibile (400 nm - 700 nm), ma la struttura cellulare delle foglie, invece, è caratterizzata da una forte riflettanza quando ci si spinge verso la banda del vicino infrarosso (700 nm- 1100 nm).

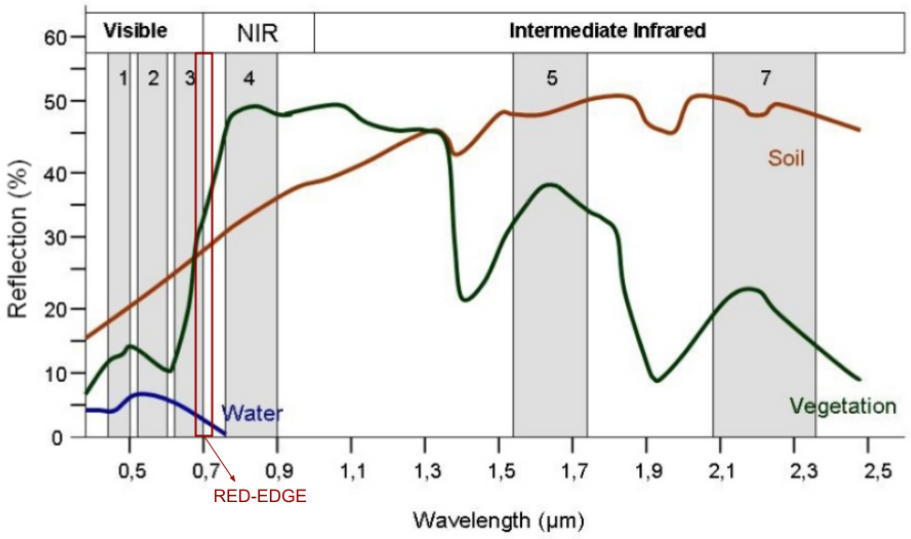


Figura 1-3: Firme spettrali di acqua, vegetazione e terreno

Nell'immagine soprastante possiamo vedere quelle che vengono definite come *Firme Spettrali.*
Si definisce *firma spettrale* la radiazione riflessa da una superficie, in funzione della lunghezza d'onda.

Notiamo da esse, come la vegetazione verde ed in salute sia caratterizzata da una bassa percentuale di luce riflessa nello spettro del visibile.
Infatti questa presenta una bassa riflettanza nelle regioni spettrali del blu e specialmente del rosso, che sono infatti i range più coinvolti nel processo di fotosintesi.
Mostra valori leggermente più alti nella range del verde *[e da qui il colore verde visibile più frequente nelle foglie, ndr]*.
La riflettanza però, inizia drasticamente a salire mano a mano che ci spinge dal rosso verso il NIR, mostrando come range di massima variazione proprio la banda denominata RED-EDGE.
Questo perché, nel corso del tempo, le cellule delle foglie si sono evolute per disperdere la radiazione solare nel range del NIR, in quanto il livello di energia per fotone in questo dominio non è sufficiente per la sintesi di molecole organiche e un suo assorbimento di essa comporterebbe semplicemente un riscaldamento eccessivo e la perdita di linfa per evaporazione[*4, esa.int, physicsopenlab.org*].
La radiazione solare che le piante utilizzano come fonte di energia nel processo di fotosintesi, prende il nome di *radiazione fotosinteticamente attiva* (*PAR - Photosynthetically active radiation).*
Quindi le piante appariranno piuttosto scure nelle immagini scattate nel range del PAR e relativamente luminose nel vicino infrarosso.

È interessante notare come, conoscendo la curva di riflettanza spettrale di vegetazione in salute sia possibile andare a individuare la vegetazione che invece risulta essere in sofferenza, in quanto essa sarà caratterizzata da percentuali di riflettanza più basse nella banda del NIR (e oltre), e percentuali più alte nel visibile.
Tramite lo stesso principio (*come si vedrà più approfonditamente discutendo degli indici vegetazionali),* è possibile discriminare dalla vegetazione, sia il terreno che l'eventuale presenza di acqua.

Infatti notiamo come, la curva di riflettanza dell'acqua risulti caratterizzata da un'altissima percentuale di assorbimento nel range dell'infrarosso e oltre.
Mentre quella del terreno privo di vegetazione presenta una curva molto più uniforme, non mostrando evidenti *spikes* ad alcun range spettrale.
Da attenzionare però che la firma spettrale del terreno è comunque soggetta a diverse sue caratteristiche, come ad esempio: il contenuto di umidità, la sua consistenza, le caratteristiche superficiali (roccioso, sabbioso, argilloso), l'eventuale presenza di ossido di ferro ed infine il materiale organico in esso presente.

1.3 Immagini iperspettrali

Nel paragrafo precedente abbiamo visto come e in che quantità le immagini multispettrali apportano informazioni.
Nonostante i sensori applicati nel progetto permettano esclusivamente l'acquisizione di immagini multispettrali, è importante sapere come, con la tecnologia oggi in commercio, è possibile ottenere una tipologia di informazione più accurata e ricca tramite l'utilizzo di sensori iperspettrali.
Essi si differenziano sia nel numero di bande che riescono a catturare, sia nella distanza tra ognuna di esse.
Un'immagine multispettrale generalmente cattura dalle 3 alle 10 bande tra cui sono quasi sempre inclusi i canali del rosso, del verde e del blu. A questi generalmente, come nel nostro caso, vengono aggiunti altre bande ben definite e che sono caratterizzate da un nome ben preciso, come ad esempio: Red-Edge, NIR, SWIR (*Short-wave infrared*).

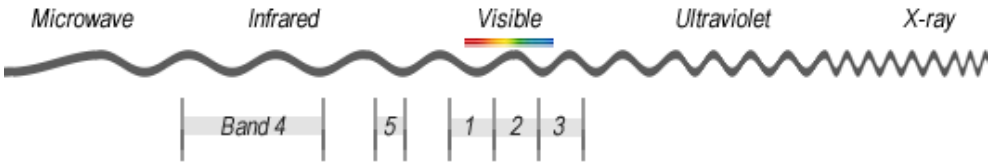


Figura 1-4: Esempio di canali inclusi in immagine multispettrale

Per quanto riguarda le immagini iperspettrali invece, non solo comprendono un maggior numero di bande, ma quest'ultime sono estremamente vicine tra di loro (*quasi continue)*.
Un'immagine iperspettrale può avere centinaia di bande, a cui in generale non è assegnato un nome.

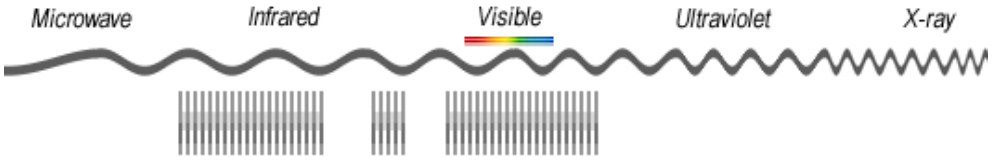


Figura 1-5: Esempio di canali inclusi in immagine iperspettrale

Nel caso di un'immagine iperspettrale, la firma spettrale risultante ovviamente sarà maggiormente continua rispetto a quella multispettrale che risulterà discretizzata, come è possibile notare nella *figura 1-6*.

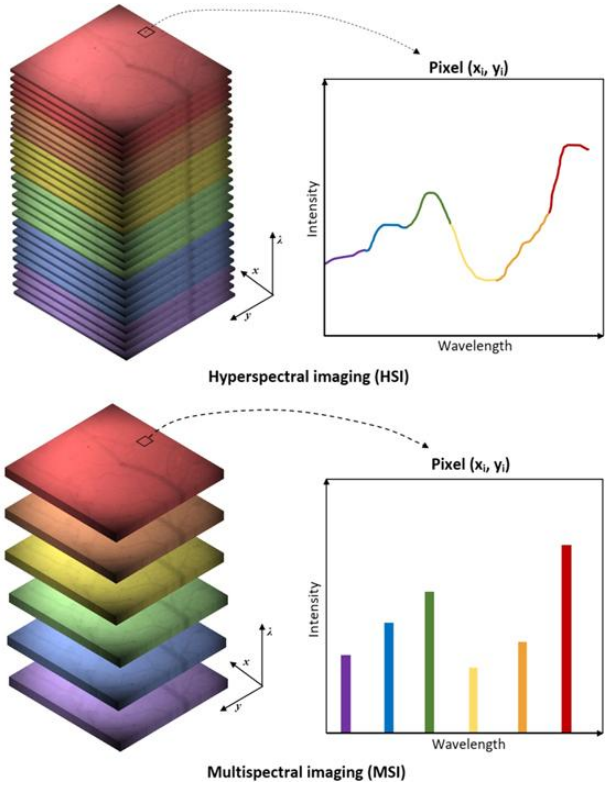


Figura 1-6: Differenza tra immagine iperspettrale (sopra) e multispettrale (sotto)

Questo tipo di analisi apporta sicuramente molte più informazioni rispetto all'analisi multispettrale, permettendo un maggior grado di discriminazione tra tipologie e stato del terreno, vegetazione sana/malata e addirittura tipologia e causa di stress della pianta.

Di contro però si possono evidenziare alcuni svantaggi di questo approccio come ad esempio:

- Ridondanza di informazioni tra le varie bande (*Riducibile tramite PCA);*
- Incremento in termini di dimensione (Byte) dei dati (*Si arriva facilmente a GB o TB di dati);*
- Costo dei sensori decisamente più elevato.

Tuttavia data la mole di informazioni utili che questo approccio apporta, esso dovrebbe in futuro essere preso in considerazione.

Capitolo 2

2.1 Indici Vegetazionali

Abbiamo visto quindi come lo studio della luce riflessa da un soggetto in esame, in diversi canali o range dello spettro elettromagnetico, apporti diverse tipologie di informazione.
Ogni pianta o coltivazione infatti, come mostrato precedentemente, è caratterizzata da una sua firma spettrale, che ne descrive caratteristiche e stato di salute.
Esiste tuttavia uno strumento in grado di enfatizzare determinate caratteristiche relative ad un oggetto preso in esame, combinando il quantitativo di radiazioni riflesse in diverse specifiche bande.
Esso prende il nome di *Indice Vegetazionale (VI - Vegetation Index).*
Si può definire l'Indice Vegetazionale come una trasformazione spettrale di una o più bande al fine non solo di evidenziare caratteristiche e proprietà della vegetazione, ma di permettere anche un'analisi comparativa, sia spaziale che temporale, dell'attività fotosintetica terrestre e delle variazioni strutturali nella vegetazione.
Esistono davvero molti Indici Vegetazionali (più di 150), ognuno di essi con funzionalità differenti, ma che spesso possono coincidere.
Ma soltanto un numero ridotto di essi è stato accuratamente testato e risulta avere solide basi bio-fisiche.
Molti di loro si basano sulla relazione inversa che vi è tra la riflettanza del canale rosso e quella della banda del NIR che caratterizza, come esposto nel capitolo precedente, la firma spettrale della vegetazione.

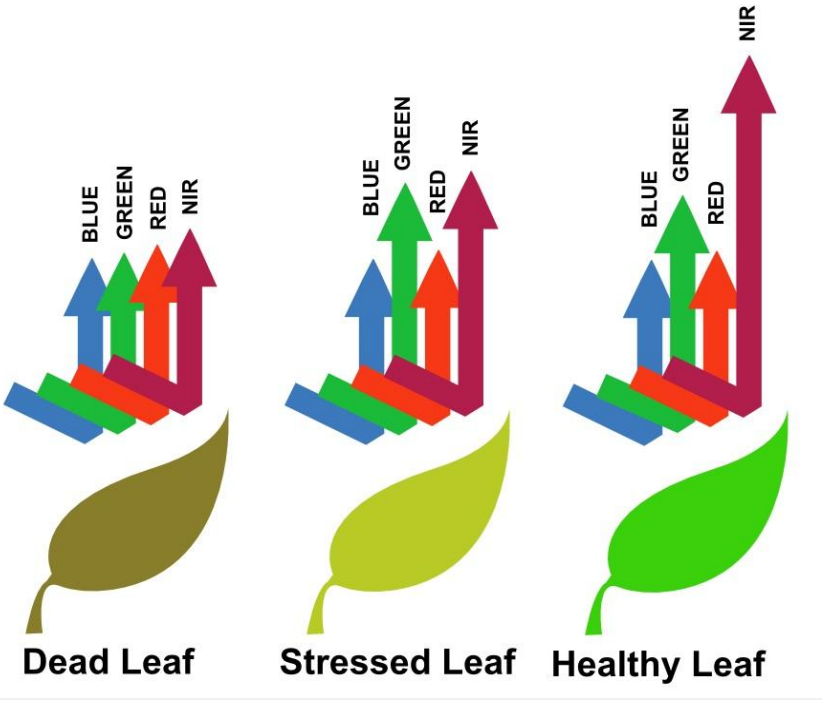


Figura 2-1: Comportamento spettrale al variare dello stato di salute della pianta

Questa prima fase ha quindi visto una profonda ricerca sugli indici vegetazionali in generale, ma in particolare su quelli applicabili con la tipologia di foto multispettrali a disposizione.
Ovviamente ci si è concentrati su quella tipologia di indici da cui fosse possibile derivare informazioni utili alla determinazione dello stato di salute della pianta.
Negli paragrafi successivi andremo quindi a vedere nel dettaglio gli indici vegetazionali che sono stati applicati sulle foto scattate alla piante di nocciolo prese in esame, calcolandone ed evidenziandone caratteristiche, punti di forza e di debolezza.

2.1.1 NDVI

Uno dei più famosi e forse più utilizzati indici vegetazionali è il *Normalized Difference Vegetation Index*.
Esso è un indicatore grafico del livello di vigore della coltura, e si basa su una differenza normalizzata, tra la luce riflessa nella banda del NIR e quella nella banda del RED.

Esso si calcola tramite la seguente formula:

$$NDVI = \frac{NIR - RED}{NIR + RED}$$

Formula 2-1: Calcolo indice NDVI

L'indice è definito in un range [-1, +1];

Valori strettamente negativi sono attribuibili ad acqua, neve, o nuvole.

Valori nell'intorno di 0 sono invece caratteristici del terreno, di rocce e sabbia.

Mentre i valori che appartengono al range [0.2, 0.3] possono caratterizzare aree con arbusti prevalentemente legnosi.

Più ci si spinge verso valori che superano lo 0.3, più l'indice è indicativo di aree verdi tipiche di alberi, foreste o campi.

Ovviamente, nel momento in cui esso viene applicato ad una coltura, valori al di sotto di una determinata soglia, segnalano una zona di essa in sofferenza.

Questo coerentemente al principio che, una pianta in salute assorbe energia nello spettro del visibile, ma la riflette in quella del NIR.

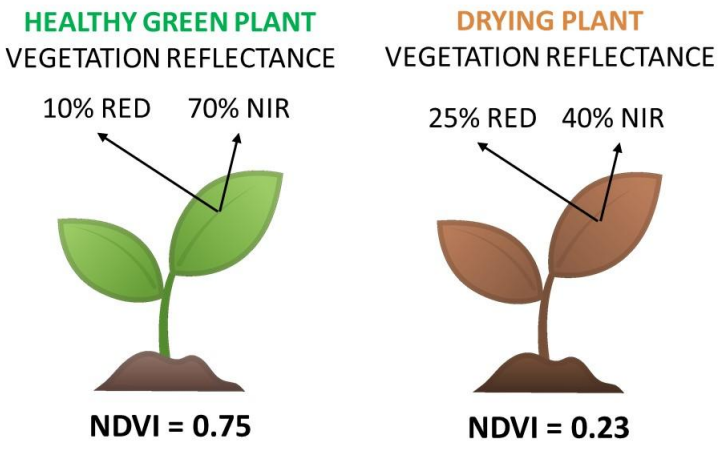


Figura 2-2: Variazione del NDVI in funzione dello stato di salute della pianta

Una generica classificazione dei valori che l'indice, in campi coltivati, può assumere è la seguente:

NDVI	Interpretazione
<0.1	Suolo nudo, nuvole o acqua
0.1 - 0.2	Copertura vegetale quasi assente
0.2 - 0.3	Copertura vegetale molto bassa
0.3 - 0.4	Copertura vegetale bassa con vigoria bassa, o copertura vegetale molto bassa con vigoria alta
0.4 - 0.5	Copertura vegetale medio-bassa con vigoria bassa o copertura vegetale molto bassa con vigoria alta
0.5 - 0.6	Copertura vegetale media con vigoria bassa o copertura vegetale medio-bassa con vigoria alta
0.6 - 0.7	Copertura vegetale medio-alta con vigoria bassa o copertura vegetale media con vigoria alta
0.7 - 0.8	Copertura vegetale alta con vigoria alta
0.8 - 0.9	Copertura vegetale molto alta con vigoria molto alta
0.9 - 1	Copertura vegetale totale con vigoria molto alta

Tabella 2-1: Interpretazione dell'indice NDVI in aree coltivate

Questi valori tuttavia sono indicativi, e variano molto dal tipo di coltura o vegetazione presa in esame e, specialmente, dalla stagione in cui essi vengono registrati.

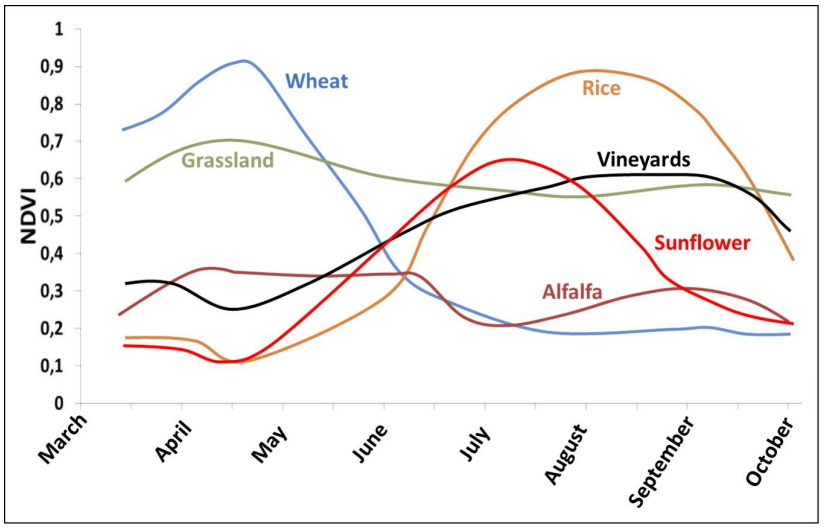


Figura 2-3: Variazione del NDVI in relazione al mese e alla tipologia di pianta

Per questo è importante sempre mettere in relazione i valori restituiti dall'indice con la stagionalità e la tipologia di coltura in esame.
Sotto vediamo un esempio del NDVI applicato ad uno dei noccioli presi in esame.

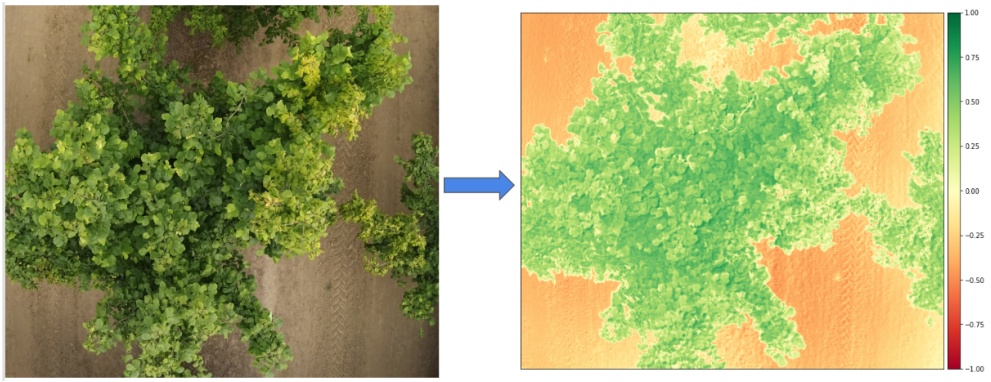


Figura 2-4: NDVI applicato ad una pianta di nocciolo

2.1.2 SAVI

L'indice SAVI (*Soil-Adjusted Vegetation Index)* è una variazione del sopra esposto NDVI.
Esso è stato pensato con l'intento di minimizzare l'influenza della luminosità dovuta alle diverse caratteristiche del suolo, tramite l'utilizzo di un apposito fattore di correzione *L.*

Si è infatti visto che i valori dell'indice NDVI possono risultare meno precisi in presenza di determinate caratteristiche del terreno, in particolare quando la copertura vegetativa è bassa.
L'indice si basa sulla seguente formula:

$$SAVI = \frac{(1+L)*(NIR - RED)}{(NIR + RED + L)}$$

Formula 2-2: Calcolo indice SAVI

Tramite il parametro L è possibile adattare l'indice al livello di copertura vegetativa presente e alle caratteristiche del suolo.
Esso può spaziare da 0 a 1. E si noti come con L = 0, si ottenga nuovamente la formula del NDVI.
Generalmente in aree con assenza di copertura vegetativa, L = 1;
In aree con moderata vegetazione, L = 0.5;
Mentre in aree molto coperte dalla vegetazione, L = 0 (*NDVI).*
Essendo anch'esso una differenza normalizzata, il range di valori a cui può appartenere il risultato è compreso in [-1, +1].
Fermo restando che è necessario testare diversi valori di L, scegliendo il migliore per la situazione in esame, un valore che spesso risulta essere un buon compromesso è L = 0.5.
Lo vediamo applicato alla pianta vista in precedenza, nella foto sottostante:

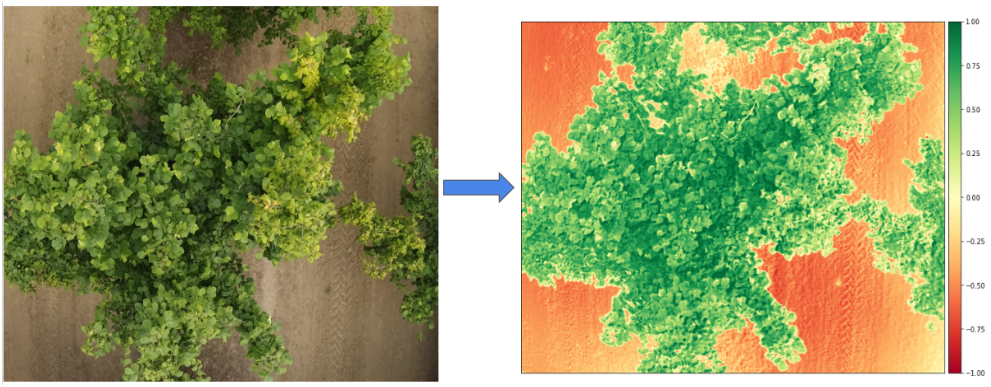


Figura 2-5: Indice SAVI con L = 0.5 applicato ad una pianta di nocciolo

2.1.3 GCI

Il GCI (*Green Chlorophyll Index*) è il terzo indice che si è deciso di approfondire e utilizzare.

Esso è utilizzato per stimare il contenuto di clorofilla nelle foglie delle piante, il quale risulta essere direttamente legato allo stato fisiologico della pianta.

Dato che esso decresce in piante con stress vegetativo, è un ottimo indicatore della condizione di salute di esse.

Esso si basa su due bande, NIR e la banda del Green nello spettro del visibile.

Il suo calcolo è basato sulla seguente formula:

$$GCI = \left(\frac{NIR}{GREEN}\right) - 1$$

Formula 2-3: Calcolo indice GCI

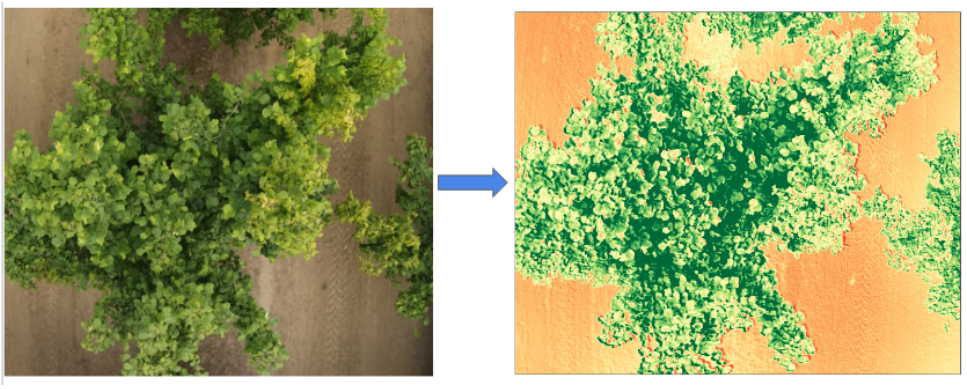


Figura 2-6: Indice GCI applicato ad una pianta di nocciolo

2.2 Metriche Calcolate

Come visto, gli Indici Vegetazionali, sono uno strumento molto utile in grado di mettere in risalto caratteristiche che l'occhio umano non è in grado di cogliere.

Essi però implicano che vi sia sempre la supervisione di un esperto in

grado di poter visualizzare l'immagine derivante dall'indice e dire se la pianta in esame risulta essere in sofferenza o meno.

Questo perché di per sé, gli Indici Vegetazionali sono uno strumento *grafico-visivo*. Il valore dell'indice in sé, è riferito al singolo pixel. Infatti è l'insieme dei valori dell'indice, calcolato pixel per pixel, e graficato, a permettere infine di poterne dare una rappresentazione grafica.

Per ovviare a ciò, sulla base degli indici vegetazionali descritti, si è cercato di calcolare delle metriche in grado di dare una rappresentazione dell'indice relativo ad un'immagine, in termini numerici piuttosto che grafici.

Valori numerici permetteranno, non solo di poter avere delle statistiche riguardanti una o più piante, ma anche di poter applicare successivamente degli algoritmi di machine learning con lo scopo di distinguere, sulla base delle suddette metriche, le piante sane da quelle malate.

Da notare che, le seguenti metriche sono state calcolate escludendo il terreno; questo può essere fatto tramite due metodologie:

Rimozione del terreno tramite photo-editing dell'immagine (*come vedremo non facilmente fattibile*), o meglio, tramite esclusione di range di valore dell'indice vegetazione che sono appunto caratteristici del suolo (*per lo più valori < 0*).

Si definisce quindi la seguente nomenclatura*:*

- IV: Valore dell'indice vegetazionale generico;
- $Threshold_1$, $Threshold_2$: due soglie ben definite;
- *Healthy Range:* insieme dei pixels aventi $IV > Threshold_1$;
- *Unhealthy Range:* insieme dei pixels aventi $Threshold_1 < IV < Threshold_2$;
- N: numero di pixels ∈ Unhealthy Range;
- P: numero di pixels ∈ Healthy Range.

Le metriche calcolate, sono le seguenti:

1. **Healthy Range Mean:** Media dei pixels aventi $IV > Threshold$;
2. **Unhealthy Range Mean:** Media dei pixels aventi $Threshold_1 < IV < Threshold_2$;
3. **Unhealthy Ratio:** $\frac{N}{N+P}$
4. **Healthy Ratio:** $\frac{P}{N+P}$
5. **Mean IV:** Indice vegetazionale medio, calcolato sull'intera immagine, escludendo il terreno.

Capitolo 3

3.1 Strumenti e linguaggio utilizzati

Proseguendo nei diversi capitoli, vedremo il calcolo degli indici vegetazionali e delle relative metriche.
Successivamente vedremo anche che sulla base delle metriche, ma anche delle singole foto, verranno costruiti dei modelli di machine learning per la classificazione dello stato di salute delle piante.
Per rendere possibile ciò si è scelto di fare uso del linguaggio di programmazione/scripting *Python 3.7* con relative librerie, e della suite *Google Colab.*
Si è deciso di utilizzare *Python* per la sua adattabilità e flessibilità nello sviluppo di prototipi funzionanti.
Inoltre esso supporta una vasta quantità di librerie che permettono di concentrarsi sul *cosa fare* piuttosto che sul *come farlo*.
Risulta rapido nella stesura di scripts, ma all'evenienza consente approcci più avanzati come l'*object oriented programming* e il *multithreading.*
Grazie alle numerosissime librerie e moduli permette, in poche righe di codice, l'analisi e la rappresentazione grafica dei dati, che nel nostro caso ci tornerà estremamente utile.
Ed in ultimo, ma non per importanza, esso offre pieno supporto a librerie per il machine learning, come *Scikit-Learn* e *TensorFlow.*

Come *code editor* si è scelto di utilizzare quello incluso nella suite di Google Collaboratory.
Esso non è esclusivamente un editor di codice, ma uno strumento sviluppato da Google, che permette la condivisione e l'esecuzione di codice Python all'interno di un browser senza che sia necessaria alcuna configurazione.

Più precisamente, esso permette l'esecuzione di un normale *Jupyter Notebook,* ma online, facendo uso di cloud computing.
Questo consente di slegare il programma e la sua esecuzione, dall'hardware su cui il browser è in esecuzione, permettendo quindi anche a computer obsoleti, o meno prestanti, lo sviluppo e l'esecuzione di processi estremamente onerosi in termini di risorse.
Esso dispone *out-of-the-box* della maggior parte delle librerie più famose e utili disponibili per python, senza alcuna installazione necessaria; Ed anche qualora esso non disponesse di una particolare libreria, sarà sempre possibile installarla sul momento.
Come detto, Colab fa parte della suite Google, e come tale è in grado di interfacciarsi ad altri strumenti proprietari come Google Drive.
Questo permette la rapida condivisione e portabilità, non solo del codice, ma anche dei dati da esso usati.

Capitolo 4

4.1 Il Dataset

Una volta esaminati gli indici vegetazionali da voler applicare, è necessario dare uno sguardo al dataset a disposizione.
Esso è composto da 186 set di foto, relative a circa 40 piante.
Ogni set comprende 6 fotografie:

- 1 in formato JPG comprendente i tre canali RGB;
- 3 distinte in formato TIF per i distinti canali R, G, B;
- 1 in formato TIF per la banda RED-EDGE;
- 1 in formato TIF per la banda NIR.

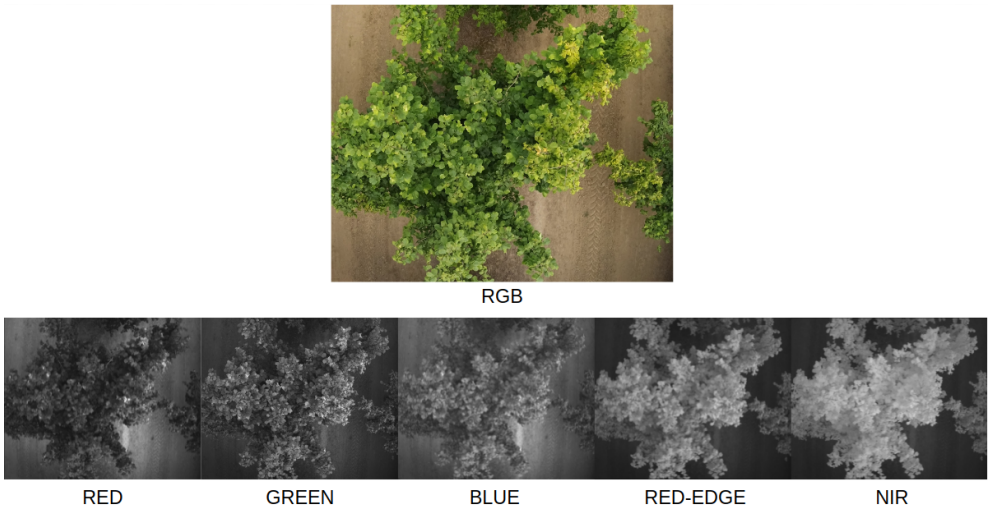


Figura 4-1: Foto RGB insieme alle singole bande in grayscale

Le foto sono state raccolte nelle giornate del 17/06/2021, 29/06/2021, 15/07/2021 ed infine 22/07/2021.
Ad ogni giornata vi è allegato un file di testo contenente le informazioni relative allo stato di salute di ogni singola pianta, e comprende:

- **Filename Immagine**: Composto dal prefisso iniziale DJI concatenato ad un codice di 4 cifre ed ovviamente all'estensione.
  Il codice di 4 cifre oltre ad essere dipendente dall'ordine

sequenziale di scatto, contiene come ultima cifra, un numero identificativo della banda in cui è stata scattata la foto, con la seguente regola:
DJI_xxx0.jpg → **RGB***; DJI_xxx1.tif* → **BLU***;*
DJI_xxx2.tif → **GREEN***; DJI_xxx3.tif* → **RED***;*
DJI_xxx4.tif → ***Red-Edge****; DJI_xxx5.tif* → ***NIR****;*

- **Label_Fisio** e **Label_Pato:** Queste due colonne contengono lo stato di salute della pianta, sia da un punto di vista fisiologico che da un punto di vista patologico.
  Esse sono state codificate in binario, con la regola:
  Sano → 0; Malato →1;
- **Plant_id:** Contiene l'identificativo univoco della pianta, utile a distinguerla tra le diverse riprese.
  È del tipo: *P_xxxx;*
- **Altri dati fisiologici:** Insieme di colonne contenenti misure raccolte dagli esperti botanici nella sessione di ripresa. Essi non sono presenti per ogni pianta, ed in caso di assenza viene segnalato con il carattere "/".

4.2 Obiettivi e descrizione approcci

Come detto precedentemente, quello che vuole essere lo scopo di questo progetto, è quello di esplorare varie tecniche di analisi delle riprese delle piante di nocciolo, al fine poi, di poter classificare automaticamente le immagini, e quindi le piante, in sane e malate.
A tale scopo, si sono esplorati principalmente due diversi approcci.
Il primo approccio vede l'utilizzo dell'intera fotografia, il secondo invece ne vede la suddivisione in più riquadri, e l'analisi di ognuno di essi.
Nei capitoli che seguono, esploriamo inizialmente il primo approccio, successivamente vedremo nel dettaglio il secondo.

4.3 Approccio 1: Analisi dell'intera foto

4.3.1 Problemi del dataset

Il primo approccio prevede quindi l'analisi dell'intera foto.

Come abbiamo detto, il dataset è composto da delle riprese di circa 40 piante.

Un primo problema che si presenta è quindi dato dalla scarsità di immagini all'interno del dataset.

In particolare, come vediamo dalla *figura 4-2*, le piante etichettate come malate, risultano essere una frazione molto ridotta del totale delle piante.

Relativamente a ciò è necessario fare inoltre una precisazione;

In questa prima fase del progetto si è deciso di non distinguere piante malate "fisiologicamente" e piante malate "patologicamente", ma supposto che uno dei due bit, all'interno del file di testo contenente le labels, sia settato, la pianta verrà considerata malata.

Questo perché un'ulteriore suddivisione delle piante in malate "fisiologicamente" e "patologicamente", avrebbe portato per ognuna delle due classi un quantitativo di immagini irrisorio.

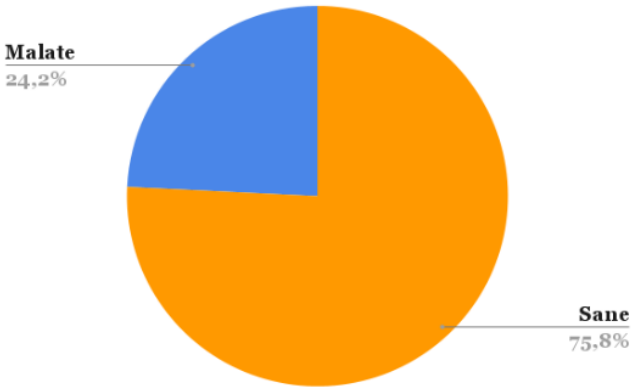


Figura 4-2: Distribuzione delle due classi nel dataset

Un ulteriore problema che si presenta, è relativo alla fotografia, in particolare è legato sia alle caratteristiche fisiche delle piante di nocciolo, che alla disposizione che esse hanno nel terreno utilizzato per le riprese.

Infatti come possiamo notare dalle immagini sottostanti, nella quasi

totalità delle foto, non risulta essere presente esclusivamente una pianta, ma di più.

Questo al momento del calcolo di metriche basate sugli indici vegetazionali, porta ad un risultato falsato, poiché non imputabile ad una singola pianta.

Figura 4-3: Esempio di scatto che include più una pianta.

Per risolvere tale problema, risulta necessario effettuare un editing delle immagini, al fine di isolare la singola pianta.

In una prima fase di esplorazione di questo approccio, tale processo ha richiesto l'uso di un software di photo-editing, e ha visto i seguenti steps:

1. **Estrazione di una maschera dalla foto a colori RGB**.
2. **Applicazione della maschera sulle foto nelle diverse bande:** Infatti è bene sottolineare come, la pianta debba essere estratta anche dalle immagini nelle varie bande (*immagini in scala di grigi*).
3. **Correzione delle differenze tra le varie foto:** Come vedremo tra poco, un'ulteriore complicazione è che le immagini nelle varie bande non sono sovrapponibili e questo comporta problematiche nel calcolo degli indici vegetazionali.

4. **Rimozione piante non in esame:**
   NB: Questa rimozione dovrebbe essere fatta MANUALMENTE, per ogni singola pianta. Questo non è concepibile in quanto si vuole un processo più automatizzato possibile.

4.3.2 Problematiche photo-editing

Questo approccio evidenzia 2 principali problematiche:

1. **Difficoltà nel definire i contorni di una pianta**
2. **Immagini, nelle varie bande, non sovrapponibili**

Il primo problema è legato sempre alla fisionomia delle piante e alla distanza che separa l'una dall'altra.
Infatti, dalle foto sottostanti, possiamo notare come molto spesso le piante appaiono intrecciate e accavallate tra loro rendendo impossibile andare a stabilire dove inizia e finisce il "corpo" di una pianta.
Anche andando ad effettuare un editing manuale (*non praticabile/non automatizzabile)*, è impossibile capire dove finisca una pianta ed inizi l'altra.

Figura 4-4: Esempio di foto che include più piante.
Notare come sia complesso andarne a delimitare i contorni.

Figura 4-5: Ulteriore esempio in cui risulta complesso definire i contorni della pianta in esame.

Il secondo problema è invece legato al fatto che le foto nelle varie bande, non sono sovrapponibili.

Per capire meglio il concetto, supponiamo a titolo di esempio, di voler applicare il processo di editing descritto sopra ad un'immagine presa dal dataset, e sul risultato di voler calcolare l'indice vegetazionale NDVI.

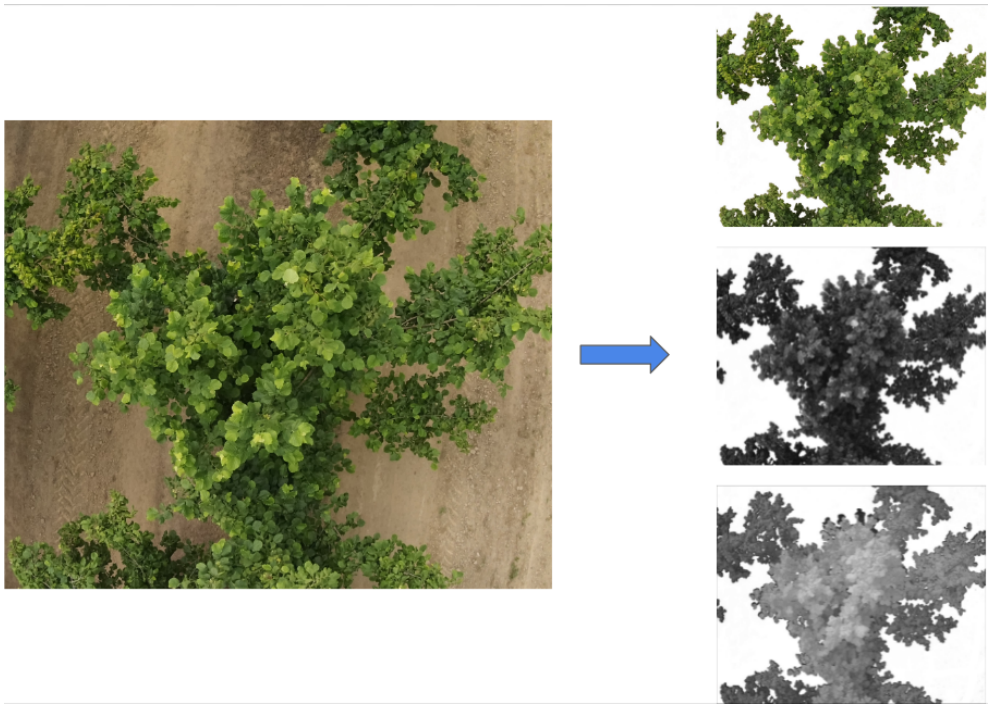


Figura 4-6: Risultato estrapolazione pianta nelle diverse bande

Nella *figura 4-6* è raffigurata l'estrazione della pianta in tutte le bande necessarie al calcolo dell'indice NDVI.

Il calcolo dell'indice, e quindi dell'immagine finale, avviene applicando la *formula 2-1,* pixel per pixel, alle varie immagini nelle diverse bande.

Il risultato, che ricordiamo essere compreso tra -1 e +1, verrà successivamente codificato con un determinato colore.

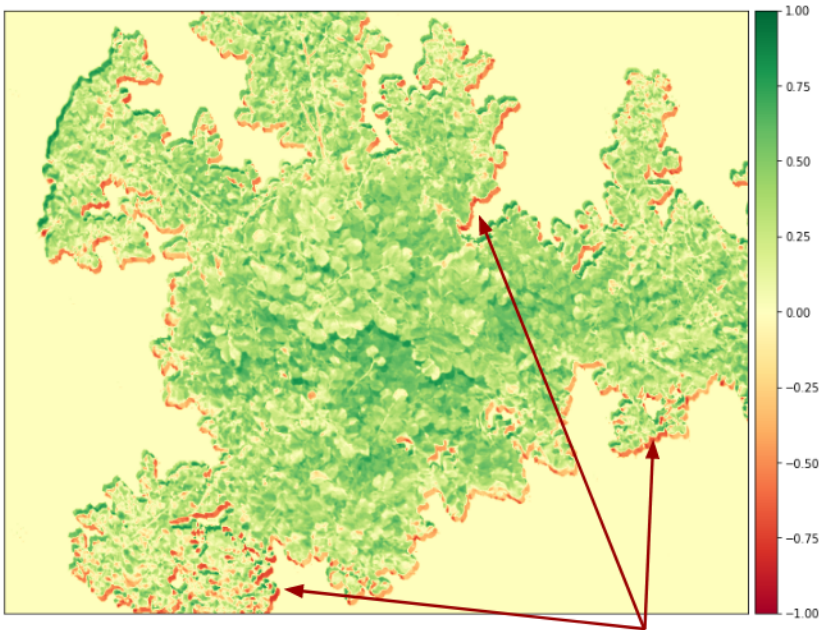


Figura 4-7: NDVI applicato alla foto editata

Nella *figura 4-7,* è possibile vedere il risultato di tutte le operazioni, e quindi del NDVI della pianta in esame.

Notiamo come le aree più verdi, rappresentino un indice con valore più alto, mentre quelle più rosse invece un indice con valore più basso.

È possibile vedere anche che lo sfondo è stato codificato come 0.

Quello che però salta subito all'occhio, è che alcune zone della pianta, in particolare in questo caso i contorni di essa, risultano estremamente rossi.

Questo rossore non è rappresentativo di zone della pianta malate, ma bensì è un artefatto che si viene a generare a causa della differenza che vi è tra i vari scatti nelle varie bande.

Ricordiamo che l'indice vegetazionale non è altro che un'operazione aritmetica applicata tra i vari pixel delle varie bande.

Uno sfasamento/movimento tra le varie immagini relative a bande diverse, porterà ad un valore dell'indice falsato in corrispondenza della zona in cui si presenta tale spostamento.

Come detto precedentemente infatti, le immagini tra le varie bande non risultano essere sovrapponibili, e questo principalmente per 2 motivi:

1. Errore di parallasse dovuto alla diversa posizione dei sensori nel drone.
2. Errore dovuto al movimento del drone tra uno scatto e l'altro. In particolare è problematico il delay temporale che intercorre tra lo scatto della foto RGB e quelle multispettrali. Questo perché, è proprio la foto RGB che viene presa come riferimento per la creazione della maschera necessaria all'estrazione della pianta nelle immagini multispettrali.

A prova di ciò è possibile dare uno sguardo alle due immagini seguenti.

Figura 4-8: Movimento tra uno scatto e l'altro

Notiamo come la persona nell'inquadratura si trovi in una posizione diversa tra lo scatto in RGB e quello nello spettro del NIR.

Quindi volendo riassumere le problematiche relative alle immagini ed in particolare l'estrapolazione della pianta in esame abbiamo le seguenti complicazioni:

1. Presenza all’interno di ogni foto di più piante e impossibilità nel distinguere i contorni della pianta in esame.
2. La rimozione delle piante non in esame, data la forte eterogeneità delle foto e il problema 1 esposto sopra, deve avvenire manualmente. In un processo automatizzato questo è impensabile.
3. L’estrazione della pianta tramite maschera porta ad artefatti nel risultato finale dell’indice vegetazionale. Questo dovuto al fatto che le immagini nelle varie bande non sono sovrapponibili, in particolare non lo sono le immagini RGB con quelle multispettrali.

A valle di queste considerazioni, si è deciso di non applicare photo-editing alle immagini, e quindi di considerare nell’analisi, tutte le piante all’interno di esse.
Andando quindi a specificare, se nella foto, in generale, risultano esserci piante in sofferenza o meno.

4.3.3 Condizioni di luminosità negli scatti

Un'ulteriore problematica di cui ci si è accorti durante l’analisi delle fotografie, e che risulta particolarmente importante nel calcolo degli IV, deriva dall’esposizione fotografica e le variazioni di luminosità tra i vari scatti.
In particolare, sfogliando le foto è possibile notare come, tra le diverse giornate di riprese, le condizioni meteo cambino molto.
Prendiamo come esempio due scatti della stessa pianta, avvenuti a meno di 7 giorni l’uno dall’altro.

Figura 4-9: Stessa pianta con condizioni di luminosità differenti

Salta subito all'occhio, come nella foto di sinistra la luce colpisca uniformemente la pianta, lasciando ben poche zone in ombra.
Nella foto a destra invece risultano esservi molte zone in ombra, e la luce colpisce per lo più solo il lato sinistro della pianta.
Ovviamente le zone in ombra, in cui i dettagli della pianta non sono visibili, comportano una perdita di informazione.
Ma il problema si propaga anche agli scatti multispettrali, oltre che alla foto RGB.

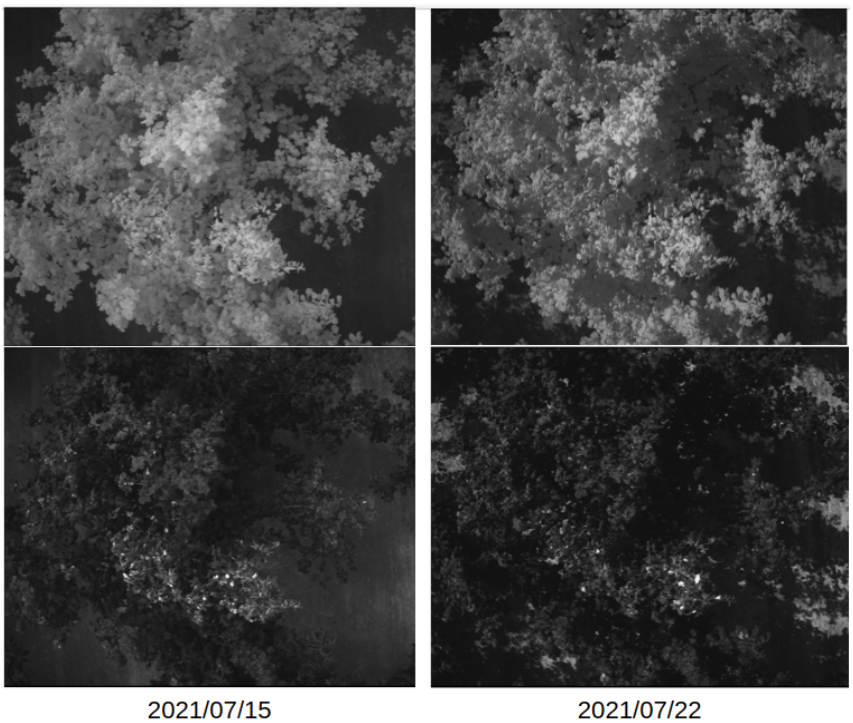


Figura 4-10: Riga superiore scatti NIR, riga inferiore canale Rosso

Nella figura soprastante, si è presa la stessa pianta di *figura 4-9,* ma in particolare si mettono a confronto, in questo caso, il canale Red e il NIR.
Come detto prima, le due coppie di foto si differenziano in termini di tempo, di meno di 7 giorni.
Possiamo notare in particolare che nella foto appartenente al canale Rosso, il terreno tende a mescolarsi con la pianta, e non risulta ben definito e distaccato. Cosa ben diversa invece nella foto del 15 luglio.
Questo porta a delle complicazioni nel calcolo degli indici vegetazionali, come possiamo vedere dalle successive immagini:

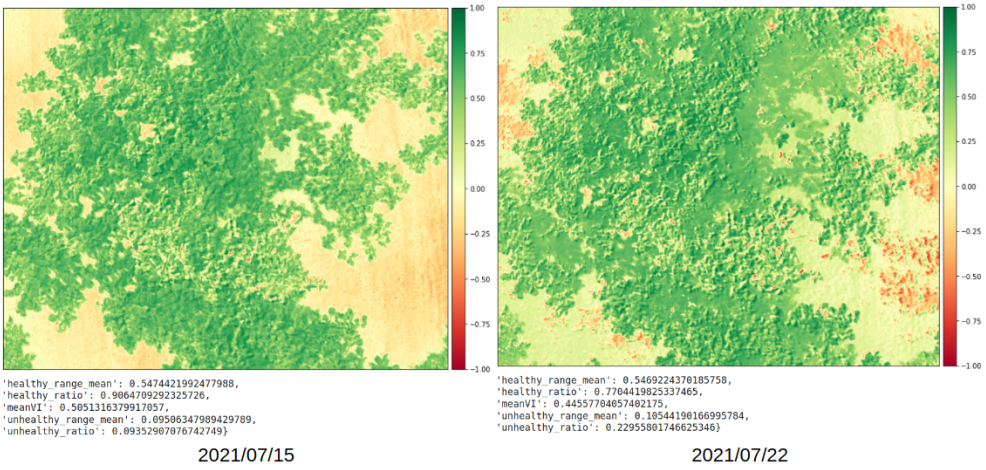


Figura 4-11: Differenza nel NDVI dovuta alla diversa esposizione

Notiamo come, nel NDVI calcolato per le piante in esame, sia graficamente che in termine di metriche, si abbiano risultati piuttosto differenti.
Visivamente notiamo come, nella foto di destra, parte del terreno presenti delle sfumature di verde, questo comporta problematiche nel momento in cui, nel calcolo delle metriche, il terreno viene escluso per valori dell'indice sotto una determinata soglia.
Infatti, quelle sfumature di verde, corrispondono a valori del NDVI sopra lo 0, che vengono quindi presi in considerazione nel calcolo delle metriche.
Notiamo inoltre che, zone in ombra dell'albero, portino a valori dell'indice

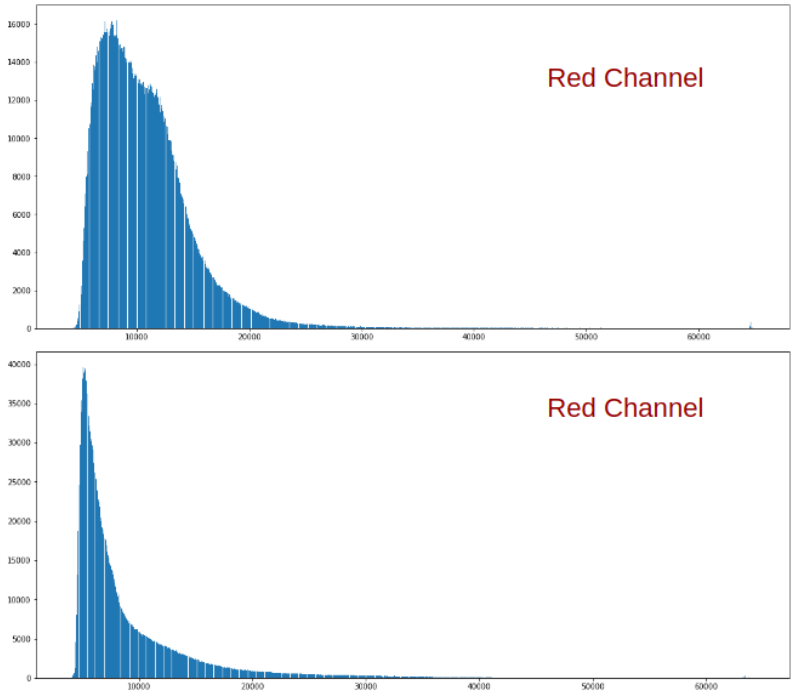
sensibilmente più bassi (*verde più chiaro*).
In generale, osservando le metriche, le differenze più marcate le notiamo:

- Nel MeanVI: Corrispondente al NDVI medio
- Healthy Ratio: Corrispondente alla porzione della pianta in salute
- Unhealthy Ratio: Corrispondente alla porzione della pianta in sofferenza

Dai valori di queste metriche notiamo come, la foto del 15 luglio (*a sinistra*), riporti circa un 9% (*unhealthy ratio * 100*) della pianta in sofferenza.
Mentre quella di destra del 22 luglio, ne riporti invece quasi il 23%.
Ed ovviamente la differenza si presenta anche nel NDVI medio, che va da 0.50 nella foto di sinistra, a 0.44 in quella a destra.
In generale questi valori starebbero ad indicare a sinistra una pianta decisamente più in salute rispetto quella di destra, nonostante alla fine siano la stessa pianta a distanza di pochi giorni.

Come dicevamo, questo problema è principalmente dovuto ad un errato bilanciamento dell'esposizione.
Per averne un'ulteriore prova, è possibile utilizzare uno strumento conosciuto come *istogramma di un'immagine*.
Esso è un tipo di istogramma che permette di graficare la distribuzione tonale di un'immagine digitale.
In particolare quindi, esso permette di quantificare il numero di pixel per ogni valore tonale dell'immagine.
Si ha che l'asse orizzontale del grafico rappresenta le variazioni tonali, mentre l'asse verticale rappresenta il numero di pixel di quel particolare tono.
In particolare, la parte sinistra dell'istogramma rappresenta le aree scure, mentre più ci si spinge verso destra, più si vanno a quantificare le aree più chiare.

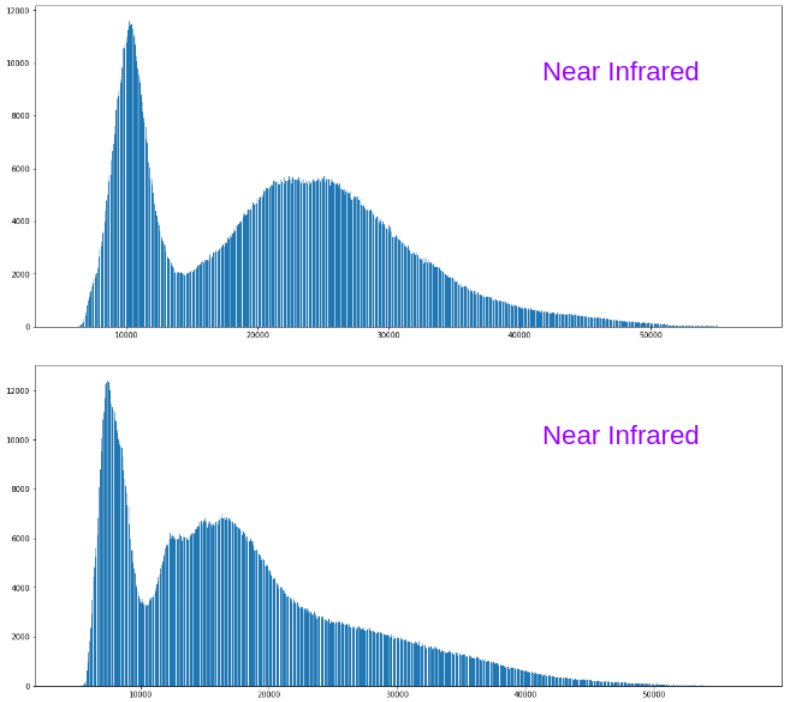
Andando quindi a visualizzare gli istogrammi per le immagini precedenti coinvolte nel calcolo del NDVI, possiamo vedere la differenza che vi è tra le due giornate e le due diverse esposizioni:



2021/07/15

2021/07/22

Figura 4-12: Istogramma del canale rosso



2021/07/15

2021/07/22

Figura 4-13: Istogramma dell'immagine NIR

Nelle due coppie di istogrammi, in alto è rappresentato quello di una fotografia con un buon bilanciamento dell'esposizione, mentre nel grafico al di sotto è possibile osservare l'istogramma di una che invece presenta un'errata esposizione.
È possibile notare come nella *figura 4-12*, l'istogramma della foto del 22/07, sia fortemente *schiacciato* verso la parte sinistra. Questo è sinonimo di una elevata quantità di pixels tendenti al nero, che quindi apportano un'informazione errata (*parte dell'informazione è effettivamente persa*) e vanno a falsare il risultato del calcolo dell'indice vegetazionale.
Questo rende estremamente difficile un'analisi comparativa e temporale dell'evoluzione dello stato delle piante, in quanto le differenze non saranno dettate solo da variazioni nello stato di salute, ma bensì da esposizioni e condizioni di luminosità differenti.
Quindi al momento delle riprese, ed in particolare dei singoli scatti, oltre a cercare di effettuarli tutti all'incirca allo stesso orario, e con il sole allo zenit, è necessario un corretto bilanciamento dell'esposizione, al fine di ridurre al minimo l'errore introdotto da differenti condizioni di scatto.

4.4 Calcolo e risultati metriche

Alla fine del processo di analisi delle immagini, sono state calcolate le diverse metriche, esposte in precedenza, basate sugli indici vegetazionali.
In questo primo approccio, come già esposto, si è presa in analisi l'intera pianta, o meglio, tutte le piante all'interno dello scatto.
In fase di calcolo metriche, al fine di non far incidere il terreno con i valori relativi ad un indice non legato alla vegetazione, è stata fissata una soglia, al di sotto della quale i valori vengono esclusi dal calcolo.

Questa soglia può variare in base all'indice in esame, ma tendenzialmente è stata fissata ad un valore di circa 0, in modo da escludere tutti i valori al di sotto, e quindi negativi, che non sono rappresentativi di vegetazione.
In ogni caso, diverse soglie in un successivo momento potranno essere settate in base alle performance ottenute.

Calcolate le metriche, si è andati a ricercare delle correlazioni tra i valori ottenuti, e lo stato della pianta.
I risultati ottenuti però, non ne hanno permesso una corretta discriminazione, infatti, nell'immagine seguente è raffigurata la matrice di correlazione calcolata sugli scatti di giugno, e come possiamo notare, la correlazione tra le metriche e lo stato di salute, è molto bassa.

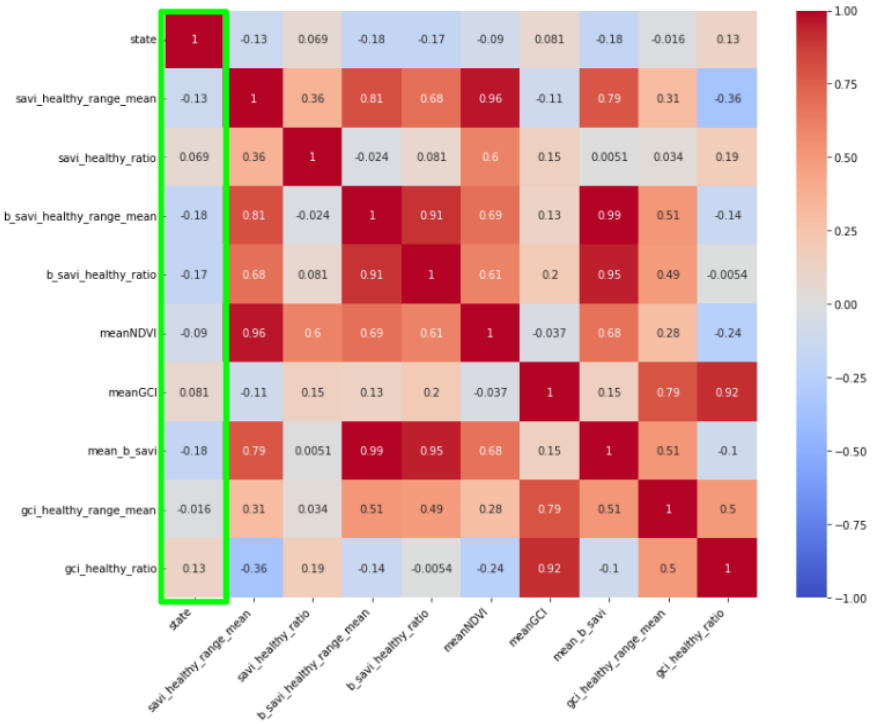


Figura 4-14: Matrice di correlazione

Questo è causato dal fatto che molto spesso la porzione di pianta malata è molto ridotta rispetto all'intera pianta.
Se a questo ci si aggiunge il fatto che, all'interno delle foto sono sempre

presenti anche altre piante al di fuori di quella in esame, il rapporto porzione-malata/intera-vegetazione si riduce ancor di più.
Altre volte invece la pianta in esame è etichettata come sana, ma le piante intorno ad essa sono visibilmente malate.
Questo fa si che non si riesca a fare una vera e propria discriminazione sana/malata dell'intera pianta se i calcoli vengono effettuati sull'intera fotografia.
Ma invece risulta necessario suddividerla in sezioni di dimensione ben definita, e analizzare individualmente ognuna di esse.
In questo modo si andrà ad analizzare il dettaglio di ogni singola pianta e si ridurrà il rumore proveniente da vegetazione al momento non in esame.
Questo secondo approccio, che verrà descritto in dettaglio nel capitolo successivo, come vedremo porterà a risultati più interessanti e significativi.

Capitolo 5

5.1 Approccio 2: Slicing dell'immagine

Il secondo approccio che si è voluto andare ad applicare, consiste nel suddividere la foto intera in sotto-riquadri, ognuno contenente una porzione di pianta.
Solo successivamente si procederà al calcolo degli Indici Vegetazionali e delle metriche per ognuno dei riquadri ed infine alla loro classificazione.
Questo secondo approccio non solo consente di individuare una pianta malata (*banalmente se un riquadro di una pianta è malato ne segue che la pianta lo è anche*), ma, dato che una porzione di pianta malata non ne implica necessariamente l'intera compromissione, esso permette di andare ad individuare quale o quali porzioni sono effettivamente in sofferenza e programmare un intervento esclusivamente su di esse.

La prima fase di questo secondo approccio, consiste nell'andare a definire con che criterio effettuare i ritagli dell'immagine.
La strada che si è deciso di intraprendere, consiste nel suddividere l'immagine in una matrice NxN, dove ogni elemento di essa, corrisponde ad un riquadro della foto.
Risulta quindi necessario andare a definire in quante sotto-immagini dovrà quindi essere suddivisa la foto, in sostanza quindi, definire N.
Per questa decisione, è necessario tenere in considerazione diversi fattori, come ad esempio, l'etichettatura di tutti sotto-riquadri derivanti da un'immagine.
Infatti, le labels a disposizione, si riferiscono al dataset originale, ma questo con la suddivisione di ogni foto in una matrice NxN di *sotto-immagini*, ovviamente cambierà e di conseguenza, tutte le nuove

immagini risultanti dalla suddivisione dovranno essere etichettate nuovamente da personale esperto.

Un altro fattore da tenere in considerazione nella decisione di N è legato alla risoluzione delle foto.

Un valore di N troppo grande comporterebbe ritagli molto piccoli e di bassa definizione, e queste risulterebbero difficilmente interpretabili dal team di botanici. Ne seguirebbe quindi una fase di applicazione delle labels estremamente difficile, lunga e poco precisa.

D'altro canto un valore di N molto piccolo tende a non risolvere il problema per il quale si è deciso di intraprendere questo approccio. Infatti, esso porterebbe a foto contenenti aree ancora piuttosto ampie che possono contenere porzioni di piante in diverse condizioni.

Si provi a pensare infatti, al problema per cui, all'interno di un riquadro la stragrande maggioranza della vegetazione è in salute ma una piccola parte è malata. Anche in questo caso con un eventuale label *malata:1,* ci sarebbe un'alta probabilità di sbagliare la previsione, a causa di metriche poco precise che faticano ad evidenziare la presenza di una piccola porzione di vegetazione effettivamente sotto stress.

Figura 5-1: Slicing con N=3 e zoom

Figura 5-2: Slicing con N=4 e zoom

Dopo attente valutazioni, comprendenti anche il parere del CNR, si è deciso di fissare $N = 3$, e di ottenere quindi i ritagli da una matrice 3x3.
Per l'automatizzazione del processo di slicing è stato scritto uno script in python che, dato in ingresso una foto, restituisce in output nove distinte immagini, rinominate secondo una precisa regola che permette la localizzazione di ogni riquadro all'interno della fotografia originale.
Quindi, ogni file contenente il riquadro, sarà rinominato secondo la regola: *DJI_XXXX_C_R,* dove *XXXX* è il codice identificativo della foto originale, *C* è la colonna ed *R* è la riga.
In questo modo, individuato un riquadro malato, sarà possibile risalire alla foto e di conseguenza alla pianta.

5.2 Creazione labels e analisi nuovo dataset

La fase successiva ha coinvolto fortemente il team di botanici del CNR nella creazione delle labels.
Ognuno dei riquadri è stato quindi esaminato e tramite sia un'analisi visiva, che tramite i dati fisiopatologici relativi alla pianta la cui porzione apparteneva, è stato redatto un documento comprendente tutte le labels per tutti i riquadri.

Questo ha permesso di procedere con lo step successivo e calcolare quindi gli indici vegetazionali e le relative metriche.

Prima di procedere con la descrizione dello step successivo, è importante visionare la figura che segue, dove è possibile vedere come il rapporto tra immagini etichettate come *sane* ed immagini etichettate come *malate*, sia profondamente cambiato.

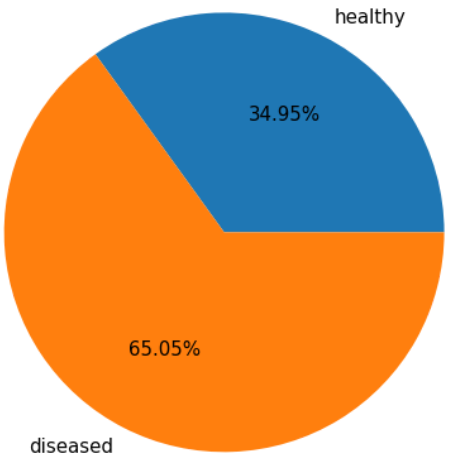


Figura 5-3: Distribuzione delle due classi nel nuovo dataset

Notiamo infatti che in *figura 4-2,* la porzione dominante del grafico a torta era relativa alle piante sane, mentre risultavano in numero minore quelle malate, in particolare circa il 76% per le prime, e il 24% per queste ultime. Qui la proporzione tra le due classi si è capovolta, e questo è attribuibile ad alcuni fattori:

Le labels relative al primo dataset sono state assegnate in base alle informazioni derivanti dalla combinazione di analisi di laboratorio su dei campioni prelevati dalla pianta e ispezione visive di essa.

L'ispezione sul campo e il prelievo dei campioni però non ha permesso di visionare e analizzare le zone della pianta relative alla chioma, cosa che invece è stata possibile tramite le foto ed i ritagli.

Inoltre c'è da sottolineare come spesso, anche tramite le analisi di laboratorio, fosse evidente come solo alcune foglie/branche della pianta

risultassero sintomatiche, piuttosto che l'intero albero; ed in questa casistica si è preferito etichettare la pianta come *sana.*

Nei ritagli dell'immagine invece, si stanno andando ad esaminare i dettagli di essa, permettendo quindi un'assegnazione delle labels più precisa ed univoca, e portando di conseguenza ad un sostanziale aumento dei data-points appartenenti alla classe *malata.*

Questo quindi spiega la grande variazione di proporzione tra le due classi.

Vediamo adesso nell'immagine che segue una porzione della tabella rappresentativa del nuovo dataset.

plant_id	state	meanSAVI	savi_healthy_range_mean	savi_unhealthy_range_mean	savi_healthy_ratio	mean_b_savi	b_savi_healthy_range_mean	b_savi_healthy_ratio	meanGCI	gci_healthy_range_mean	gci_unhealthy_range_mean	gci_healthy_ratio
P_0017	0	0.585777	0.634537	0.123976	0.904496	0.295691	0.360553	0.777548	0.796249	1.149123	0.285050	0.591616
P_0017	0	0.611819	0.673377	0.098316	0.892954	0.335268	0.395969	0.818705	0.823858	1.253149	0.236760	0.577631
P_0017	1	0.553457	0.622828	0.126667	0.860184	0.325951	0.393024	0.796102	0.690825	1.121002	0.256248	0.502543
P_0017	0	0.685727	0.703511	0.137566	0.968576	0.345938	0.406232	0.820365	1.017571	1.319719	0.338994	0.691913
P_0017	0	0.762607	0.782992	0.135140	0.968534	0.395061	0.442650	0.872264	1.314515	1.572773	0.338563	0.790751
P_0017	1	0.482885	0.568216	0.126967	0.806614	0.281825	0.363882	0.728998	0.575193	1.043304	0.223771	0.428807
P_0017	1	0.593941	0.652036	0.112274	0.892370	0.303347	0.378883	0.761760	0.768303	1.253466	0.191913	0.542969
P_0017	0	0.629055	0.673562	0.130735	0.918008	0.327784	0.407631	0.768441	0.929828	1.373195	0.273280	0.596908
P_0017	1	0.446150	0.554928	0.103509	0.759030	0.258695	0.345415	0.696146	0.308603	0.974253	0.105571	0.233724

Figura 5-4: Prime nove entries del dataset

Qui di seguito invece è possibile visualizzare la matrice di correlazione tra i diversi attributi:

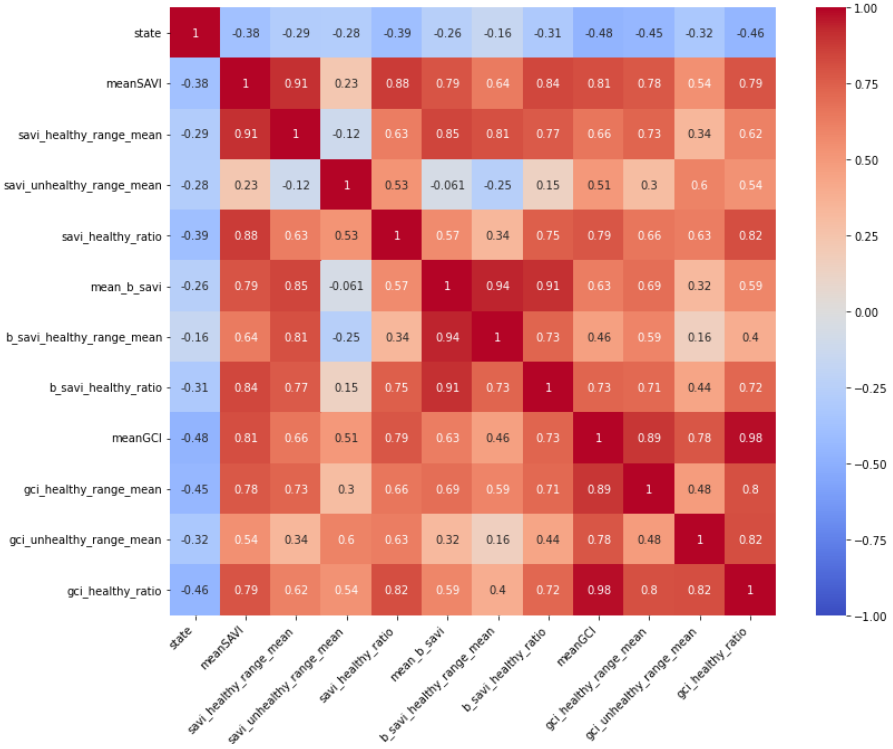


Figura 5-5: Matrice di correlazione

Confrontando essa con la matrice vista in *figura 4-14,* possiamo notare un aumento della correlazione tra i vari attributi e lo stato.
In particolare è visibile una correlazione inversa, questo perché tendenzialmente gli attributi scelti tendono ad avere valori più alti per piante sane, che ricordiamo essere rappresentate dal valore binario 0, e di contro, si ha invece il valore binario 1 per quelle malate.
Si possono notare anche attributi che presentano una fortissima correlazione tra loro, e di conseguenza uno dei due potrebbe probabilmente essere scartato, in quanto plausibilmente non apporterebbe grandi informazioni al fine della classificazione.
Si deve però tener presente che la matrice di correlazione quantifica e mostra solo le correlazioni lineari che vi sono tra le features.
Si guardi a titolo d'esempio, la feature *gci_healthy_ratio* e *meanGCI*. Esse risultano fortemente correlate tra loro, ma è da notare che esse forniscono due tipologie di informazioni diverse.
Mentre l'ultima ci segnala un valore aggregato e medio dell'indice GCI per l'immagine, l'altra ci quantifica l'area della porzione di pianta in esame, effettivamente *"malata".*
E questo è un valore importante, in quanto, si potrebbe avere una porzione di pianta malata, delineata e limitata, e il restante della pianta estremamente in salute con valori dell'indice molto elevati. Questo porterebbe ad un indice medio nel complesso non troppo basso, e di conseguenza ad una possibile classificazione errata della pianta.
Per questo la decisione della rimozione di una feature, deve essere presa soltanto dopo diverse analisi statistiche che ne confermano l'effettiva scarsa utilità.
In ogni caso, data la non eccessiva numerosità delle features nel dataset, si è deciso in questa prima fase di mantenerle tutte.
Verranno comunque effettuati dei test su modelli che prendono in esame soltanto un sottoinsieme di esse, come verrà spiegato più dettagliatamente nel *paragrafo 6.6*.

Ma in questo elaborato si è deciso di mostrare le performance dei modelli operanti su tutte le features del dataset.
Vediamo adesso la distribuzione dei valori delle principali features:

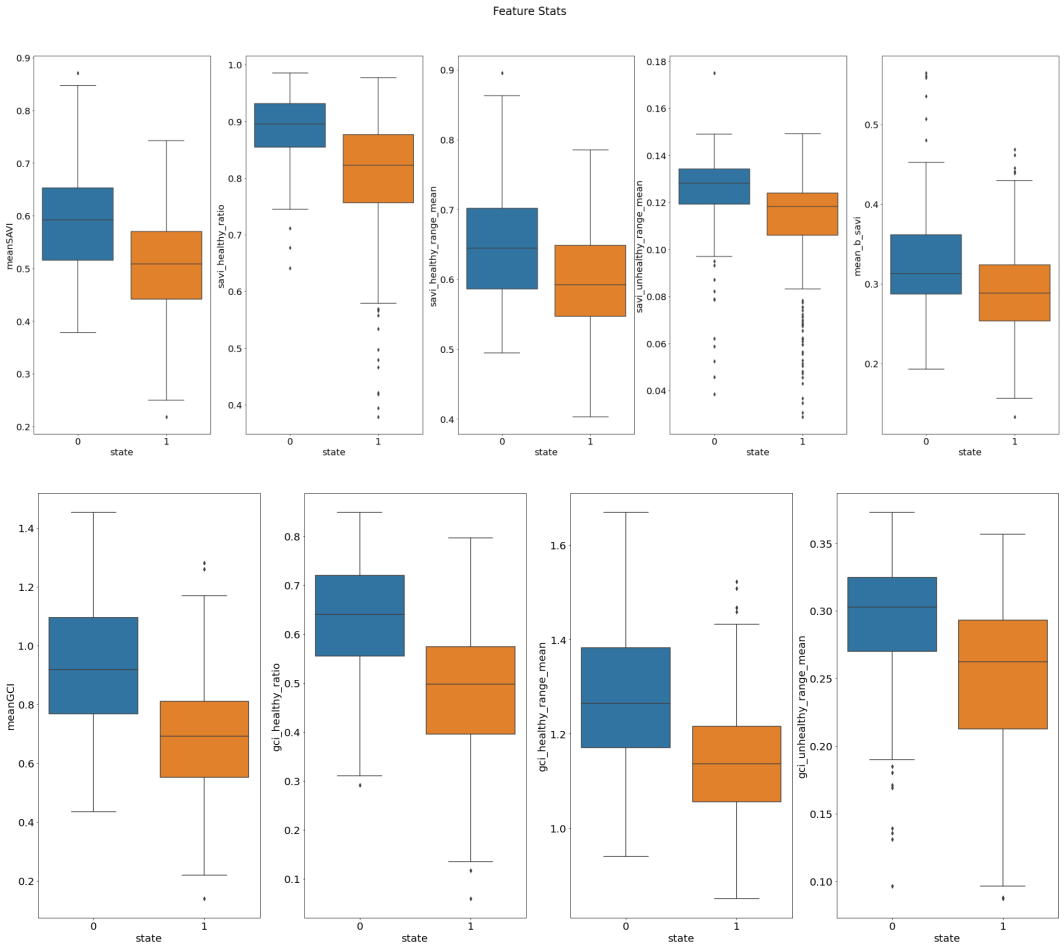


Figura 5-6: Distribuzione valori tramite box-plots

Possiamo notare che vi è una discreta sovrapposizione nella distribuzione dei valori degli attributi, ma si nota comunque un pattern che evidenzia in generale valori più bassi per le piante malate e valori più alti per quelle sane.
Di seguito verranno applicati degli algoritmi di machine learning, per provare a predire lo stato di salute della porzione di pianta all'interno di un riquadro.

Capitolo 6

6.1 Applicazione modelli di Machine Learning

Prima della vera e propria applicazione dei modelli, è necessario effettuare una suddivisione del dataset in un training set, per la fase di apprendimento e di un test set per il processo di valutazione delle prestazioni.

Date le dimensioni dell'intero dataset, si è deciso di mantenere un rapporto 80:20; in particolare, l'80% del dataset originale verrà usato per il training, e il 20% per il test set.

È importante sottolineare come il partizionamento sia avvenuto rispettando le caratteristiche e le proporzioni tra le due classi, all'interno del dataset (*stratified random sampling*).

In ogni caso ogni modello verrà poi validato tramite un processo di *k-fold stratified cross-validation,* utile per ottenere una stima delle prestazioni del modello sulla classificazione di dati nuovi.

Al fine inoltre di trovare per ogni modello le migliori combinazioni di *hyperparameters*, verranno utilizzate delle tecniche di ricerca parametri, come *Random Search* e *Grid Search.*

Una breve descrizione del funzionamento di tutti questi algoritmi verrà data più avanti, insieme alla descrizione del processo in cui sono coinvolti.

I tre algoritmi di *supervised learning* utilizzati sono:

1. Logistic Regression
2. Support Vector Machines (*SVM)*
3. Random Forest

Verrà data una descrizione di essi, nei paragrafi che seguono.

6.2 Criteri di valutazione

Come si è potuto evincere dai paragrafi precedenti, il dataset attuale risulta fortemente sbilanciato. In particolare i riquadri contenenti porzioni di piante classificate come malate, sono quasi il doppio rispetto a quelli classificati come sani.

Questo deve necessariamente essere tenuto in considerazione nella scelta dei criteri di valutazione delle performance dei modelli.

Infatti, nonostante si possa essere tentati dall'osservare principalmente l'accuratezza (*accuracy)* del modello, definita come il rapporto tra il numero di istanze correttamente classificate e il totale delle istanze presenti nel test set, nel nostro caso, questo porterebbe ad un'errata valutazione delle sue performance.

Si pensi infatti che, un banale modello che non effettua alcuna predizione, ma si limita a classificare tutti i datapoint come *malati*, avrebbe già un'accuratezza del 65%.

Per questo è necessario introdurre altri criteri, ed in particolare verranno valutati i seguenti tre: *Recall, Precision* e *F1-Score.*

Sia definita la seguente matrice di confusione:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Tabella 6-1: Matrice di confusione

Definiamo la metrica *Precision,* come:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} = \frac{True\ Positive}{Total\ Predicted\ Positive}$$

Formula 6-1: Calcolo Precision

Mentre, definiamo la metrica *Recall,* come:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{True\ Positive}{Total\ Actual\ Positive}$$

Formula 6-2: Calcolo Recall

In particolare la metrica *Precision,* misurerà quanti dei predetti malati, lo erano effettivamente.

Mentre, la metrica *Recall,* misurerà, quanti della totalità dei riquadri etichettati come malati, ne sono stati correttamente predetti e quindi identificati.

Nel nostro caso, è estremamente importante identificare una porzione di pianta malata, in quanto essa potrebbe essere contagiosa o necessitare di un trattamento tempestivo.

D'altro canto, abbiamo visto come il dataset a disposizione sia sbilanciato verso la classe *diseased,* e quindi nonostante si preferisca probabilmente essere più "*pessimisti*", e classificare in caso di forte dubbio, una pianta come *malata* piuttosto che come *sana*, risulta comunque necessario bilanciare le due metriche.

Per il bilanciamento delle suddette risulta estremamente utile la metrica denominata *F1-Score* e definita come media armonica tra *precision* e *recall*:

$$F1 = 2 \times \frac{Precision*Recall}{Precision+Recall}$$

Formula 6-3: Calcolo F1-Score

Nella fase di tuning dei modelli, che vedremo successivamente, saranno queste (ed in particolare quest'ultima) le metriche che si cercherà di ottimizzare.

6.3 Logistic Regression

Il primo modello che si è costruito è basato sull'algoritmo di *logistic regression.* Esso viene normalmente utilizzato per stimare la probabilità di un'istanza di appartenere ad una particolare classe.
Se la probabilità è maggiore del 50% (*o di una determinata soglia)*, allora il modello classificherà quel datapoint come appartenente a quella ben definita classe; altrimenti se essa è inferiore al 50%, l'altra classe verrà scelta.
Da questo si deduce che tendenzialmente, questo tipo di algoritmo si adatta principalmente a casi di classificazione binaria. Tuttavia è possibile adattarlo al fine di supportare classi multiple.
Il nostro problema comunque ricade nella prima categoria, ovvero quella della classificazione binaria *healthy/diseased.*

Il funzionamento dei modelli basati su Logistic Regression, un po' come i modelli di regressione lineare (*linear regression*), si basano sul calcolo di una somma pesata degli attributi (*features)* in input, ma anziché restituire in output direttamente il risultato, su di esso viene applicata una funzione logistica. Questa prende il nome di funzione sigmoidale ed è caratterizzata dalla proprietà di restituire in output valori compresi tra 0 e 1.

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

Formula 6-4: Funzione sigmoidale

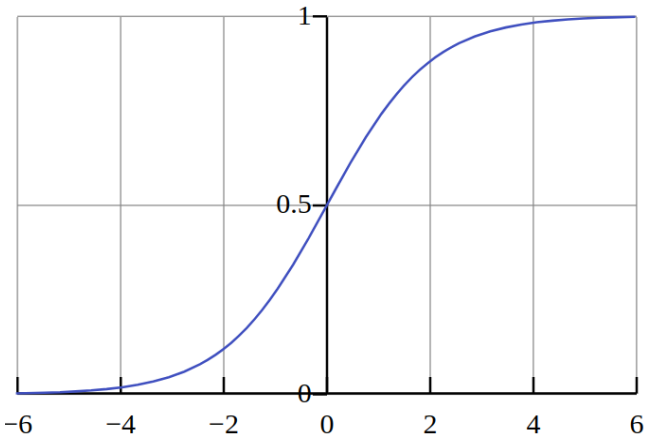


Figura 6-1: Curva logistica

Una volta che il modello avrà stimato la probabilità $\hat{p}$ che un istanza x appartenga alla classe positiva, esso effettuerà la predizione di $\hat{y}$ secondo la regola seguente:

$$\hat{y} = \{0\ se\ \hat{p} < 0.5\ \vee\ 1\ se\ \hat{p}\ \geq 0.5\}$$

Quindi la fase di training del modello consisterà nel trovare il vettore di parametri θ tale che il modello stimi un'alta probabilità per le istanze appartenenti alla classe positiva 1, e una bassa probabilità per quelle appartenenti alla classe negativa 0.

Questo è esprimibile anche tramite la seguente formula relativa alla funzione di costo (*cost function*):

$$c(\theta)\ =\ \{-\ log(\hat{p})\ se\ y\ =\ 1\ \vee\ \ -\ log(1-\hat{p})\ se\ y\ =\ 0\}$$

6.3.1 Implementazione modello e risultati

Definito quindi il dataset per il training, si è proceduto con la creazione del modello e l'ottimizzazione degli *hyperparameters* tramite la procedura denominata *Random Search Cross-validation.*

Essa consiste nel definire per ogni *hyperparameter*, uno spazio di ricerca all'interno di un dominio limitato di valori, ed effettuare un campionamento casuale di questi ultimi.

Per ogni combinazione di valori, scelta casualmente all'interno del

dominio, verrà quindi costruito un modello, e le sue prestazioni verranno valutate tramite un processo di *k-fold cross-validation.*

Quest'ultimo prevede di suddividere il training set in k partizioni, effettuare il training del modello su $k - 1$ di esse e testarlo sul set rimanente, misurandone ovviamente la varie metriche necessarie alla valutazione del modello (*accuracy, recall, precision ecc.*).

Questo processo viene ripetuto finché ognuna delle k suddivisioni non è stata usata come test set.

Da notare bene che, ogni volta che un modello viene testato e valutato, il processo prevede che esso venga scartato.

Alla fine, la combinazione di hyperparameters che ha caratterizzato il modello con più alte prestazioni, viene scelta.

Tutto questo processo viene ripetuto per un ben definito numero di iterazioni.

Nel nostro caso, questo è reso possibile tramite il modulo *RandomSearchCV* di *Scikit-Learn.*

Di seguito è possibile visionare le performance del modello sul test set, dopo la fase di tuning degli hyperparameters:

	precision	recall	f1-score	support
0	0.89	0.55	0.68	29
1	0.80	0.96	0.87	54
accuracy			0.82	83
macro avg	0.84	0.76	0.78	83
weighted avg	0.83	0.82	0.81	83

Figura 6-2: Report delle performance del modello

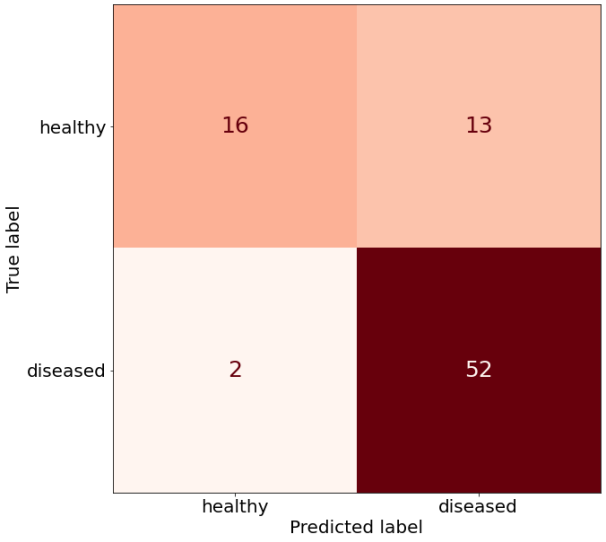


Figura 6-3: Matrice di confusione del modello applicato al test set

Il modello finale ha quindi portato ad un'accuratezza del 82%, insieme ad un *f1-score* per la classe *diseased* di 0.87.

In generale possiamo notare dalla matrice di confusione, una discreta presenza di falsi positivi (13), ovvero riquadri predetti *diseased* ma che in realtà erano etichettati come *healthy*.

A fronte però di un esiguo numero di falsi negativi (2*)*.

È possibile avere una rappresentazione delle performance del modello, tramite la cosiddetta *curva ROC* e la misura AUC.

Essa è un grafico che permette di visualizzare le performance del modello rispetto tutte le diverse soglie decisionali.

La curva rappresentata è funzione di due parametri, *True Positive Rate* e *False Positive Rate*, rispettivamente calcolati tramite le seguenti formule:

$$TPR = \frac{TP}{TP + FN}$$

$$FPR = \frac{FP}{FP + TN}$$

Formula 6-5,6-6: Calcolo True Positive Rate e False Positive Rate

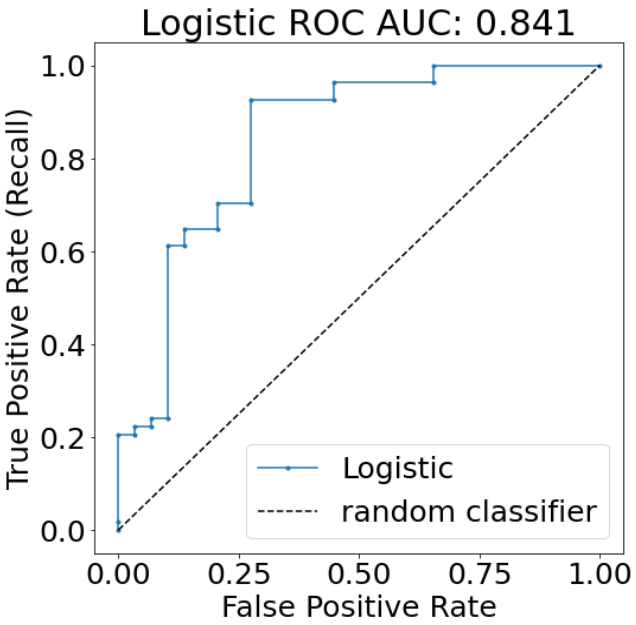


Figura 6-4: Curva ROC-AUC

Per confrontare le performance di diversi modelli e avere un valore che a partire dalla suddetta curva le riassume, si utilizza l'*AUC - Area Under the ROC Curve.*

Questa quantifica l'area sottostante la curva ROC, e i valori possono variare da 0 a 1.

Un modello le cui predizioni sono tutte errate, presenta un valore AUC = 0, mentre contrariamente, un modello con il 100% di predizioni corrette presenta un AUC = 1.

A titolo di confronto, nel grafico oltre alla curva ROC del modello testato, è presente anche quella di un modello le cui predizioni sono totalmente casuali. Quest'ultimo modello presenterebbe un AUC = 0.5 circa.

Il nostro modello preso in esame, presenta un AUC = 0.841.

6.4 SVM - Support Vector Machine

Il Support Vector Machine, (*o macchina a vettori di supporti in italiano*), è un modello di machine learning potente e versatile, in grado di effettuare non solo classificazioni lineari e non lineari, ma anche regressioni e detection degli outliers.

Esso è in realtà è un'estensione di un modello più semplice denominato *support vector classifier*, che però può essere applicato soltanto a quei dataset che contengono classi linearmente separabili.

I modelli basati sul *support vector classifier,* utilizzano il concetto di *hyperplane,* o iperpiano, che in un spazio *p*-dimensionale, consiste in un sottospazio di dimensione $p - 1$tale che esso possa dividere lo spazio p-dimensionale in due parti tra di loro separate.

In termini matematici esso può essere espresso come:

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots\ldots + + \beta_p X_p = 0$$

E dato un punto X, se esso soddisfa l'equazione, allora esso è localizzato sull'iperpiano, altrimenti, in base al fatto che il risultato sia > 0 oppure < 0, il punto si troverà da un lato o dall'altro di esso.

Come detto prima, il Support Vector Machine è un'estensione del support vector classifier, che si basa sull'applicazione di kernels, mappando implicitamente i propri ingressi (features) in un diverso spazio multidimensionale.

Esistono diverse tipologie di kernels, come il Lineare, il Polinomiale e il Radiale.

Si evita in questa sede di scendere nei dettagli teorico-matematici del modello, per la quale è disponibile una grande quantità di letteratura a riguardo.

6.4.1 Implementazione modello e risultati

Anche per questo modello, esistono un gran numero di hyperparameters da dover configurare, ed anche in questo caso, l'ottimizzazione di essi viene effettuata tramite un processo di Random Search Cross Validation.
Per la precisione in questo caso la ricerca randomica degli hyperparameters viene effettuata in accoppiata alla ricerca tramite griglia di valori, denominata Grid Search Cross Validation.
Le due tecniche sono simili, ma mentre la prima estrae dei valori random da un dominio ben definito, la seconda estrae e prova tutti i valori presenti in quel dominio.
Da questo se ne deduce che nonostante la seconda tecnica sia più precisa e completa, essa è enormemente più onerosa, specialmente in termini di tempo.
Una strategia che di solito viene utilizzata è quella di andare ad utilizzare prima l'approccio Random Search in diversi domini o intervalli di valori. Successivamente, individuato un dominio che restituisce valori più promettenti degli altri, si procede con una ricerca più fine all'interno di quel dominio tramite Grid Search.
In questo caso, gli hyperparameters per la quale si è cercata una configurazione più ottimale possibile sono:

- Kernel
- C - Cost Value
- Gamma

I risultati ottenuti dal modello finale in fase di test, sono illustrati nelle immagini che seguono.

	precision	recall	f1-score	support
0	0.83	0.52	0.64	29
1	0.78	0.94	0.86	54
accuracy			0.80	83
macro avg	0.81	0.73	0.75	83
weighted avg	0.80	0.80	0.78	83

Figura 6-5: Report delle performance del modello

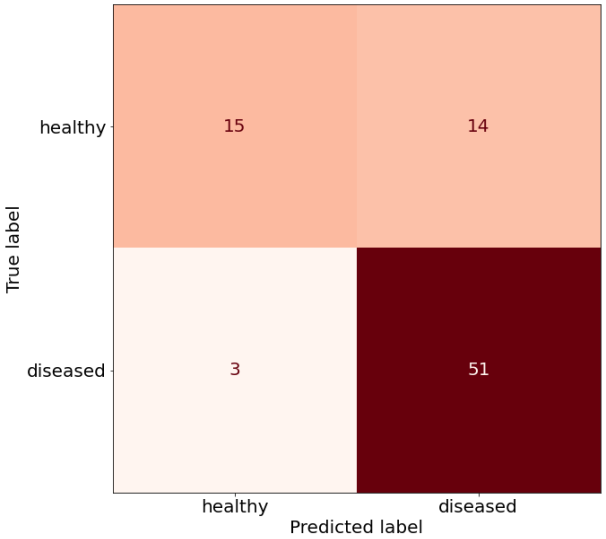


Figura 6-6: Matrice di confusione del modello

Il modello presenta performance leggermente inferiori rispetto al modello di logistic regression precedente, con un f1-score per la classe *diseased* di 0.86, e anche un recall più basso per la classe *healthy.*

In ultimo, guardando quindi all'accuratezza generale, il modello SVM presenta un valore più basso rispetto al modello precedente (*SVM: 80%, LGR:82%).*

6.5 Random Forest

Il prossimo metodo che si è andato a testare sul dataset è basato su i cosiddetti *Alberi Decisionali*, o *Decision Trees*.

L'Albero decisionale è un algoritmo di apprendimento che fa uso di un grafo con una struttura ad albero, dove i nodi foglia rappresentano delle classificazioni e le ramificazioni invece l'insieme delle proprietà o valori dei predittori che portano a quella determinata classificazione.

Ne segue che a sua volta, ogni nodo interno risulta essere una macro-classe, costituita dall'unione delle classi associate ai suoi nodi figli.

Il predicato associato ad ogni nodo interno prende il nome di *condizione di split.*

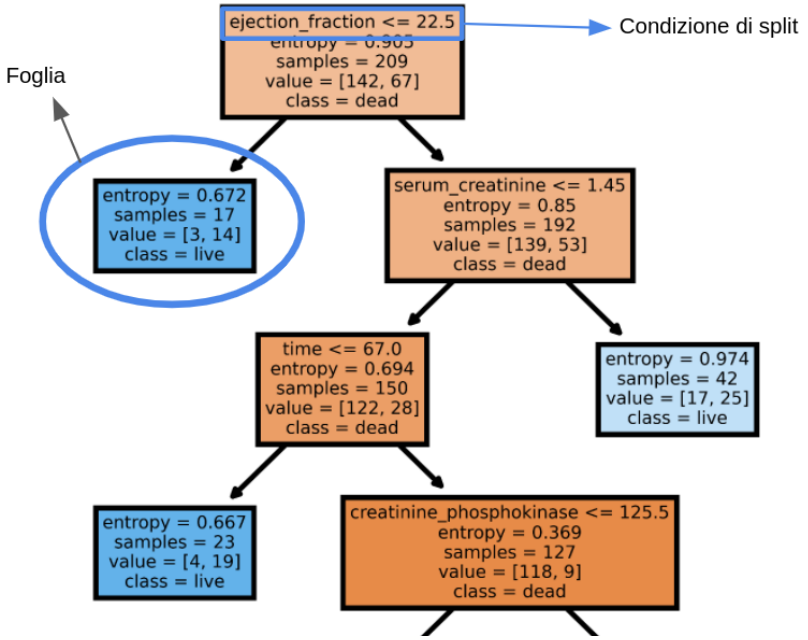


Figura 6-7: Porzione di un albero decisionale

Ad ogni nodo è inoltre associato un valore denominato *impurità.*

Un nodo si definisce "puro", se tutte le istanze del training set rientranti in esso appartengono alla stessa classe.

Questo valore può essere calcolato in diversi modi (*gini-index/entropy)*, ma il concetto dietro essi è il medesimo e spesso portano a risultati molto simili.

I metodi che fanno uso di alberi decisionali possono generalmente essere applicati sia per problemi di regressione, che per problemi di classificazione, come quello affrontato in questo progetto.

Questa tipologia di modelli (*decision tree*) ha il grande vantaggio di produrre risultati di facile interpretazione.

Tuttavia, un singolo *decision tree* di per sé risulta avere grossi limiti, e non è lontanamente paragonabile in termini di accuratezza delle predizioni, ad altri modelli di supervised learning.

Questo perché gli alberi decisionali sono estremamente sensibili ai dati su cui viene effettuato il training;

Più precisamente essi sono caratterizzati da un'alta varianza.

Questo significa che piccoli cambiamenti nei dati, possono portare a significativi cambiamenti nella struttura dell'albero, e quindi nelle predizioni. In altre parole, essi sono più propensi all'overfitting dei dati.

Sulle loro peculiari caratteristiche però si fondano le *Random Forests.*

Essi sono modelli basati sul cosiddetto *Ensemble Learning,* ovvero metodi che fanno un uso concomitante di diversi modelli di apprendimento al fine di ottenere migliori performance predittive rispetto all'utilizzare essi singolarmente.

I metodi rientranti in questa categoria ed in particolare le Random Forests, fanno uso dell'approccio denominato *Bootstrap aggregation* o *bagging.*

Esso consiste in una procedura finalizzata a ridurre la varianza degli metodi statistici di apprendimento.

Si osservi infatti che, dato un set di *n* osservazioni indipendenti $Z_1,...Z_n$, ognuno con una varianza σ^2, la varianza della media delle osservazioni $\overline{Z}$ è data da $\frac{\sigma^2}{n}$.

Detto in altri termini, mediare un set di osservazioni indipendenti tra loro, ne riduce la varianza.

Quindi, un modo per ridurre la varianza e di conseguenza aumentare l'accuratezza di un metodo di apprendimento è quelli di utilizzare diversi training sets, costruire per ognuno di essi un modello diverso, e mediare il risultati ottenuti da ognuno dei singoli modelli.

Dato che però solitamente non si possiede più di un training set, l'approccio migliore è quelli di generare B differenti training sets tramite *campionamento bootstrap.*

Questo consiste nell'ottenere diversi training sets tramite campionamento con rimpiazzamento.

Attenzione, non si otterranno quindi dei sottoinsieme del training set, in quanto i training sets derivanti dall'originale, avranno le sue stesse dimensioni.

Le Random Forests si basano su quanto spiegato e su un altro importante espediente finalizzato a decorrelare quanto più possibile gli alberi decisionali all'interno.

Questo consiste nel costruire, come spiegato, un numero più o meno grande di *decision trees,* ma per ogni split di ogni albero, si va a considerare solamente un sottoinsieme degli *m* predittori, anziché tutti, e quest'ultimo è scelto casualmente.

Quindi si otterrà che, per ogni split, si avrà un sottoinsieme diverso di predittori considerati. E solitamente la dimensioni del sottoinsieme è $m \simeq \sqrt{p}$, con p numero di predittori.

Questo perchè, anche se è vero che mediare i valori di molte osservazioni ne riduce la varianza, se esse sono fortemente correlate tra loro, questa riduzione risulta poco incisiva.

6.5.1 Implementazione modello e risultati

Anche per il modello Random Forest, come per gli altri visti precedentemente, è stato necessario effettuare una fase di ricerca e tuning degli hyperparameters.

Gli hyperparameters per cui si è cercata una combinazione ottimale sono:

- *n_estimators*: Ovvero il numero di alberi utilizzati all'interno;
- *max_features*: Il numero di features da considerare per la ricerca dello split ottimale;
- *max_depth*: La massima profondità o estensione dell'albero;
- *min_samples_split*: Il minimo numero richiesto di osservazioni o datapoint per effettuare uno split;
- *min_samples_leaf*: Il minimo numero di osservazioni richiesto per considerare un nodo come *nodo-foglia;*
- *Criterion*: Criterio o funzione necessaria a misurare la qualità o la purezza dello split.

Prima di visionare i risultati ottenuti, è possibile, grazie al modello applicato e alla libreria scikit-learn, andare a visualizzare quali features si sono rivelate più importanti nella classificazione tramite l'attributo *features_importance_.*

Il valore di importanza di ogni feature, viene calcolato come la media, sui diversi alberi costituenti, del decremento totale dell'impurità di un nodo (pesato per la probabilità di raggiungerlo) dovuta a quella determinata feature.

Quanto più una feature riesce a fornire splits *puri*, quanto più quella feature risulta importante.

Spesso in inglese ci si riferisce a questo valore come *mean decrease impurity,* e quanto più alto è, tanto più la feature è significativa.

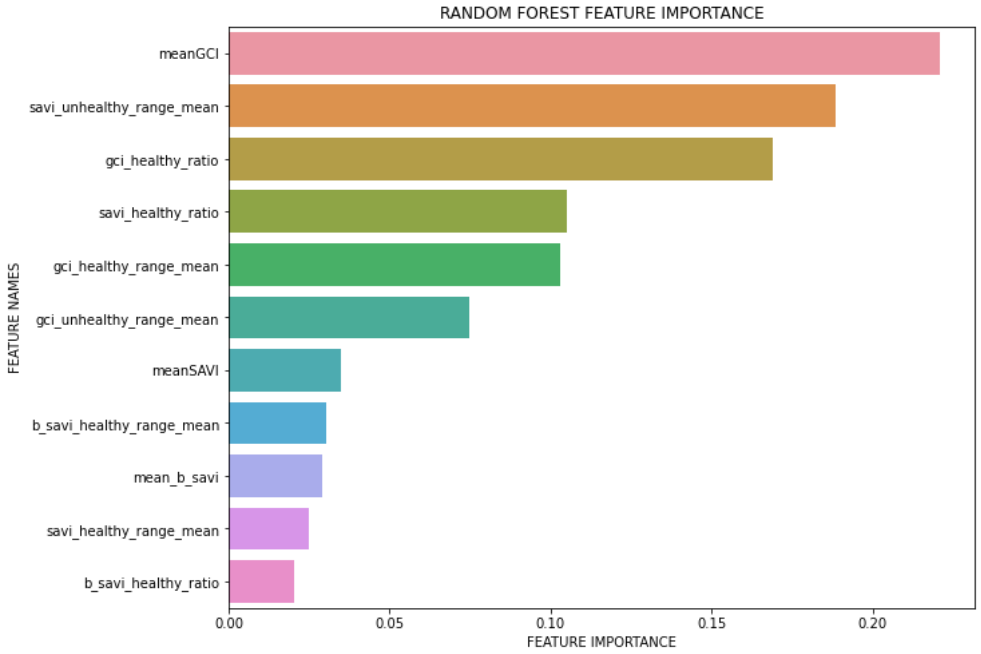


Figura 6-8: Valore di importanza per ogni feature

Visionando il grafico è possibile notare come cinque delle undici features risultino poco significative nella predizione dello stato di salute delle porzioni di pianta.

Si noti invece come abbia una grande rilevanza l'indice vegetazionale *GCI* insieme alle metriche su di esso basate.

È interessante inoltre notare come il valore medio del SAVI non risulti avere un valore di importanza elevato, ma le metriche *savi_healthy_ratio* e *savi_unhealthy_range_mean* si.

Nelle immagini che seguono è invece possibile andare a visualizzare i risultati ottenuti tramite il modello Random Forest.

	precision	recall	f1-score	support
0	0.90	0.62	0.73	29
1	0.83	0.96	0.89	54
accuracy			0.84	83
macro avg	0.86	0.79	0.81	83
weighted avg	0.85	0.84	0.84	83

Figura 6-9: Performance Random Forest

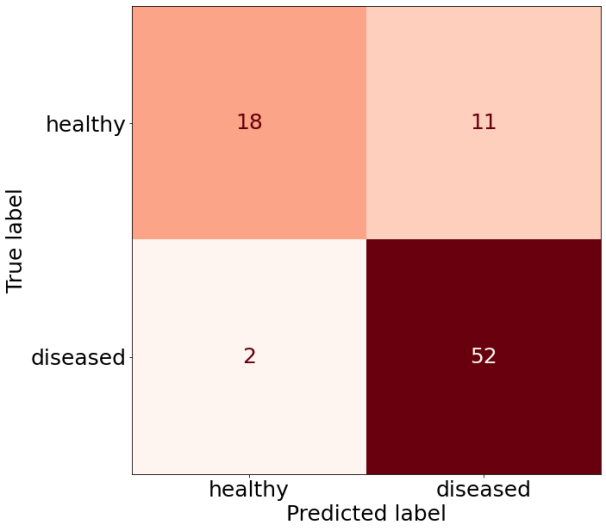


Figura 6-10: Matrice di confusione Random Forest

Dalle immagini si può osservare come le performance ottenute da quest'ultimo modello sia più elevate rispetto ai due precedenti. In particolare migliorano in modo marcato sia l'accuratezza generale del modello che il valore *f1*.

Nel paragrafo che segue, si comparano in ultimo i risultati dei tre modelli testati.

6.6 Comparazione modelli e osservazioni

Si è visto come, tutti è tre i modelli abbiano restituito risultati leggermente diversi tra loro, e volendone stabilire uno più promettente, il modello *Random Forest* è sicuramente quello con le performance migliori.
A seguire, con prestazioni non eccessivamente diverse, abbiamo il *logistic regression* e il modello basato su *SVM.*

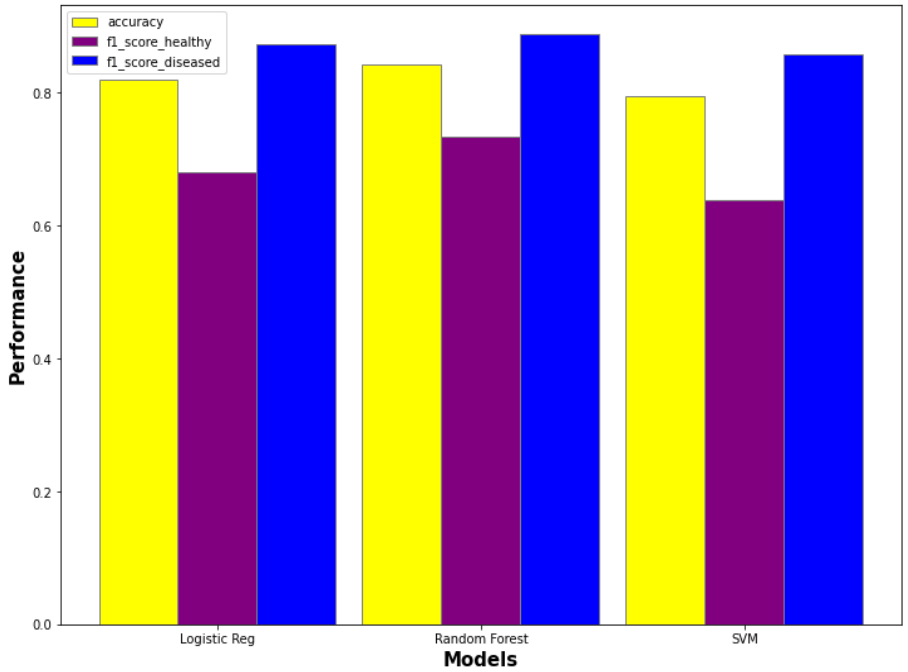


Figura 6-11: Confronto performance modelli

È sicuramente necessario osservare come tali performance possano differire al variare delle features fornite ai modelli durante la fase di training.

Nonostante non siano stati riportati tutti i vari tentativi, durante gli svariati test effettuati, è doveroso sottolineare come si sono provate diverse combinazioni di features.

Come infatti abbiamo visto, alcune di esse risultano fortemente correlate tra loro, e plausibilmente alcune potrebbero essere rimosse.

Nonostante ciò, questa osservazione necessita comunque di diverse fasi utili al suo accertamento.

Inoltre come si è potuto notare dalla *figura 6-8* riportante l'importanza di ogni feature, alcune di esse risultano avere poco peso decisionale in fase di classificazione, e mentre nel migliore dei casi non apportano alcun contributo, nel peggiore dei casi possono sviare e portare a risultati leggermente inferiori, oltre che comunque aumentare la complessità computazionale.

Alla fine nonostante si siano effettivamente creati (e salvati) dei modelli il cui training è stato effettuato solo su alcune features, dato che le

performance differivano solo di qualche punto percentuale per ogni modello, si è preferito riportare i risultati di quelli che in input ricevono tutte le features.
Questo lo si è deciso anche per le limitazioni derivanti dal dataset, che risultando comunque di dimensioni piuttosto limitate, rende rischioso inferire che alcune features non siano effettivamente utili.

Infine, nella figura che segue è possibile visionare riassuntivamente, l'intero processo descritto, che consiste partendo dalle foto delle piante nei seguenti step:

1. Suddivisione delle foto in sotto-riquadri
2. Calcolo degli Indici Vegetazionali su ognuno dei riquadri
3. Estrazione delle diverse metriche e salvataggio su un documento
4. Applicazione dei modelli di classificazione con in input le metriche.

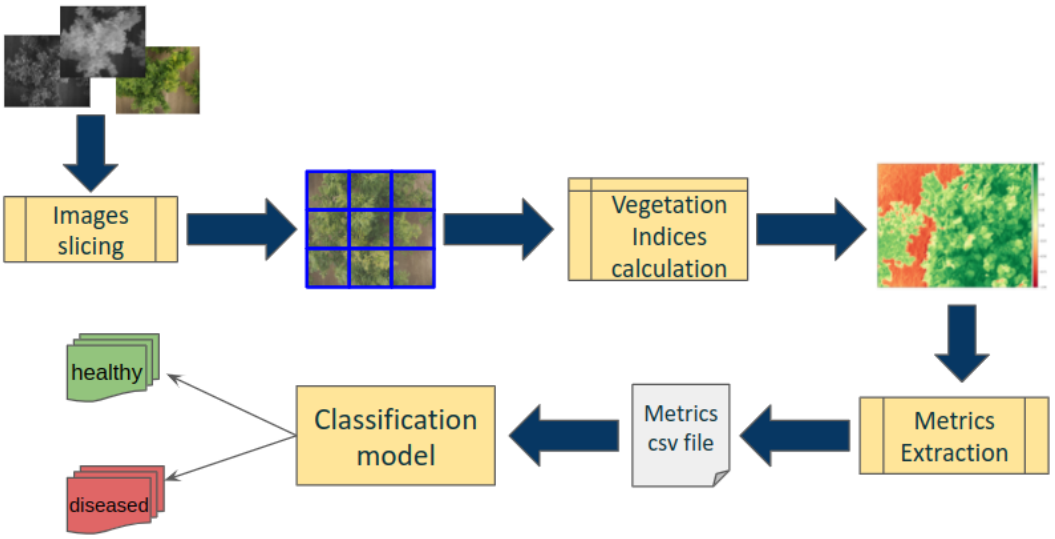


Figura 6-12: Processo di classificazione

Un ulteriore e ultimo confronto che si vuole mostrare è quello che vede non più l'uso degli Indici Vegetazionali in accoppiata ai modelli visti in questo capitolo, ma bensì l'utilizzo di reti neurali con in input le foto contenenti le porzioni di piante.
Questo approccio verrà illustrato nel capitolo che segue.

Capitolo 7

7.1 Un diverso approccio - Reti Neurali

Nei capitoli precedenti si sono illustrati i pro e i contro di un approccio che fa uso degli Indici Vegetazionali.
In particolare abbiamo visto che essi generalmente vengono utilizzati in accoppiata ad un'ispezione visiva, permettendo di vedere, tramite combinazione di immagini multispettrali, ciò che il semplice occhio umano fa fatica a cogliere.
Lo scopo però del progetto è quello di automatizzare il processo di analisi, e far sì che l'ispezione da parte di personale con competenze nel settore insieme alla necessità di costose strumentazioni non sia più necessaria, se non per eventuali conferme.

Questo ha quindi portato al calcolo di metriche basate sugli IV e la successiva applicazione di algoritmi di Machine Learning.
Tutto questo processo è quindi composto da diverse fasi, riassunte e illustrate in *figura 6-12.*
A titolo di confronto però, in questo capitolo si vuole mostrare un altro approccio, che non fa più uso di indici vegetazionali accoppiati a modelli *classici* di machine learning, ma bensì di reti neurali artificiali applicate direttamente sui ritagli.
In particolare, le tipologie di reti che si vogliono applicare prendono il nome di *CNNs*, *Convolutional Neural Networks.*
Esse fanno parte del ventaglio di tecniche e algoritmi appartenenti al campo del *deep learning* che risultano particolarmente utili in problemi riguardanti, riconoscimento di immagini e video, classificazioni e segmentazione di immagini, analisi di immagini mediche ecc.

Come detto precedentemente, non si vuole in questa sede scendere nei dettagli implementativi degli diversi algoritmi, pertanto, di seguito si espongono solamente le componenti principali di un'architettura di reti di tipo CNN.

7.2 CNN - Convolutional Neural Network

Come detto poco sopra, le reti di tipo CNN, sono una particolare tipologia di rete neurale artificiale, la cui architettura è basata su quella della corteccia visiva.

Brevemente, ricordiamo che una rete neurale artificiale è un sistema computazionale il cui funzionamento è basato sulle reti neurali biologiche costituenti la struttura anatomica del cervello.

Essa è generalmente strutturata su diversi livelli, costituiti da delle unità basilari, denominati neuroni artificiali, connessi in modo particolare tra loro.

Ad ogni connessione è associato un parametro/peso numerico e ad ogni neurone è invece associata una *funzione di attivazione,* assimilabile concettualmente ad funzione di trasferimento.

I diversi livelli in base alla loro profondità vengono denominati in modi differenti. Il primo livello, ovvero quello che permette l'input delle informazioni da analizzare, prende il nome di *input layer.*

Intuitivamente, l'ultimo livello, ovvero quello il cui compito è restituire un risultato o una predizione, prende il nome di *output layer.*

I diversi livelli posizionati tra i due appena esposti prendono il nome di *hidden layers*.

Tutti i livelli, costituiti da una quantità ben definita di neuroni, sono *completamente connessi* o *fully connected* tra di loro, ovvero, ogni singolo neurone è connesso a tutti quelli del layer successivo, e così per ogni layer, fino a giungere all'output layer.

Quando questa struttura raggiunge un'elevata profondità prende il nome di *deep neural network;* Queste reti particolari sono il fulcro di quel sottoinsieme di algoritmi e tecniche di Machine Learning che prende il nome di *Deep Learning.*

Su queste reti l'algoritmo fondamentale che viene eseguito è quello di *backpropagation*, che al fine di ridurre l'errore della rete, ne calcola il gradiente al fine di aggiornare ed ottimizzare i parametri/pesi di ogni connessione.

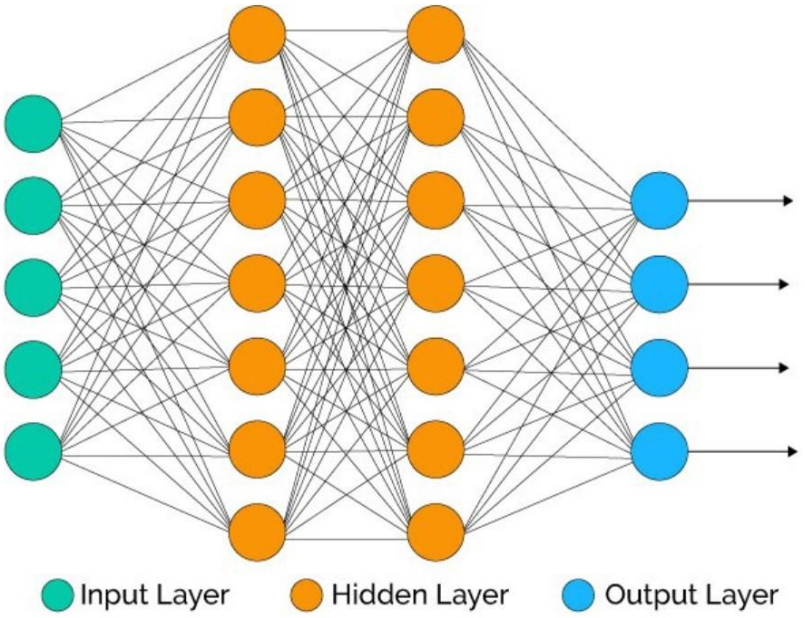


Figura 7-1: Esempio di rete neurale con i diversi livelli

Le CNN nascono dall'esigenza di affrontare tipologie di problemi che utilizzando le *deep neural networks,* caratterizzate da *fully connected layers,* sarebbe impossibile affrontare.

Questo perché un'architettura *fully connected*, applicata all'analisi di una seppur piccola immagine (*es 100x100-pixel*) porterebbe ad un numero spropositato di neuroni e connessioni tra essi e di conseguenza ad una complessità computazionale non gestibile.

Le CNN risolvono questo problema usando dei cosiddetti *partially connected layers*, ovvero livelli solo parzialmente connessi.

Le convolutional neural networks fanno generalmente uso di 3 blocchi:

- Convolutional layers
- Pooling layers
- Fully connected layers

Il blocco più importante è senza dubbio il cosiddetto *convolutional layer*.
Esso è basato su un'operazione matematica denominata *convoluzione*, che coinvolge due funzioni e consiste nell'integrare il prodotto tra la prima funzione e la seconda traslata di un certo valore.
Trova ampio uso nella teoria dei segnali. (*In realtà l'operazione realmente usata è quella di Correlazione Incrociata, ma le due sono estremamente simili, ndr*).
Nei *convolutional layers,* i neuroni non sono tutti connessi tra loro tra i diversi livelli, ma bensì ogni neurone sarà connesso solo a quelli rientranti nel suo *campo recettivo,* immaginabile come una ristretta area rettangolare. Questo ricorsivamente anche per i layers successivi.
Ad ogni livello convoluzionale, viene accoppiato un filtro, ovvero un set di pesi(valori) apprendibili (chiamati anche *convolution kernels*), che verranno coinvolti nell'operazione di convoluzione con l'input e che hanno lo scopo di evidenziare determinate features dell'immagine.
Come risultato dell'operazione ripetuta sui diversi layers (*forward pass*), la rete apprenderà dei filtri sempre più articolati e complessi, che si attiveranno alla presenza di particolari tipologie di features.
Più in profondità si va nella rete, più le features evidenziate dai filtri saranno complesse; ad esempio, tramite i filtri iniziali, la rete sarà in grado di determinare linee verticali o orizzontali, mentre tramite i filtri più in profondità essa sarà in grado di riconoscere foglie, rami e terreno.
Come risultato di ciò, un layer contenente tutti neuroni caratterizzati dallo stesso filtro, restituisce in output una cosiddetta *feature map,* in grado di evidenziare nell'intera immagine, le aree che più attivano quel determinato filtro.

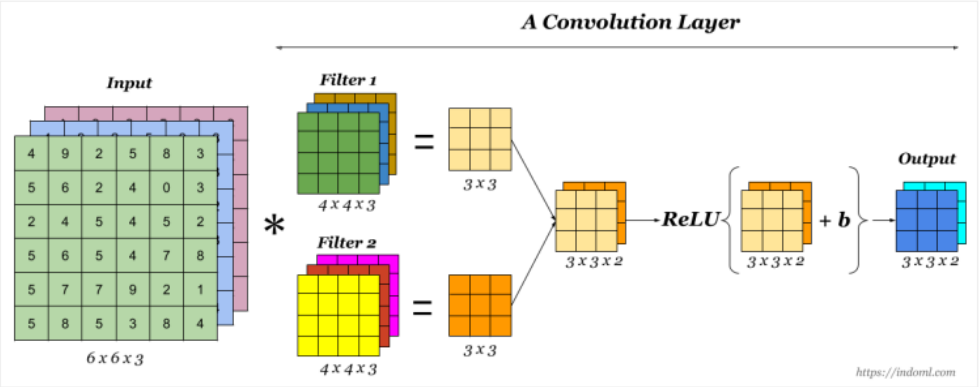


Figura 7-2: Operazione di convoluzione in un convolutional layer

Da attenzionare che i filtri non vengono definiti manualmente, ma bensì, come già esposto, vengono autonomamente appresi dalla rete.

L'ulteriore componente fondamentale nelle reti convoluzionali è il *pooling layer*. Esso ha uno scopo molto semplice, ovvero quello di sottocampionare l'input in modo da, non solo ridurre il carico computazionale e l'uso di memoria, ma anche il numero di parametri da apprendere. Questo in parte ha l'effetto anche di limitare il rischio di overfitting.

Proprio come nel *convolution layer*, i neuroni all'interno del *pooling layer* sono connessi solo ad un numero di neuroni limitato del livello precedente. In parole semplici, osservano solo i neuroni appartenenti al loro *campo recettivo*.

A differenza però dei neuroni appartenenti ai *convolution layers*, quelli che fanno parte di questo livello non possiedono dei pesi o dei parametri numerici che debbono essere appresi, ma si occupano semplicemente di aggregare le informazioni provenienti dal layer precedente, tramite una funzione di aggregazione che può essere una funzione di *max* o di *media.*

Infine, tipicamente nelle architetture CNN, viene aggiunta una rete neurale *classica,* composta da alcuni livelli *fully connected* caratterizzati quindi anche da una determinata funzione di attivazione e da un *output layer* per la predizione finale.

In generale, come vedremo a breve, le architetture CNN fanno uso di molti di questi *layers* appena esposti, posizionati sequenzialmente, in

modo da formare un vero è proprio *stack* di livelli come visibile nell'immagine seguente.

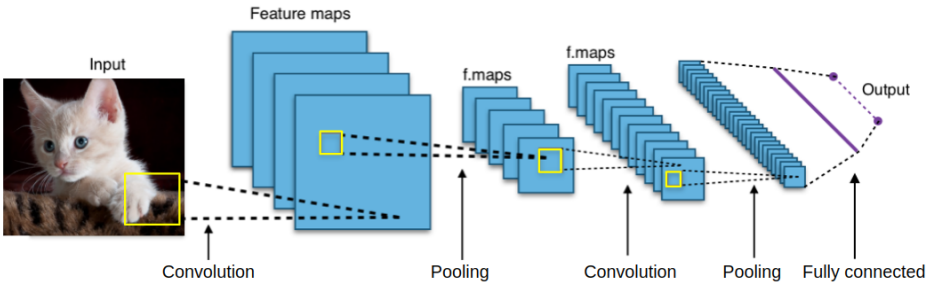


Figura 7-3: Tipica architettura CNN

7.3 Modelli classici vs Deep Learning

Esistono diversi pro e contro nell'utilizzo di tecniche di deep learning, rispetto ai classici algoritmi di machine learning, quali ad esempio *logistic regression o SVM.*

In questo paragrafo si vuole spiegare in cosa differiscono i due approcci e quando è effettivamente consigliato utilizzare l'uno piuttosto che l'altro.

È necessario subito evidenziare che i modelli di deep learning, in generale, presentano performance di tutt'altro livello rispetto ai classici modelli di ML.

Questo è estremamente evidente in campi particolarmente complessi quali: *Computer Vision*, NLP (*Natural Language Processing*), *speech recognition* ecc.

Questi campi hanno tutti un minimo comune denominatore: L'enorme quantità di dati.

Un problema che affligge i classici modelli di ML, è legato al fatto che le performance dei suddetti non scalano all'aumentare della mole di dati in input.

Le *deep neural networks*, scalano molto meglio in confronto.

Spesso infatti, per migliorare le performance delle reti neurali basta semplicemente avere più dati a disposizione.

Un grafico che spesso viene utilizzato per mostrare queste differenze è quello mostrato nella figura seguente.

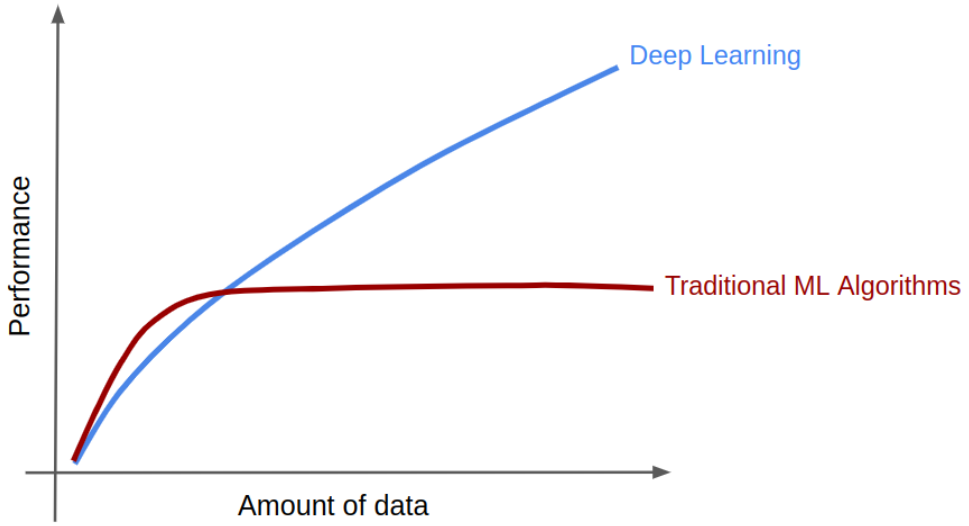


Figura 7-4: Confronto performance tra Deep Learning e algoritmi classici di ML

Un'altra caratteristica a vantaggio del *deep learning* è legata al lavoro di *feature engineering* che è spesso necessario effettuare nelle fasi precedenti all'applicazione di algoritmi di ML classici.

Come abbiamo infatti visto nel capitolo 5, spesso è necessario effettuare un'analisi delle features a disposizione, andandone a studiare ad esempio la correlazione tra esse, e la loro importanza nella classificazione, ed eventualmente decidere se andare ad applicare degli algoritmi di riduzione della dimensionalità (es: PCA).

All'aumentare infatti delle features nel dataset, la complessità computazionale degli algoritmi classici di ML aumenta esponenzialmente, e, se nel migliore dei casi questo comporta soltanto tempi di elaborazione più lunghi, nel peggiore dei casi può portare a risultati piuttosto scadenti in termini di accuratezza.

Questo problema in inglese, prende il nome di *Curse of dimensionality*.

Per quanto concerne invece il *deep learning*, questo problema è molto meno presente, e nella maggioranza dei casi tutte le features vengono date in input al modello.

Un ultimo grande vantaggio delle tecniche di *deep learning*, è la loro grande flessibilità e adattabilità a problemi di varia natura.
Un esempio di questa grande flessibilità è dato dal *Transfer Learning*, una branca del machine learning che si focalizza sul conservare e riutilizzare le conoscenze acquisite da un modello, relative ad uno specifico problema, su un diverso problema appartenente allo stesso dominio.
L'esempio più famoso coinvolge il dataset *ImageNet* e l'ampio set di modelli *pre-addestrati* su di esso.
Essi consistono in modelli che sono stati allenati nella classificazione di centinaia di migliaia di immagini, appartenenti al dataset ImageNet, la cui "conoscenza" può essere trasferita ed utilizzata ad esempio nella classificazione di immagini non appartenenti ad ImageNet, o anche all'*object detection* all'interno di un'immagine.
Come vedremo meglio dopo, ImageNet non include immagini di alberi di nocciolo sani o malati, ma questo non vieta di utilizzare le informazioni che questi modelli hanno appresso da esso, al fine di costruire un modello in grado di effettuare quella classificazione.
Questo perché il *transfer learning* si basa su features comuni tra i diversi problemi e mentre un modello costruito da zero, dovrà andare ad apprendere le più piccole e ripetitive features, come gli i bordi verticali o orizzontali di un oggetto all'interno di un'immagine, un modello *pre-allenato*, saprà già effettuare distinzioni di più alto livello, come riconoscere all'interno delle immagini, quelle features relative ad esempio a delle foglie.
Questo non solo permette di risparmiare un'enorme quantità di tempo per la fase di training, ma il più delle volte permette un grande aumento delle performance finali.
Ed in ultimo, il *transfer learning* consente di ridurre drasticamente l'ammontare di dati necessario al training, proprio perché gran parte delle informazioni non sarà necessario apprenderle nuovamente.

È necessario però andare ad evidenziare che i modelli di *deep learning* presentano anche degli svantaggi rispetto ai classici modelli di machine learning.

Primo tra tutti, come si è appreso dalla *figura 7-4,* questi modelli performano peggio quando si hanno a disposizione pochi dati.

In particolare per raggiungere alte performance, vediamo dall'immagine che i modelli di *deep learning* necessitano di datasets molto vasti.

Si pensi che i modelli *pre-trained* accennati precedentemente, sono stati allenati su centinaia di migliaia di immagini.

Nonostante oggigiorno i dati siano disponibili in grosse quantità per alcuni ambiti, per altri invece tali dimensioni per un dataset sono difficilmente raggiungibili.

Si pensi che il dataset a nostra disposizione, contenente le foto degli alberi (non i ritagli, ma l'immagine intera), consiste in poco più di 200 foto RGB, insufficienti per allenare da zero un modello CNN.

Per questo. come vedremo, si approccerà il problema facendo sia uso del *transfer learning* che di tecniche di *data augmentation*.

Da non trascurare inoltre che i modelli classici di ML, richiedono molte meno risorse, in termini computazionali, rispetto ai modelli CNN.

Se per il training e l'utilizzo di un modello classico può andar bene una buona CPU, per un modello CNN è spesso necessario, se non indispensabile, avere a disposizione potenti GPU e un buon quantitativo di RAM.

Questo si traduce in tempi di training estremamente più lunghi, rendendo molto complesso ed oneroso anche il tuning e il testing delle diverse combinazioni di *hyperparameters*.

In ultimo, i modelli classici, contrariamente ai modelli di *deep learning,* sono spesso di più facile interpretazione e permettono ad esempio di andare a capire quali features incidono di più in una determinata classificazione e il perché.

7.4 Preparazione e scelta delle architetture

Come già ampiamente esposto nell'elaborato, purtroppo il dataset a disposizione non è molto vasto.

Tuttavia, come visto nel paragrafo precedente, i modelli di *deep learning* performano estremamente meglio in presenza di grandi quantitativi di dati.

Questo comporta che, per l'applicazione di reti di tipo CNN, è stato necessario fare uso di due tecniche, che andranno per quanto possibile, a mitigare questo problema, ovvero *transfer learning* e *data augmentation*.

La prima tecnica, già brevemente esposta nel paragrafo precedente, ci permette, di trasferire ed utilizzare le informazioni apprese da dei modelli già ampiamente addestrati, al fine di risolvere un diverso problema appartenente però allo stesso dominio.

Per quanto riguarda le CNN, specialmente grazie all'organizzazione di competizioni internazionali, sono stati fatti grandi passi avanti nello sviluppo di nuove architetture fondamentali.

In particolare ILSVRC è una competizione che prevede di classificare e rilevare correttamente oggetti e scene contenuti in immagini appartenenti ad un vastissimo dataset (*http://image-net.org*), ed essa, assieme alla grande ricerca svolta in questo campo ha portato alla nascita di diverse architetture, come: GoogLeNet, VGGNet, ResNet, Xception ecc.

Nell'immagine sottostante, possiamo vedere i diversi modelli, pre-allenati, messi a disposizione da Keras.

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4

Figura 7-5: Alcuni dei modelli pre-trained disponibili su Keras

Ovviamente date le risorse a disposizione, sia in termini computazionali che in termini di tempo, non è possibile provare un grande numero di architetture, questo perché, come vedremo, le fasi caratterizzanti il processo di *transfer learning* e di training, risultano molto lunghe e complesse.

Inoltre ogni combinazione di *hyperparameters* è caratterizzata da molte variabili (*scelta optimizer, epoche, learning rate ecc..*) e a differenza dei modelli classici, per essere testata, ognuna richiede una grande potenza computazionale e di conseguenza tempi molto più lunghi.

Per questo motivo, si è deciso di provare due delle architetture più conosciute ed utilizzate: ResNet e VGG, nella loro versione *ResNet50 e VGG16.*

Le immagini su cui questi modelli sono stati addestrati, sono i ritagli RGB contenenti le porzioni di piante.

In questo modo sarà possibile fare un paragone diretto tra le performance raggiunte dai modelli classici, e quelle invece raggiunte dalle reti in questione.

7.5 Creazione modelli e training

Prima di passare alla creazione dei modelli e l'effettivo processo di *transfer learning*, è stato necessario suddividere il dataset in tre diversi set: Training set, Test set e Validation set.

A tale scopo, si è scelto di suddividere il dataset secondo le seguenti proporzioni:

Training set: 70%, Test set: 15%, Validation set: 15%.

Similmente a quanto fatto per i modelli classici, si è deciso di mantenere nei diversi set, il rapporto tra ritagli sani e malati che caratterizza l'intero dataset.

Suddiviso il dataset nei tre rispettivi set, si è proceduto con la configurazione dell'oggetto *ImageDataGenerator,* messo a disposizione dal modulo di *Tensorflow.*

Esso non solo consente la lettura delle foto dal disco e il loro trasferimento come input ai modelli, ma permette l'applicazione di diverse strategie di *data augmentation.*

Esse consentono di incrementare il quantitativo di dati, creando delle copie delle foto che però differiscono in una o più caratteristiche.

Questo avrà un effetto di regolarizzazione del modello, andrà in un certo senso ad incrementare le dimensioni del dataset e di conseguenza aiuterà a ridurre l'overfitting.

Si sono applicati diverse tipologie di data augmentation come:

flipping orizzontale della foto, variazione casuale della luminosità, rotazione casuale dell'immagine.

In questo modo, per ogni epoca di training, il modello non riceve in input sempre lo stesso set di foto, ma bensì una copia di esso che però differisce in diverse caratteristiche.

Questo renderà più robusto il modello migliorando le sue capacità di generalizzazione.

Figura 7-6: Esempio di rotazione casuale per data augmentation

Altre tipologie molto utilizzate di data augmentation prevedono ad esempio lo zoom o il ritaglio casuale di porzioni della foto.
Queste ultime si è scelto di non applicarle, in quanto, potrebbero effettivamente andare a nascondere le caratteristiche che vanno a definire lo stato di salute della porzione di pianta.
Si pensi ad esempio ad un ritaglio contrassegnato come malato, ma caratterizzato solo da una piccola porzione vegetativa effettivamente sotto stress. Un eventuale zoom casuale potrebbe nascondere la porzione malata e portare ad un errato apprendimento da parte del modello.
Mentre ad esempio, si è reputato importante variare casualmente la luminosità delle foto, in quanto questa potrebbe effettivamente essere analoga alla variazione delle condizioni di luminosità durante gli scatti.

La fase successiva invece ha riguardato strettamente il processo di transfer learning.
Esso ha visto primariamente l'istanziazione del modello già addestrato. Fondamentale specificare in questa fase di costruzione che si vogliono importare i pesi di ImageNet, tramite il parametro `weight = 'imagenet'`.
Sempre in fase di costruzione, si è andati ad escludere l'ultimo layer del modello, questo perché esso successivamente verrà sostituito con un nostro layer, che quindi sarà addestrato per il nostro specifico problema.
Ovviamente non si vuole riaddestrare l'intero modello, ed infatti il

settaggio dell'attributo `trainable = False` permette di congelare i valori dei pesi del modello originale.

Al di sopra di questo modello, si è quindi aggiunto quello da noi definito, composto semplicemente da un *pooling layer* per la trasformazione dell'output del modello base in un vettore, e da un livello di tipo *Dense* o *Fully connected*.

Tra i due livelli appena esposti è stato aggiunto anche un *Dropout layer*, questo permette di regolarizzare il modello, diminuendo la probabilità che esso possa andare in overfitting.

Nell'immagine seguente possiamo vedere sommariamente come è strutturato il modello risultante, in questo caso il VGG16.

```
model.compile(tf.keras.optimizers.Adam(learning_rate=0.005),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=[tf.keras.metrics.BinaryAccuracy(), tf.keras.metrics.MeanSquaredError()])
# Print The Summary of The Model
model.summary()
```

```
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_9 (InputLayer)        [(None, 224, 224, 3)]     0

 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 global_average_pooling2d_2   (None, 512)              0
 (GlobalAveragePooling2D)

 dropout_2 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 1)                 513

=================================================================
Total params: 14,715,201
Trainable params: 513
Non-trainable params: 14,714,688
```

Figura 7-7: Modello basato su VGG16

7.5.2 Training e fine-tuning

La fase di training ha visto la prova, per entrambi i modelli, di diverse combinazioni di: Optimizer, Learning Rate e numero di epoche.

È da sottolineare come, a differenza dei modelli classici di ML, qui la ricerca di una combinazione ottimale di hyperparameters sia enormemente più complessa, e richieda tempi molto più lunghi. Per questo, se non con hardware e potenza computazionale adeguata, è estremamente complesso ad esempio applicare l'approccio *GridSearch* che si è visto applicato per i modelli classici.

In questo caso la ricerca, che ha comunque richiesto diversi giorni, è stata effettuata tramite tentativi. Partendo da una combinazione ragionevole del modello, e andando a variare i valori dei suoi parametri in base alle sue performance.

Ovviamente questo processo ha previsto di salvare di volta in volta il modello che precedentemente aveva portato a risultati validi.

Una volta trovata un combinazione che restituisse dei risultati soddisfacenti, o comunque una volta notata una certa stazionarietà nell'andamento delle performance è stato possibile procedere ad un'altra fase del processo di transfer learning, denominata *fine tuning*.

Essa prevede l'attivazione dell'apprendimento di tutti o alcuni livelli del modello base, quindi il parametro prima settato come `trainable = False`, adesso verrà impostato come `trainable = True`.

Quando si effettua questa operazione è necessario portare molta attenzione e settare un learning_rate piuttosto basso, altrimenti si rischia di sporcare le informazioni apprese nella fase precedente.

```
feature_extractor.trainable = True
# Let's take a look to see how many layers are in the base model
print("Number of layers in the base model: ", len(feature_extractor.layers))
```

```
Number of layers in the base model:  19
```

```
# Fine-tune from this layer onwards
fine_tune_at = 15
# Freeze all the layers before the `fine_tune_at` layer
for layer in feature_extractor.layers[:fine_tune_at]:
  layer.trainable = False
model.compile(tf.keras.optimizers.Adam(learning_rate=0.000001),
              loss=tf.keras.losses.BinaryCrossentropy(),
              metrics=[tf.keras.metrics.BinaryAccuracy(), tf.keras.metrics.MeanSquaredError()])
```

Figura 7-8: Esempio di fine tuning su alcuni layers del modello base

Nelle immagini che seguono è possibile visionare per entrambi i modelli l'andamento dell'accuratezza relativo sia al training che al test set, durante la fase di apprendimento.

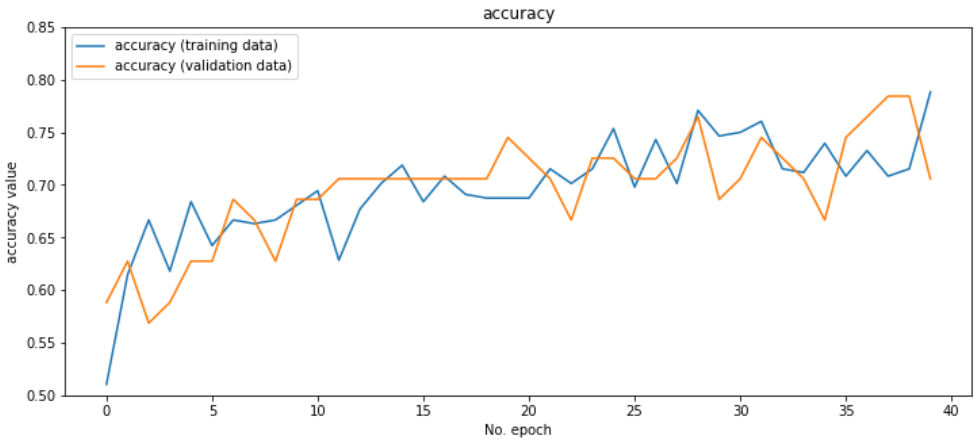


Figura 7-9: Andamento accuratezza del modello basato su ResNet50

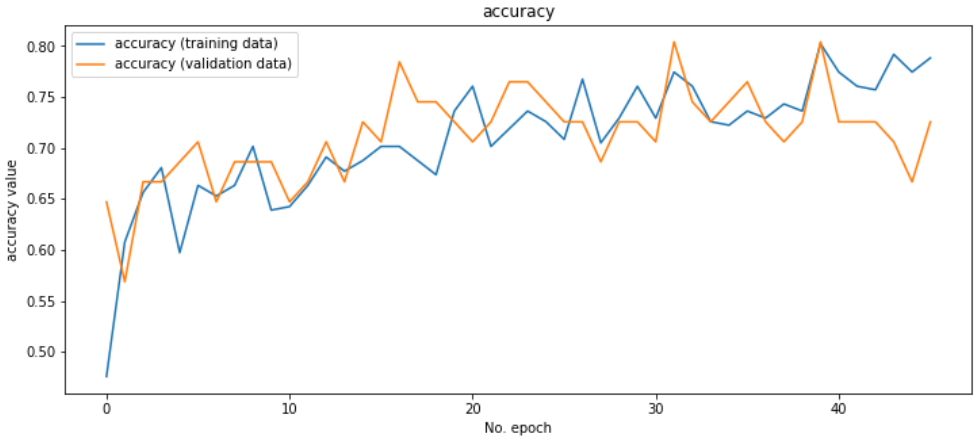


Figura 7-10: Andamento accuratezza del modello basato su VGG16

Mentre di seguito è possibile visionare i risultati ottenuti sul test set, rispettivamente, dal modello basato su ResNet50 e su quello basato su VGG16.

	precision	recall	f1-score	support
diseased	0.87	0.94	0.91	36
healthy	0.83	0.67	0.74	15
accuracy			0.86	51
macro avg	0.85	0.81	0.82	51
weighted avg	0.86	0.86	0.86	51

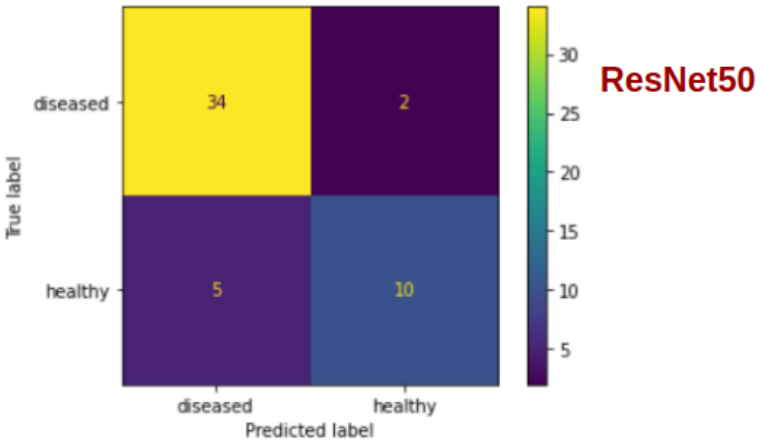


Figura 7-11: Performance ResNet50

	precision	recall	f1-score	support
diseased	0.86	0.89	0.88	36
healthy	0.71	0.67	0.69	15
accuracy			0.82	51
macro avg	0.79	0.78	0.78	51
weighted avg	0.82	0.82	0.82	51

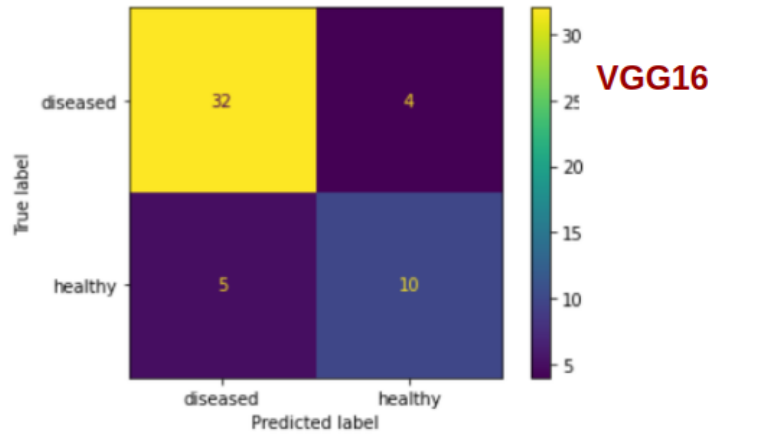


Figura 7-12: Performance VGG16

Come è possibile notare, il modello basato su ResNet50 risulta essere discretamente più performante, ovviamente non solo in termini di accuratezza, ma anche di F1-Score.

Infine, qui di seguito è illustrato il grafico delle performance in termini di F1-Score e accuratezza, di tutti i modelli, compresi quindi anche quelli classici, visti nella parte precedente dell'elaborato.

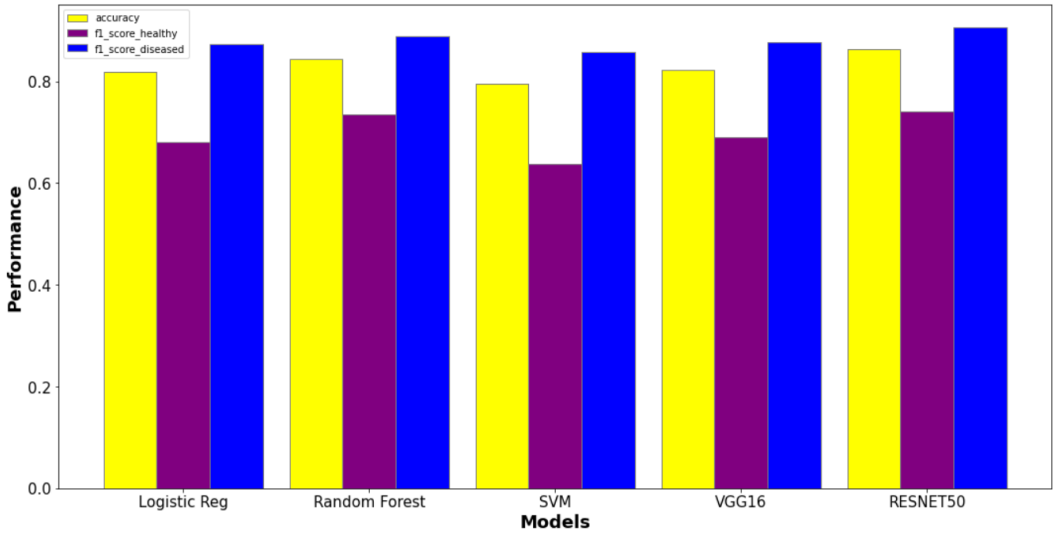


Figura 7-13: Comparativa prestazione di tutti i modelli esaminati

Osservazioni finali e conclusioni

Dalla *figura 7-13* è possibile notare come, approcci molto distanti e diversi tra loro, abbiano portato a risultati non altrettanto differenti.

In termini di accuratezza, tutti i modelli hanno prodotto valori $\geq 80\%$. In particolare il modello basato su Random Forest è quello che performa meglio tra gli algoritmi classici con un'accuratezza che si attesta sull'84%, mentre il ResNet50 è quello che performa meglio tra tutti i modelli.

Ci sono però importanti osservazioni da fare:

Una prima precisazione è necessaria riguardo i modelli CNN. Questi modelli, come visto nel capitolo dedicato, necessitano di una grande quantità di dati, non solo per performare bene, ma anche al fine di restituire risultati più affidabili possibile.

Quanti più dati essi hanno da cui apprendere, tanto più saranno in grado di generalizzare. Se il quantitativo di dati scarseggia, e il training non viene effettuato correttamente, il modello potrebbe essere estremamente soggetto ad overfitting, e questo, data la scarsità di dati su cui esso viene testato, potrebbe non palesarsi subito, ma in un secondo momento.

Per ovviare a questo problema sono state applicate diverse tecniche, sia di *data augmentation* che a livello architetturale, aggiungendo livelli di tipo *Dropout.*

Ciò non toglie che il miglior modo per ovviare a queste problematiche resta comunque quello di fornire a questi modelli, particolarmente complessi, quanti più dati possibile.

Questo si traduce, in una successiva fase, nell'esigenza di aumentare il numero di foto raccolte e quindi il numero di piante prese in esame, al fine di creare un dataset quanto più vasto e vario possibile.

Questo problema si presenta in maniera discretamente ridotta, nel caso in cui si applichino algoritmi di ML classici, come quelli visti nella prima parte dell'elaborato, utilizzati assieme ad un altro strumento estremamente utile, ovvero gli Indici Vegetazionali.

Sotto diversi punti di vista, l'applicazione del primo approccio apporta molti vantaggi.
In primis, come già detto, questi algoritmi sono meno propensi all'overfitting quando la quantità di dati è ridotta.
Inoltre esso risulta essere molto flessibile e soprattutto "estensibile".
Infatti, dagli indici vegetazionali sono state estratte delle metriche. Questo significa che successivamente, anche con l'ausilio di personale esperto, potrebbe essere possibile ideare ed applicare nuove metriche che quantificano altri aspetti degli indici vegetazionali in uso. Ma non solo.
È interessante sottolineare infatti, come sia estremamente semplice integrare ulteriori indici vegetazionali nell'algoritmo, basati ad esempio su altre bande oltre a NIR, Red-Edge ed RGB.
Un esempio potrebbe essere quello di includere indici vegetazionali basati sulla banda dello SWIR (*Short Wave Infrared*), che risultano particolarmente efficaci nell'evidenziare il contenuto di acqua nelle foglie.
Questo permette di includere ulteriori dati in ingresso ed avere una classificazione più fine, oltre che in futuro, poter identificare anche la causa dello stress vegetativo.
Inoltre, mentre i modelli CNN, come ResNet50, ci restituiscono solo il risultato o la classe finale derivante dalla previsione, il primo approccio permette la visualizzazione dei valori delle metriche che caratterizzano quella determinata pianta. Ed insieme alle immagini relative ai diversi indici, è possibile effettuare anche ulteriori analisi e osservazioni che potranno apportare informazioni utili sulle azioni da intraprendere.

Un'ultima osservazione va fatta sull'applicazione delle labels relative alle diverse immagini.
Questa, essendo effettuata principalmente tramite occhio umano, potrebbe contenere una componente di soggettività.
Il problema è più evidente quando gruppi di piante diverse risultano avere condizioni estremamente diverse tra loro.

Ad esempio, si potrebbe essere portati ad essere più critici rispetto ad alcuni gruppi di piante, dove le condizioni di salute sono tutto sommato buone, e meno critici in gruppi di piante che si trovano tutte in condizioni di stress vegetativo.
È fortemente consigliato, in una successiva fase, procedere all'analisi e alla seguente applicazione delle labels, facendo in modo che le stesse immagini vengano etichettate da un team di persone e non da una singola.
Questo per far sì che eventuali errori umani, o valutazioni involontariamente soggettive, si attenuino, aumentando mediamente non solo l'accuratezza delle labels, ma anche quella dei modelli che dovranno da esse apprendere.

In conclusione, l'applicazione sviluppata ha ottenuto risultati molto significativi, consentendo, a partire da immagini multispettrali, di **ottenere una classificazione di pianta sana/malata con un'accuratezza media intorno all'85% e permettendo inoltre l'individuazione delle porzioni di pianta in sofferenza.**
Inoltre esso sicuramente si presta ad un'ulteriore ed importante crescita.
Basti pensare alle tecnologie messe oggi a disposizione, alle tante tipologie di sensori esistenti, e di conseguenza alla grande mole di informazioni da cui questi modelli possono apprendere.
Quindi eventuali sviluppi futuri dovrebbero sicuramente prevedere:

- L'utilizzo di sensori più performanti, con una maggiore risoluzione per maggior dettaglio, ed in grado di catturare informazioni anche in altre bande (*Vedi sensori iperspettrali*);
- Nuovi indici vegetazionali che possano beneficiare di altre bande come ad esempio la SWIR;
- Uno scatto più accurato delle foto, in particolare attenzionando bene condizioni di luminosità e bilanciando il più correttamente possibile l'esposizione;

- Un’applicazione più fine delle labels da parte di un team dedicato;
- Un numero maggiore di foto acquisite, che possa costituire un robusto e vario training set;
- Un numero maggiore di combinazioni di hyperparameters da provare. Questo ovviamente avendo a disposizione una maggiore potenza computazionale.

Infine i modelli realizzati insieme a questo elaborato potranno essere tranquillamente utilizzati nella prossima primavera/estate e questo permetterà non solo una valutazione più accurata delle loro performance, ma anche un eventuale e più preciso tuning e ottimizzazione di essi.

Bibliografia

[1] **DJI Site:**
https://www.dji.com/it/p4-multispectral/specs

[2] **Seos-Project - Introduction to remote sensing**:
https://seos-project.eu/remotesensing/remotesensing-c01-p06.html

[3] **GISGeography - Multispectral vs Hyperspectral imagery**:
https://gisgeography.com/multispectral-vs-hyperspectral-imagery-explained/

[4] **Giannoni L, Lange F, Tachtsidis I. Hyperspectral imaging solutions for brain tissue metabolic and hemodynamic monitoring: past, current and future developments**:
https://pubmed.ncbi.nlm.nih.gov/29854375/

[6] **PhysicsOpenLab - Modern DIY Physics laboratory for science enthusiasts:**
https://physicsopenlab.org/2017/01/30/ndvi-index/

[7] **European Commision & DG AGRI**:
https://marswiki.jrc.ec.europa.eu/wikicap/index.php/CTS_Covid19

[9] **Agricolus Site**:
https://www.agricolus.com/indici-vegetazione-ndvi-ndmi-istruzioni-luso/

[10] **Yashwanth Medisetti - Neural Networks in Everyday life**:
https://medium.com/analytics-vidhya/neural-networks-in-everyday-life-ca2b7cb37052

[11] **Valentina Alto - Data augmentation in Deep Learning**:
https://medium.com/analytics-vidhya/data-augmentation-in-deep-learning-3d7a539f7a28

[12] **Tensorflow official website**:
https://www.tensorflow.org/

[13] **George Seif - Deep Learning vs Classical Machine Learning**:
https://towardsdatascience.com/deep-learning-vs-classical-machine-learning-9a42c6d48aa

[14] **Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow**
Aurélien Géron, Oreilly & Associates Inc; 2° edizione (11 ottobre 2019)

[15] **An Introduction to Statistical Learning: with Applications in R**
Gareth James, Daniela Witten,Trevor Hastie, Robert Tibshirani - Springer Verlag