UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Generazione e classificazione di time-series da immagini Sentinel-2 nell'ambito della Land Surface Phenology

Time-series generation and classification from Sentinel-2 images in the context of Land Surface Phenology

Relatore:
Dott. Mancini Adriano

Correlatore:
Dott. Pesaresi Simone

Laureando:
Deplano Lorenzo

Anno Accademico 2019-2020

Indice

Capitolo 1

Introduzione

Alcune specie vegetali hanno distinti cicli di vita determinati dalla caduta e dalla crescita del fogliame e da periodi di intensa attività di fotosintesi. Il continuo cambiamento del fogliame determina un cambiamento nella riflettanza elettromagnetica della superficie terrestre, che può essere misurata da sensori a distanza. Il cambiamento nel tempo di questa riflettanza viene chiamata **textitLand Surface Phenology (LSP)** [1]. Il presente lavoro di tesi si focalizza su algoritmi e script creati al fine di mappare le associazioni vegetali in determinate aree di studio. Le mappe fitosociologiche si sono dimostrate infatti uno strumento molto valido per il monitoraggio degli habitat naturali e seminaturali[2]. All'interno dell'elaborato si esporrà il processo per la creazione della mappa fitosociologica relativa a territori quali la Gola della Rossa e di Frasassi (codice: IT5320017) ed il monte Conero (codice: IT5320007). Il lavoro effettuato su queste due aree potrà essere esteso a diverse aree che necessitano lo stesso tipo di mappatura. Procedimenti di questo tipo possono aiutare a preservare e a monitorare (soltanto in Italia) 132 habitat, 90 specie di flora e 114 specie di fauna ai sensi della Direttiva 92/43/CEE "Habitat"[3]. Il progetto e la presente tesi sono stati svolti in collaborazione con i colleghi Colavito Cristian e Forconi Riccardo e al Dott. Pesaresi Simone del D3A.

1.1 Obiettivi del progetto

In Italia sono presenti 2278 ZSC (Zone Speciali di Conservazione) [4] e ogni 6 anni l'Unione Europea richiede il monitoraggio e la mappatura di quest'ultime ad ogni suo stato membro all'interno dei loro rispettivi territori [5]. La maggior parte delle volte il processo si rivela lungo e dispendioso, sia in termini di personale sia in termini economici. I rilievi e le osservazioni finalizzate alla mappatura sono infatti effettuate sul luogo, il che implica che il personale si debba recare direttamente nel sito della ricerca. Fino ad oggi infatti è stato necessario effettuare la mappatura delle associazioni vegetali riconoscendole direttamente in loco oppure attraverso il confronto manuale delle immagini [2]. Conseguentemente a tutto ciò nella maggior parte dei casi vengono consegnate delle mappe fitosociologiche poco aggiornate o che non lo sono affatto. Come se ciò non bastasse procedimenti di questo tipo non permettevano la quantificazio-

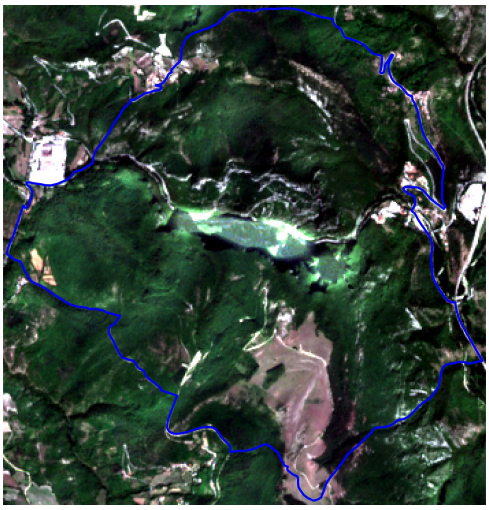

Figura 1.1 – Immagine della Gola di Frasassi rilevata da satellite

Figura 1.2 – Immagine del Monte Conero rilevata da satellite

ne dell'accuratezza della mappatura, il prodotto finale veniva quindi considerato come effettuato con il 100% di accuratezza. L'introduzione di strumenti quale il telerilevamento della zona da mappare permettono non solo la velocizzazione di un processo generalmente lento, ma anche l'introduzione di informazioni importanti come l'accuratezza della mappatura appena effettuata. L'obiettivo dello studio è quindi quello di processare immagini telerilevate per riuscire a ricavare diversi indici a terra. Attraverso gli indici ricavati si andrà a costruire un data-set che indica come essi evolvono all'interno del lasso di tempo di un anno, diviso in bi-settimane. Successivamente, dopo aver applicato una riduzione del numero di caratteristiche del data-set si andrà a produrre una mappatura delle diverse classi vegetali presenti nel territorio utilizzando dei modelli di predizione.

1.2 Il progetto

Utilizzando le immagini acquisite da satellite (in un periodo di tempo che va dall'Aprile 2017 al Marzo 2020 per entrambe le ZSC) si dovrà, in prima istanza, eseguire un pre-processing delle foto, mascherando le nuvolosità e coregistrandole. Successivamente a questa fase si avrà quindi a disposizione un data-set di immagini mascherate e cardinalmente coerenti tra loro i cui valori evidenziano chiaramente il cambiamento fenologico stagionale e annuale della vegetazione [2].

Prendendo il data-set di immagini coregistrate appena prodotto si vanno a ricavare da esse tutte le bande che necessitiamo per la creazione degli indici.

Avendo a disposizione un formulario per gli indici si andrà a combinare le bande tra loro. Successivamente all'applicazione delle varie formule si avranno a disposizione gli indici che verranno utilizzati per creare le serie temporali (*time-series*).

Lo scopo ultimo è quindi quello di confrontare serie temporali di diverse combinazioni di bande al fine di individuare quelle che producono un modello predittivo più preciso, o che minimizzano l'errore nella classificazione di certe specie.

Tabella 1.1 – Esempio di formula per la generazione degli indici

Formula numero	Forma algebrica	Tupla di input
4	$\frac{A-B}{C}$	(A,B,C)

Nella creazione delle serie temporali si andrà a produrre un file dove vengono tabulate le caratteristiche degli indici relative alle 52 settimane dell'anno per punti specifici all'interno della mappa. I punti scelti sono coloro in cui è stata effettuata la *ground truth* (verità a terra), cioè dove la classe (anche detta associazione) vegetale è stata verificata da un esperto in luogo.

In seguito alla creazione delle serie temporali si ha la possibilità di procedere in tre diverse modalità per la generazione dei modelli:

1. Lasciare il dataset invariato (utilizzando tutte e 52 le caratteristiche)
2. FPCA univariata: applicazione di uno script che genera degli FPCA scores utilizzando soltanto un indice
3. FPCA multivariata: applicazione di uno script che genera gli FPCA scores utilizzando una combinazione di indici

In questa tesi verrà discusso il caso in cui si andranno a costruire i modelli a partire dall'intero data-set (caso 1), basato sulle serie temporali complete (vedi 2.2.3). Una volta costruiti tutti i modelli su tutte le combinazioni di indici, verranno selezionati quelli che forniscono i migliori risultati in termini di precisione e errore di classificazione delle specie. Su questi ultimi verrà poi applicata la FPCA Multivariata, per osservare e quantificare un eventuale miglioramento della precisione della classificazione.

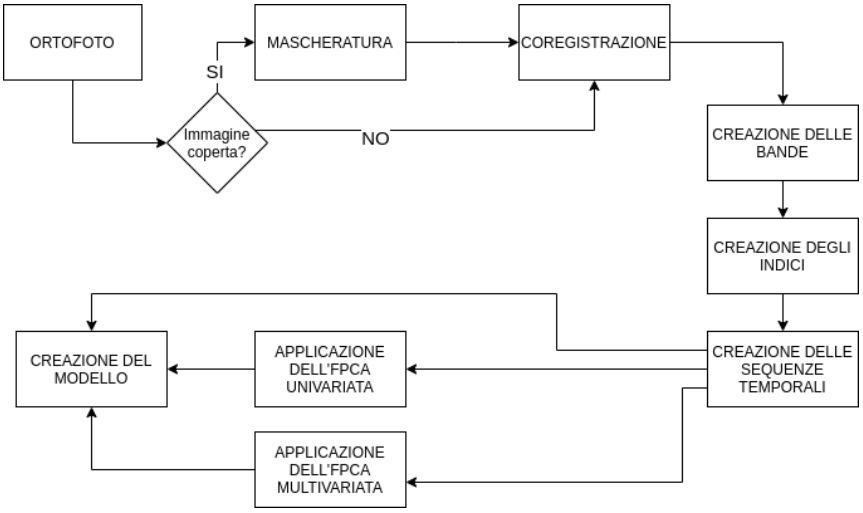

Figura 1.3 – Diagramma del progetto

1.3 Struttura della tesi

Nel secondo capitolo vengono illustrate tutti i metodi e le tecniche di cui ci si è serviti all'interno del progetto. Le tecnologie includono i software utilizzati e i linguaggi di programmazione, oltre che delle introduzioni preliminari ai concetti chiave del progetto.

Nel terzo capitolo viene illustrato il data-set utilizzato inizialmente e vengono illustrate le operazioni di:

- Mascheratura delle ortofoto del Sentinel-2 (laddove ciò fosse necessario).
- Coregistrazione delle foto mascherate.
- Estrazione delle bande utilizzate per la creazione degli indici.
- Calcolo degli indici e formule di partenza.
- Generazione delle time-series che evidenziano le caratteristiche degli indici relative a ogni settimana dell'anno.

Nel quarto capitolo si tratterà il processing del data-set attraverso l'elaborazione dei modelli di predizione utilizzando l'algorimo RandomForest.

Nel quinto capitolo si andranno a discutere i risultati relativi al progetto, in particolare quali formule ottimizzano la predizione delle associazioni vegetali e si discuterà in particolare quali di essi funzionano meglio in relazione alle specifiche classi vegetali. Sugli indici più performanti sarà applicata poi la *Multivariate Function Principal Component Analisys (MFPCA)*, andando a valutare e quantificare un ulteriore aumento della precisione delle predizioni.

Nel sesto capitolo si commenteranno le conclusioni derivanti dal progetto e verranno introdotti eventuali sviluppi futuri.

Capitolo 2

Metodi e tecniche utilizzate

In questo capitolo vengono trattati i metodi e le tecniche di cui ci si è serviti per sviluppare l'intero progetto.

2.1 Telerilevamento

Come detto nell'introduzione il telerilevamento è uno strumento fondamentale per velocizzare e semplificare la mappatura delle associazioni vegetali. I vantaggi derivanti dall'introduzione del telerilevamento sono:

- mappatura di zone irraggiungibili o difficilmente raggiungibili.
- inserimento di un numero maggiore di zone su cui effettuare lo studio.
- contribuisce a creare un iter specifico per la mappatura.
- velocizza la raccolta dei dati [2].

L'uso del telerilevamento, pur avendo grandi potenzialità per la raccolta dei dati, in particolare per lo studio in questione, non ha finora avuto un largo uso per quanto riguarda il monitoraggio in relazione al progetto Natura 2000 [2].

2.1.1 Sentinel-2

Le immagini utilizzate in questo studio derivano dai satelliti **Sentinel-2A** e **Sentinel-2B**, entrambi lanciati attraverso il lanciatore europeo Vega [6]. Entrambi i satelliti sono gestiti dalla European Space Agency (ESA) nell'ambito del programma Copernicus, programma dedicato al monitoraggio del pianeta e dell'ambiente a beneficio dei cittadini europei [7]. Le immagini elaborate all'intero del progetto sono chiamate **ortofoto**, cioè delle immagini geometricamente corrette che possono essere gestite come delle mappe [8]. I due satelliti sono sfasati di 180° tra loro e sono entrambi a bassa orbita. Il primo lancio del satellite Sentinel-2A è stato eseguito il 23 giugno 2015, mentre quello del Sentinel-2B è stato eseguito il 7 Marzo 2017 [9]. La durata di vita stimata dei due satelliti è di 7 anni ma al loro interno hanno abbastanza carburante per essere in circolo fino a 12 anni. I satelliti coprono una superficie di terreno che va dal 56° parallelo a sud all'84° parallelo a nord. Date queste configurazioni spaziali possiamo

dire che i satelliti hanno un intervallo di copertura di 5 giorni (2/3 giorni alle latitudini medie) [10].

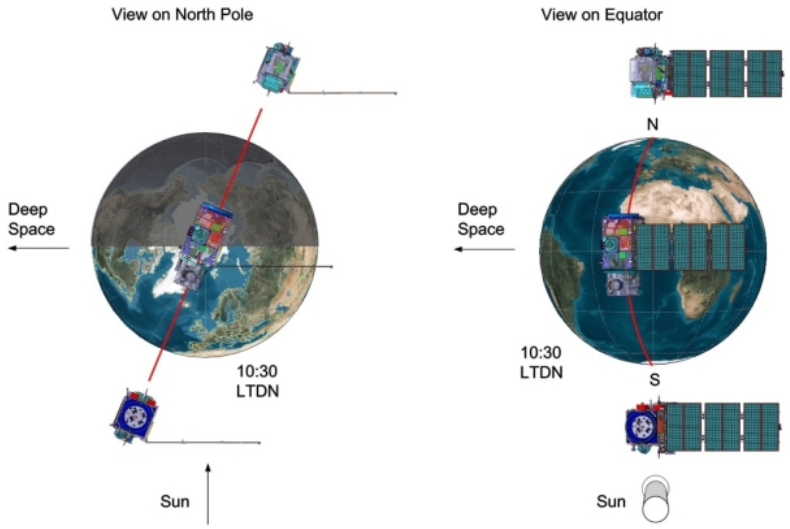

Figura 2.1 – Configurazione orbitale dei satelliti Sentinel-2 [9]

Attraverso la strumentazione ottica multi-spettrale al loro interno riescono a catturare immagini di 13 bande spettrali. Ogni banda ha una sua risoluzione, cioè ogni pixel dell'ortofoto corrisponde ad un'area in metri più o meno ampia. Quattro di queste dodici bande hanno una risoluzione di 10 metri (fig 2.2), altre sei hanno una risoluzione di 20 metri mentre le ultime tre hanno una risoluzione di 60 metri [9].

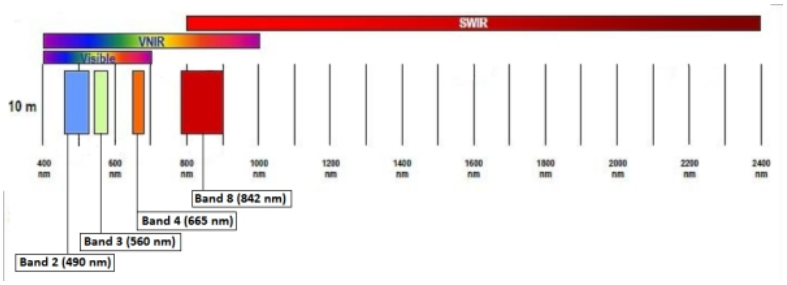

Figura 2.2 – Bande a 10 metri [11]

Le immagini provenienti dal satellite vengono calibrate per ottenere valori di riflettanza Bottom of Atmosphere (BoA) dai valori Top of Atmosphere (ToA). Difatti, essendo il satellite al di sopra dell'atmosfera terrestre, essendo puntato verso terra dovrebbe misurare la riflettanza dell'atmosfera stessa. Attraverso una correzione software riusciamo invece a misurare la riflettanza a terra, utile per effettuare le misurazioni di cui ci serviremo [12]. In figura 2.3 possiamo osservare le bande presenti nel satellite Sentinel-2:

Sentinel-2 bands	Central wavelength (µm)	Resolution (m)
Band 1 – Coastal aerosol	0.443	60
Band 2 – Blue	0.490	10
Band 3 – Green	0.560	10
Band 4 – Red	0.665	10
Band 5 – Vegetation red edge	0.705	20
Band 6 – Vegetation red edge	0.740	20
Band 7 – Vegetation red edge	0.783	20
Band 8 – NIR	0.842	10
Band 8A – Vegetation red edge	0.865	20
Band 9 – Water vapour	0.945	60
Band 10 – SWIR – Cirrius	1.375	60
Band 11 – SWIR	1.610	20
Band 12 – SWIR	2.190	20

Figura 2.3 – Bande Sentinel-2 [13]

2.2 Classificazione

2.2.1 Apprendimento supervisionato

L'apprendimento supervisionato è una tecnica di apprendimento automatico che permette di istruire un sistema informatico per poter effettuare predizioni sui dati[14]. Gli algoritmi di apprendimento supervisionato possono essere applicati in svariati campi e funzionano tutti nella stessa modalità. Un concetto fondamentale per l'addestramento di un algoritmo supervisionato è la ***ground truth***, perché ci permette di addestrare il **modello** con dei dati verificati attraverso osservazione diretta[15]. Prendendo esempio da questi il sistema riesce successivamente ad operare delle predizioni sulle altre istanze di cui non conosciamo la classe. Per la creazione del modello è quindi necessario fornire al sistema una serie di dati sulla quale esso effettua il *training*. Per *training* si intende un vero e proprio "allenamento" del sistema che, attraverso quella serie di dati chiamata *training set*, riesce a associare i dati e l'output da noi desiderato. Come è stato detto inizialmente infatti questo tipo di sistema è stato costruito per effettuare una predizione su una serie di dati che l'utente, per svariate ragioni (come ad esempio la numerosità dei campioni da classificare) non può verificare direttamente. Nell'apprendimento supervisionato si possono avere due tipi di situazioni:

- **Classificazione**: la variabile su cui deve essere effettuata una predizione è categorica (esempio fig. 2.4)

- **Regressione**: la variabile su cui deve essere effettuata una predizione è quantitativa (come ad esempio il prezzo di un oggetto) [16].

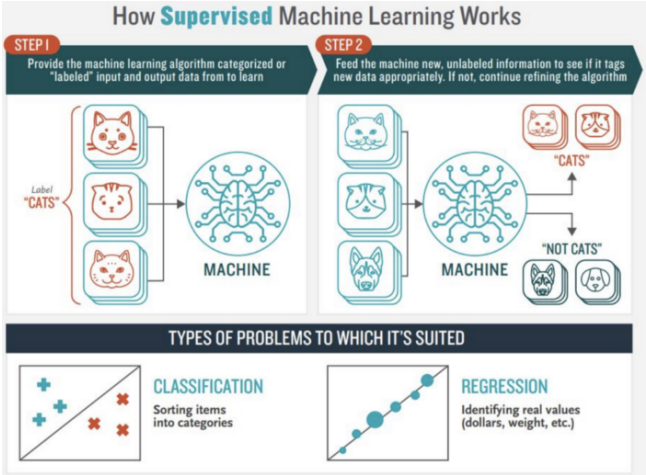

Figura 2.4 – Esempio apprendimento supervisionato [17]

Durante il progetto si andrà ad utilizzare soltanto un apprendimento supervisionato di classificazione, in particolare su 8 classi di associazioni vegetali (per dettagli sulle classi vedi capitolo 4.1).

2.2.1.1 Cross Validation

Normalmente per la creazione di un modello sarebbe necessario dividere il data-set in due parti: ***training-set*** e ***test-set***. Il primo, come detto, è necessario per l'addestramento del sistema, mentre attraverso il test-set andremmo a testare l'accuratezza del nostro algoritmo [17].

Attraverso la **cross validation** si avrà invece la possibilità di utilizzare l'intero data-set sia per l'allenamento del sistema sia per il test. Tutto ciò diventa possibile andando a dividere il nostro data-set in k cartelle, una di queste viene utilizzata per testare l'algoritmo mentre le altre $k-1$ vengono utilizzate per l'allenamento. Lo stesso procedimento viene effettuato iterativamente utilizzando come cartella di test una porzione sempre diversa dell'intero data-set [17]. Il vantaggio dell'utilizzo della cross validation è quindi quello di fornire un modello più accurato rispetto a quello che si avrebbe con la divisione in train/test, potendo utilizzare la totalità dei dati per il training.

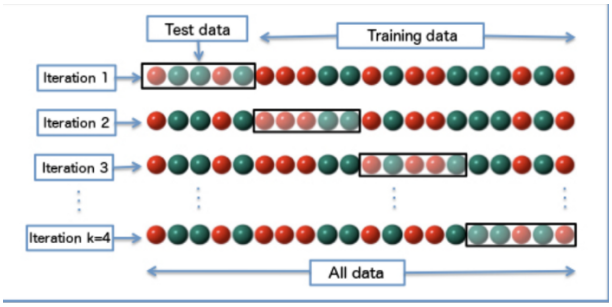

Figura 2.5 – Esempio di divisione del data-set [17]

2.2.2 Random Forest

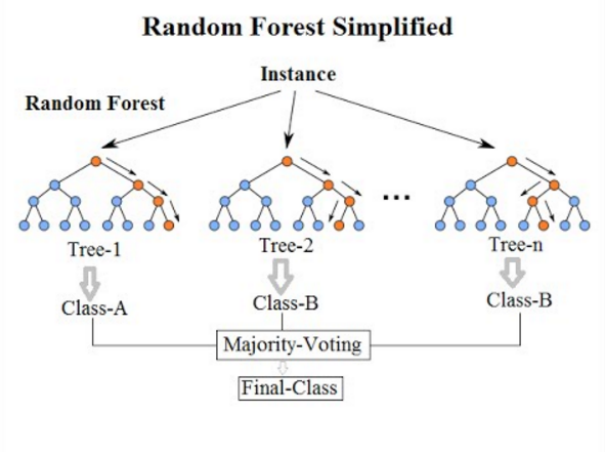

Figura 2.6 – Illustrazione semplificata del funzionamento del random forest [18]

Il ***random forest*** è un classificatore che consiste in un grande numero di alberi decisionali. Ognuno di essi prevederà una classe per ogni istanza presente all'interno del data-set.
Il concetto fondamentale dietro l'utilizzo del *random forest* è "l'unione fa la forza". Ogni albero decisionale prende una decisione che può essere giusta o errata. Ciò che rende il random forest un classificatore solido e affidabile è il fatto che la maggior parte degli alberi decisionali, a seconda delle caratteristiche dell'istanza, effettueranno una predizione corretta [19]. La decisione finale sulla classe dell'istanza si baserà infatti sulla risposta data dalla maggior parte degli alberi decisionali. Se infatti modello è ben costruito allora la maggioranza degli alberi decisionali effettuerà una predizione corretta.
Una caratteristica importante degli alberi decisionali che compongono il *random forest* è quella di avere una bassa correlazione tra loro. La bassa correlazione tra gli alberi decisionali fornisce predizioni molto diverse. Nonostante alcuni di esse siano errati, a seconda della caratteristiche, il classificatore riesce a comprendere la classe finale dell'istanza basandosi su ciò che la maggior parte degli alberi ha predetto. Nonostante questo possa sembrare un punto debole esso non lo è affatto, infatti permette all'algoritmo di esplorare tutte le diverse possibilità date dalle caratteristiche. È inoltre importante considerare che i risultati degli alberi decisionali potrebbero essere differenti a seconda del data-set sulla quale viene effettuato il *training*.
Ogni albero decisionale ha infatti a disposizione un data-set differente (composto sempre dallo stesso numero di elementi del data-set originale) ed effettua il *training* su quest'ultimo. I data-set, pur avendo lo stesso numero di elementi, sono differenti tra loro dato che alcune istanze saranno ripetute.
Ad esempio se il data-set originale è (1,2,3,4,5,6) il data-set per il training di un albero decisionale potrebbe essere (1,2,2,3,6,6) [19]. Questo concetto che permette di differenziare le predizioni degli alberi decisionali che compongono il *random forest* viene chiamato ***bagging***.

Un'altro concetto su cui si basa la diversificazione degli alberi decisionali del *random forest* è il ***feature randomness***, che consiste nell'effettuare la predizione soltanto su una parte delle caratteristiche.
Solitamente, all'interno degli alberi decisionali, viene scelta la caratteristica che divide maggiormente gli elementi che si trovano nella diramazione destra rispetto alla diramazione sinistra dell'albero [19], cosa che invece non viene effettuata negli alberi decisionali del *random forest*. Essi effettuano la decisione prendendo sempre caratteristiche diverse, in modo da differenziare nettamente gli alberi decisionali (e di conseguenza i risultati) dell'algoritmo.

2.2.3 Time-Series (Serie Temporali)

Le **serie temporali** si definiscono come un'insieme di misurazioni, solitamente effettuate con cadenza temporale fissa, che esprimono il comportamento di un certo fenomeno rispetto al tempo.[20] Con l'analisi delle serie temporali si possono individuare nel fenomeno componenti di ciclicità o stagionalità, sia predirne l'andamento futuro.
In questo progetto le serie temporali sono calcolate su singoli punti dello spazio estraendo le proprietà di quel punto su ciascun'immagine, corrispondente ad un certo giorno dell'anno. L'intervallo temporale di osservazione scelto in questo progetto è quello **settimanale**, ogni serie temporale avrà quindi 52 valori, calcolati facendo **la media** delle misurazioni dei giorni appartenenti alla stessa settimana.
Per via della presenza intrinseca della stagionalità nell'ambito di studio di questo progetto, si cerca di ricondurre le serie temporali elaborate ad un comportamento circolare, dato che la settimana 52 e la settimana 1 dell'anno di fatto coincidono in termini di caratteristiche della vegetazione in quanto appartenenti alle medesime condizioni stagionali. Questo fenomeno nell'analisi delle time-series è anche conosciuto in gergo come ***Circular Spline***. Ciò è approfondito nel capitolo 3.5. Si può notare chiaramente un comportamento di questo tipo nel grafico sottostante questa sezione.
I dati estrapolati dalle immagini, come precedentemente evidenziato, non sono ricavati direttamente da esse ma sono frutto di elaborazioni preliminari di combinazioni di diverse bande (indici), calcolati secondo diverse funzioni. Saranno prodotte quindi tante serie temporali quanti sono gli indici combinati.
Il **data-set**, ovvero l'insieme complessivo di dati su cui verrà poi effettuata la classificazione, è costituito dall'insieme delle serie temporali di ogni punto in cui è stata rilevata la *ground truth*.

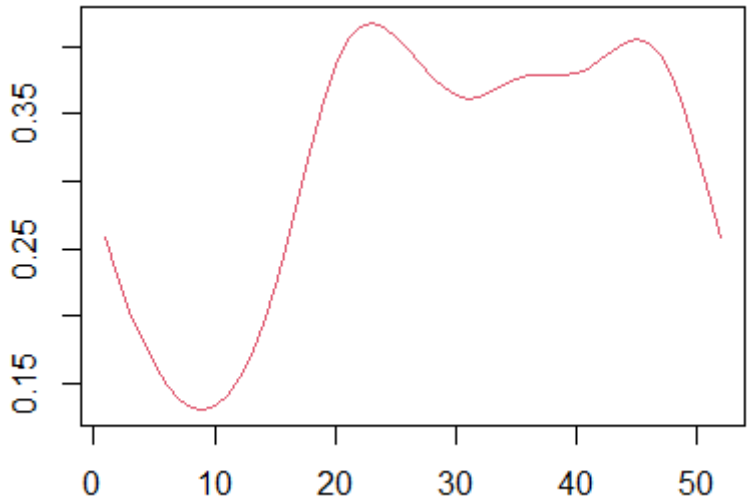

Figura 2.7 – esempio di time-series di un punto nello spazio

2.3 Strumenti software

2.3.1 QGIS

QGIS (Quantum GIS) è una applicazione che permette di visualizzare, organizzare, analizzare e rappresentare dati spaziali [21]. Il *software* è multi-piattaforma e può essere utilizzato in sistemi operativi come Windows e Linux. I dati possono essere organizzati in *layers* che possono essere sovrapposti allo scopo di creare una visione della scena più chiara e comprensibile.

Figura 2.8 – logo QGIS [22]

In particolare QGIS supporta formati di file come shapefile [23], e proprio questo sarà il suo utilizzo all'interno di questa tesi. Attraverso gli algoritmi costruiti alcune zone delle immagini prelevate da satellite sono sono difficili da escludere, ma proprio attraverso l'utilizzo di QGIS abbiamo la possibilità di effettuare il processo di eliminazione di queste parti a mano. Per il progetto si è scelto di utilizzare QGIS dato che esso supporta diversi tipologie di file (vettoriali o raster) come:

- Geopackage
- AutoCAD DXF

- JPEG/Jpeg 2000
- ESRI Shapefile [23].

2.3.2 R

Figura 2.9 – logo R [24]

R è un software open-source che permette la computazione statistica. Esso è un software multipiattaforma che funziona in sistemi operativi Linux-based, Windows e MacOS. È un linguaggio simile a S e può essere considerato come una sua implementazione [24]. R raccoglie una grande varietà di strumenti per computazione statistica e grafica come la modellazione, lineare e non, analisi delle *time-series* e la classificazione [24].

All'interno dell'applicativo abbiamo integrati diversi strumenti che includono:

- strumenti per la manipolazione efficiente dei dati e salvataggio dei dati;
- strumenti per la manipolazione e il calcolo su vettori e matrici;
- Permette l'integrazione di codice C,C++ e Fortran per compiti con alta computazione [24].

All'interno del progetto gli script in R sono stati scritti all'interno di Rstudio. Rstudio è un IDE scritto in Java, JavaScript e C++ nel 2010 ed è un software multipiattaforma [25].

2.3.2.1 Librerie R

Di seguito vengono riportate le principali librerie utilizzate all'interno di R. In particolare le librerie più ricorrenti e più utilizzate sono:

- **RStoolbox**: Strumenti per il processamento di immagini telerilevate e analisi come il calcolo degli indici spettrali e classificazione supervisionata e non supervisionata [26].
- **caret** (*Classification And Regression Training*): pacchetto creato da Max Kuhn nel 2019 è un set di funzioni creato con l'obiettivo di razionalizzare il processi per la creazione dei modelli predittivi. Il pacchetto contiene strumenti per il *data splitting* e il *pre-processing* dei dati [27].
- **randomForest**: pacchetto per la classificazione e regressione basata su una serie di alberi e utilizzando input random.

2.3.3 JupyterLab

JupyterLab è un'interfaccia utenti web-based rilasciata nel Febbraio 2018 che permette la creazione di documenti e attività come i *notebook Jupyter*, editor di testo e terminali [28]. La creazione di *notebook* permette di eseguire un codice all'interno di un elaboratore remoto utilizzando una serie di celle che possono essere eseguite separatamente. Per andare ad eseguire un *notebook* è necessario affidargli un dato *kernel*, attraverso la quale esso eseguira tutte le istruzioni all'interno del file (con estensione .ipynb). JupyterLab permette anche un modello unificato per la visualizzazione e la manipolazione di diversi tipi di formati di dati come ad esempio CSV. In particolare JupyterLab è stato utilizzato per lavorare all'interno di elaboratori remoti forniti dall'Università Politecnica della Marche. L'esecuzione degli script in Jupyter ha permesso quindi una maggiore velocità di produzione dei dati necessari al progetto permettendoci di lavorare in modo maggiormente parallelo.

Figura 2.10 – logo Jupyter [29]

2.3.3.1 PuTTy

PuTTY è un *client SSH* e *Telnet* combinato con un emulatore di terminale per gestire in remoto sistemi informatici. PuTTy è un software open-source per i sistemi Windows e Unix, in questo progetto è stato utilizzato per accedere a JupyterLab via protocollo SSH.[30]

Figura 2.11 – logo PuTTy [31]

SSH sta per *Secure Shell*, questo protocollo permette di creare una sessione remota sicura tra client e server. Per proteggere la privacy e aggirare le regole

di routing del firewall, è stato fatto un SSH tunnel, in modo da accedere alla rete dell'università politecnica delle marche.

Capitolo 3

Creazione del data-set

All'interno di questo capitolo verrà introdotto il *processing* iniziale effettuato sulle immagini Sentinel-1 che ricadevano nelle due aree di studio (descritte più in dettaglio nel capitolo 1). Queste verranno prima mascherate da eventuale nuvolosità, coregistrate, le diverse bande di colore contenute in esse verranno interpolate tra loro secondo diverse funzioni e successivamente tabulate, creando così il data-set su cui opereremo la classificazione.

3.1 Mascheratura

Le immagini del satellite racchiudono un'area molto ampia, eccessivamente grande rispetto all'area di studio che è d'interesse in questo progetto. Andranno quindi inizialmente **ritagliate** (Crop) secondo le dimensioni dell'area di studio. Non tutte le immagini sono prive di imperfezioni come nel caso della figura 1.21.2, in alcune di esse vi sono aree coperte da nuvole dove non è possibile ricavare alcuna informazione riguardo alla vegetazione sottostante. Lo stesso accade con le ombre delle nuvole proiettate sul suolo dalla luce solare, che vanno a distorcere le componenti di colore in quelle zone dell'immagine. É quindi necessario **mascherare** queste aree sostituendo al valore di colore dei pixel corrispondenti il valore *NO DATA*. Alcune immagini considerate come troppo nuvolose, con una percentuale eccessiva della superficie dell'area di studio coperta, sono state totalmente scartate anziché essere mascherate.

3.1.1 Ritaglio (Crop) dell'immagine

Le informazioni riguardanti la delimitazione dell'area di studio e altre proprietà di quest'ultima sono contenute in uno **shapefile**. Il crop delle immagini viene effettuato sovrapponendole con tale shapefile, utilizzando la funzione crop del package `Raster`[32] di R.

```
immagine <- crop(immagine, shapefile)
```

Figura 3.1 – esempio di immagine satellitare estesa non ritagliata

3.1.2 Applicazione della maschera

Le parti dell'immagine contenenti nuvole o ombre vengono riconosciute e filtrate in base al valore del **NTDCI** (Normalized Difference Thermal Cloud Index), un particolare NDCI (Normalized Difference Cloud Index)[33], calcolato dalla funzione `cloudMask` del package `RStoolBox` [26] di R.

```
supportoMaschera <- cloudMask(immagine, blue = 2, tir = 11, buffer = 5)
```

Questa funzione restituirà un raster in cui ogni pixel ha il valore di NTDCI, calcolato sulle bande blue e SWIR (Short Wave Infra-Red), nel caso di Sentinel-2 le bande sono la 2 e la 11. Successivamente viene generata la maschera filtrando valori di NDTCI sotto una certa **soglia**. Tale soglia è determinata caso per caso analizzando i valori che ciascun'immagine produce. Solitamente si ottengono buoni risultati fissandola tra -0.1 e +1.15 per le nuvole e tra 0.8 e 0.92 per le ombre delle nuvole. Vengono mantenuti valori minori della soglia nel caso delle nuvole, mentre vengono mantenuti valori maggiori della soglia nel caso delle ombre, gli altri vengono scartati. In questo modo si è generata una **maschera**, cioè un'immagine che andrà sovrapposta a quella di partenza e confrontata pixel per pixel, inserendo il *no-data* sull'immagine di partenza nelle zone che sono state riconosciute come nuvole. Per farlo viene usata la funzione `mask` del package `Raster`[32] di R.

```
scena.mascherata <- mask(immagine, maschera, inverse = TRUE)
```

Figura 3.2 – Esempio di immagine con nuvole da mascherare

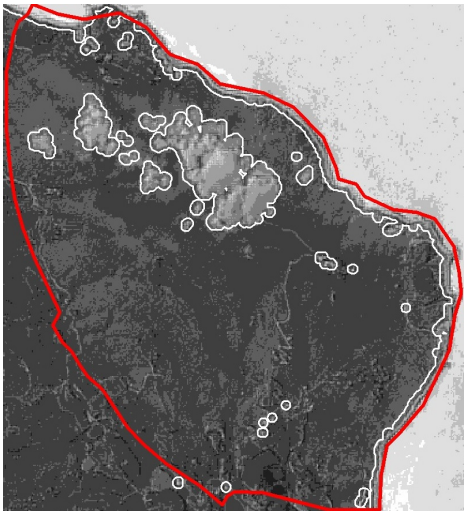

Figura 3.3 – Esempio di maschera con NTDCI filtrato con soglia a 0.2

3.1.2.1 Mascheratura manuale con QGIS

Laddove la maschera generata tramite la soglia con le modalità precedentemente descritte, risulti poco accurata e vada a mascherare eccessivamente o non sufficientemente le zone nuvolose, è possibile effettuare l'operazione manualmente servendosi del sw QGIS.
Occorre sovrapporre lo shapefile e l'immagine da mascherare, con un'opacità inferiore al 100% in modo che, sotto lo shapefile, l'immagine sia riconoscibile.
Successivamente, usando la funzione "Crea buco" di QGIS, si rimuove l'area dello shapefile in corrispondenza delle nuvole da mascherare, esportando il risultato in uno shapefile differente.

Figura 3.4 – Esempio di immagine con nuvola da mascherare manualmente

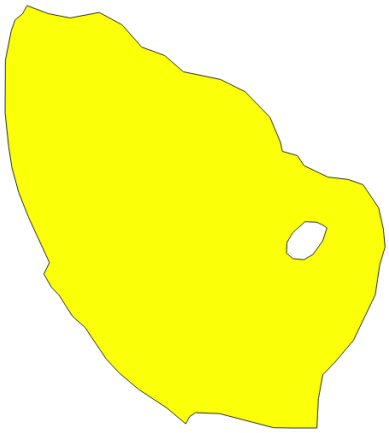

Figura 3.5 – Esempio di shapefile "bucato"

A questo punto si procede con la mascheratura tramite la funzione `mask`, similmente al caso precedente, sovrapponendo però all'immagine non più la maschera generata in automatico, ma lo shapefile "bucato", realizzato manualmente.

```
scena.mascherata <- mask(immagine, shapefile_bucato)
```

Figura 3.6 – Risultato della mascheratura manuale

3.2 Coregistrazione

Per le successive operazioni di estrazione dei dati dalle immagini, che avverranno prelevando i dati sui punti in cui si conosce la *ground truth*, occorre che esse siano **allineate** tra loro in modo che le coordinate di ciascun pixel di ciascuna foto corrispondano alla stessa area geografica. Immagini così allineate si dicono **coregistrate**. Al fine di coregistrarle, si seleziona un'immagine, detta *master*, tra tutte quelle a disposizione, e si allineano ad essa le altre, qui chiamate *target*.

Prima si preleva l'estensione dell'immagine master tramite la funzione `extent`, tale estensione verrà ampliata di 40 metri in ogni direzione, con la funzione `extend`, entrambe del package `Raster`[32] di R:

```
estensione <- extent(master)
contornoEsteso <- extend(est,40,40,40,40)
```

Questo occorrerà successivamente per ritagliare l'immagine che verrà coregistrata. Si estendono inoltre anche l'immagine master e ciascuna immagine target di 10 pixel in ogni direzione, in quanto eventuali spostamenti durante l'allineamento potrebbero portare alla perdita di informazioni se non si aumentasse l'estensione.

```
master <- extend(master, c(10,10))
...
...
target <- extend(target, c(10,10))
```

Infine si procede alla coregistrazione tramite la funzione `coregisterImages` del package `Raster`[32] di R, la quale confronta un certo numero di *samples* (campioni) di entrambe le foto, **traslando** il *target* di conseguenza. In questo caso il numero di *samples* è impostato a 20000:

```
coregistered <- coregisterImages(target, master = master, nsamples = 20000)
```

Infine l'immagine target viene ritagliata con la funzione `crop` secondo il contorno esteso creato precedentemente.

```
target.coreg <- crop(coregistered$coregImg,contornoEsteso)
```

3.3 Estrazione delle bande

Il seguente script è stato usato per andare a selezionare tutte le bande utili per la sezione seguente (la creazione degli indici). Con il seguente script si andranno a generare 11 bande, di cui due poco significative, in particolare la banda 1(*coastal aerosol*) e 9(*water vapor*) non sono state utilizzate nelle sezioni seguenti in quanto più significative in ambiti acquatici. Lo script andrà a chiedere in input:

- tutte le immagini coregistrate nella sezione precedente andando a chiedere il path, mentre con la funzione `stack()` si otterrà un oggetto contenente tutte le informazioni che necessitiamo.
- lo shapefile per delimitare la zona di interesse.

I seguenti passi verranno ripetuti per ogni singola banda. Verranno selezionati tutti i file dove verrà eseguita l'estrazione dei *layer*

```
ind.blue = grep("0.2$", layer)
```

Con il comando `grep()` si andranno a selezionare gli indici dei livelli che terminano con 0.2, in particolare il livello 0.2 è quello relativo alla banda blu.

```
BLUE <- subset(mylayers,ind.blue)
BLUE <- crop(BLUE, shapefile)
BLUE <- mask(BLUE, shapefile)
```

Si andrà quindi ad assegnare a `BLUE` solo i dati utili. Successivamente andremo poi a selezionare attraverso lo shapefile l'area di interesse sulla quale è stata selezionata solo la banda richiesta.

Si andrà successivamente a creare la sottocartella relativa alla banda che si sta elaborando attraverso il comando `dir.create()`.
Infine la funzione `writeRaster()` creerà la banda relativa a quella immagine e

lo farà per tutte le immagini caricate inizialmente.
Un esempio della banda 8(NIR) lo possiamo trovare in figura 3.7

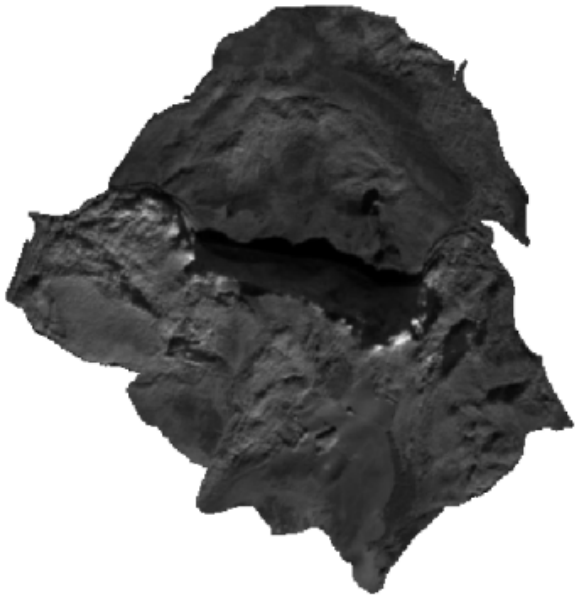

Figura 3.7 – Esempio di selezione della banda NIR

3.4 Creazione degli indici

Attraverso l'utilizzo delle bande create nella sezione precedente si passa alla creazione degli **indici**. La loro creazione è fondamentale per la ricerca di un mix di bande che ottimizzi la identificazione delle associazioni vegetali nei luoghi di studio. In particolare vengono utilizzate delle formule prestabilite per legare le diverse bande delle immagini tra loro.

Tabella 3.1 – Tabella delle formule degli indici computati

Indici	**Formula**	**Operandi**	**Regola 1**	**Regola 2**
1	$A - B$	2	A > B	-
2	$\frac{A}{B}$	2	A > B	-
3	$\frac{A}{B}$	2	A > B	-

Indici	Formula	Operandi	Regola 1	Regola 2
4	$\frac{(A-B)}{C}$	3	A > B	C > B
5	$\frac{(A-B)}{(C+B)}$	3	A > B	C > B
6	$\frac{(A-B)}{(C-B)}$	3	A > B	C > B
7	$\frac{\frac{(A-B)}{(A+B)}}{\frac{(A-C)}{(A+C)}^{-1}}$	3	A > B	A > C
8	$\frac{\frac{(A-B)}{(A+B)}}{\frac{(D-C)}{(D+C)}^{-1}}$	4	A > B	D > C
9	$\frac{A}{B} \cdot \frac{(C-D)}{(C+D)}$	4	A > B	C > D
10	$\frac{A}{B} \cdot \frac{(A-C)}{(A+C)}$	3	A > C	-
11	$\frac{A}{B} \cdot \frac{(B-C)}{(B+C)}$	3	B > C	-
13	$\frac{A}{B} \cdot \frac{C}{D}$	4	-	-
15	$\frac{(A-B)}{((A+B+C)+1)}$	3	A > B	-

Indici	Formula	Operandi	Regola 1	Regola 2
16	$\frac{((A-C)-(B-C))}{((A-C)+(A+C))}$	3	A > C	B > C
17	$\frac{((A-C)-(B-D))}{((A-C)+(B-D))}$	4	A > C	B > D
19	$\frac{(2A-B-C)}{(2A+B+C)}$	3	-	-
20	$\frac{(A-B)}{(A+B+C)}$	3	A > B	-
21	$\frac{(A-(B+C))}{(A+(B+C)}$	3	A > B	A > C
22	$log(\frac{A}{B})$	2	-	-
23	$(A-B)\cdot C$	3	A > B	-
25	$\sqrt{\frac{(A-B)}{(A+B)}}$	2	A > B	-

Il seguente codice si riferisce alla costruzione degli indici per quanto riguarda la funzione 01. Per creare gli indici delle altre funzioni è stato seguito lo stesso schema di ragionamento, andando a cambiare soltanto alcuni dettagli relativi alla generazione delle combinazioni delle bande.

```
for (A in bands)
{
  for(B in bands)
```

```
  {
    if(B >= A){next}
    combinazioni <- rbind(combinazioni,c(A,B))
  }
}
```

Nella prima parte dello script si vanno a costruire le combinazioni delle bande della funzione in questione. In particolare, come possiamo osservare nella tabella 3.1, è necessario seguire una serie di regole per quanto riguarda la generazione degli indici di ogni funzione. Se ciò non fosse fatto si rischierebbe di rendere gli indici negativi e avere delle ripetizioni nel data-set che andremmo a creare. Le regole seguite per la creazione di ogni indice delle funzioni si trovano nelle colonne "regola 1" e "regola 2" della tabella 3.1.

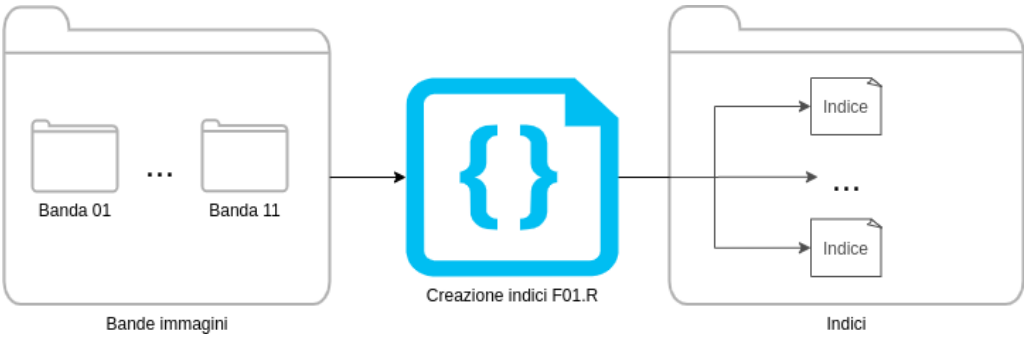

Figura 3.8 – Diagramma dell'esecuzione dello script per la creazione degli indici

```
for(i in c(1:nrow(combinazioni)))
```

all'interno di un ciclo come quello appena presentato possiamo andare ad automatizzare la creazione degli indici per ogni funzione, senza dover operare l'intero processo a mano. All'interno di `combinazioni` avremo infatti un contenuto simile a:

```
     [,1] [,2]
[1,] "11" "10"
[2,] "11" "08"
[3,] "11" "07"
[4,] "11" "06"
[5,] "11" "05"
.................
```

dove la prima colonna si riferisce alla banda A nella tabella 3.1 e la seconda alla banda B. In seguito, per andare a reperire le diverse bande, andiamo ad operare una concatenazione per inserirci nella cartella interessata.

```
path1 <- paste(path,combinazioni[i,1],sep='')
percorsi.A <- list.files(path=path1, pattern="*.tif",full.names = TRUE)
```

In `path` è contenuto il percorso della cartella dove sono presenti tutte le bande. Andando a concatenare subito dopo il numero della banda interessata possiamo inserirci direttamente all'interno della cartella con le immagini di interesse. Attraverso il comando `list.files()` andiamo poi a reperire tutte le immagini con estensione `.tif` che si trovano nella cartella. In seguito alla creazione di uno stack per ogni banda coinvolta nella creazione dell'indice attraverso la funzione `A <- stack(percorsi.A)` andiamo ad assegnare il nome al file con estensione `.tif` che verrà generato. Il nome del file sarà del tipo

```
S2_F01_bandaA_bandaB_00_00_date_20180620.tif
```

dove al posto di `bandaA` e `bandaB` verranno inseriti i numeri delle bande che compongono l'indice e dopo `date` verrà inserita la data relativa alla foto in questione. Nella sezione degli indici si va a creare una funzione che ci permette di effettuare il mix delle nostre bande. Ad esempio, per la funzione 1:

```
one <- function(x,y) x-y
```

Successivamente, attraverso l'utilizzo della funzione `overlay` andiamo a sovrapporre le due bande e a creare l'indice. Dopo aver creato le cartelle dove inserire tutti gli indici andiamo a salvare le immagini utilizzando `writeRaster()`, specificando che il formato del file dev'essere `GTiff`. In seguito ad aver effettuato tale processo per tutte le funzioni presenti abbiamo a disposizione un data-set contenente tutti gli indici dalla quale andremo a ricavare le **time-series**.

3.5 Creazione delle time series

In questa fase le immagini vengono convertite in dati numerici, in modo da essere poi processate dagli algoritmi di classificazione. Questi dati numerici saranno estrapolati da ciascuna immagine secondo uno *shapefile* puntuale, che contiene, oltre alle coordinate geografiche e spaziali dei punti, anche le informazioni sulla classe di appartenenza di tali punti, risultato dei rilevamenti della verità a terra2.2.1. I dati verranno inoltre suddivisi in settimane, facendo una media tra le informazioni contenute in foto che appartengono alla stessa settimana.

Inizialmente vengono caricate le immagini corrispondenti ad un dato indice e inserite in uno stack. Dei singoli punti dello *shapefile* viene prima fatto un *buffer* con la funzione buffer di R, il singolo punto diviene così una superficie più estesa (in questo caso un cerchio di raggio 30 pixel), in modo da avere delle aree più ampie da cui estrapolare i valori delle immagini. Per estrarre tali dati viene usata prima la funzione `mask` che ritaglia le immagini in corrispondenza dei punti dello *shapefile* (figura 3.10), per poi successivamente applicare la funzione `rasterToPoints`, convertendone il risultato in un data-frame in modo da averne una tabulazione. Tutte le funzioni citate fanno parte del package Raster[32] di R.

```
shapefile_buffer <- buffer(shapefile, 30)#buffer di 30 pixel
...
...
```

```
masked_rasters <- mask(rasters, shapefile.buffer)
...
pointsData <- as.data.frame(rasterToPoints(maskedRasters))
```

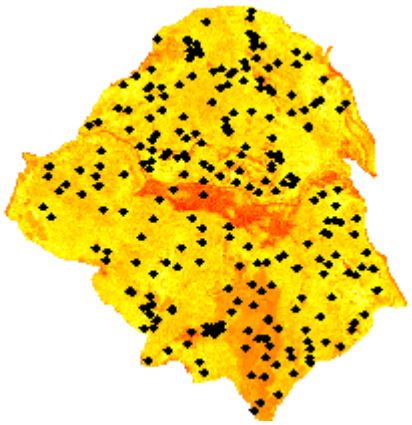

Figura 3.9 – sovrapposizione tra raster di partenza e punti dello shapefile puntuale

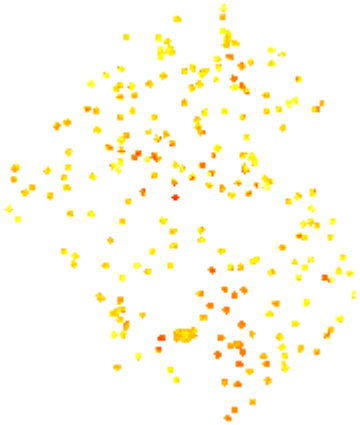

Figura 3.10 – esempio di raster mascherato dopo il buffer sui singoli punti

A questo punto i dati generati vengono ordinati cronologicamente in base ai giorni dell'anno. Vengono poi aggregati, per ciascun punto cioè si ottengono 52 valori, dove ognuno è la media dei valori di una stessa settimana. Per farlo si usa la funzione `aggregate` del package Raster[32].

```
#aggregazione per settimana, secondo la media (mean),
i no-data vengono scartati (na.rm = true)

dati_aggregati <- aggregate(pointsData, by = pointsData[,"WEEK"],
                            FUN = mean, na.rm = TRUE)
```

Per escludere valori anomali che porterebbero a margini di errore più elevati nelle fasi successive, i valori vengono resi più omogenei applicando un modello

GAM[34] (figura 3.11). Avere valori più omogenei, che approssimano meglio cioè il reale andamento delle caratteristiche delle funzioni che si vogliono rappresentare, permette di migliorare l'aspetto già citato nel capitolo 2.2.3 della *Circular Spline*[35]

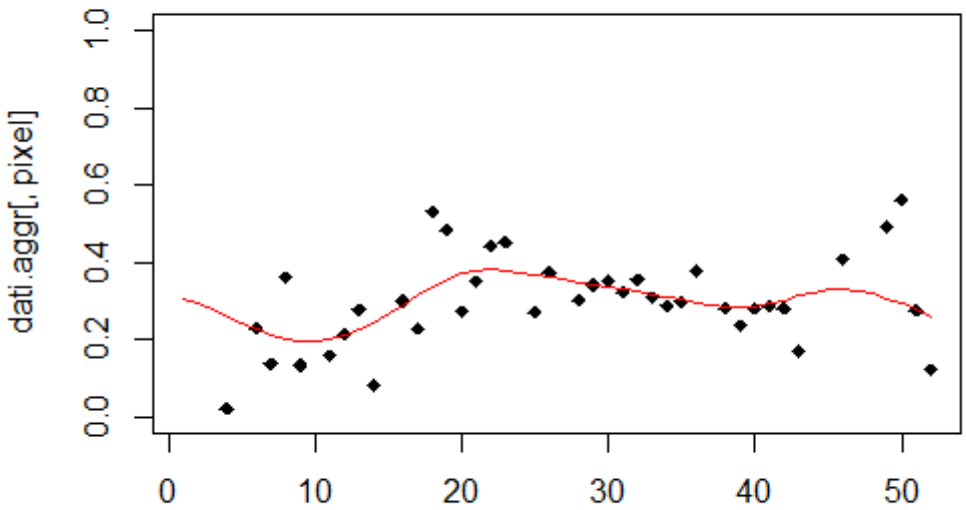

Figura 3.11 – esempio di "lisciatura" dei dati tramite modello GAM

Infine, i valori dei punti sino a qui elaborati, che inizialmente erano stati estratti spazialmente da aree più ampie di singoli punti in quanto è stato esteso lo shapefile puntuale tramite buffer, sono reinseriti in un raster e successivamente riestratti, questa volta però come singoli punti. Perchè ciò avvenga è necessario interpolare i valori dei pixel del buffer attorno al punto. In questo progetto si è scelto di utilizzare l'interpolazione bilineare[36]. Si usa la funzione textttextract del package Raster[32]

```
dati_finali <- extract(raster,shapefile,  method = 'bilinear')
```

A questo punto il la time-series ha preso forma, vi sono 52 colonne di dati, corrispondenti alle settimane dell'anno, e 242 righe, corrispondenti a ciascun punto dello shapefile di classificazione.

L'ultimo passo consiste nell'associare ad ogni punto (riga del dataset) la corrispondente classe di appartenenza, e salvare il file.
In questo progetto i file sono stati salvati in formato .txt, indicando nel nome

la funzione e le bande corrispondenti all'indice, la risoluzione spaziale (in metri) corrispondente a ciascun pixel, e una sigla per indicare il tipo d'interpolazione. Il file è generato dalla funzione `WriteTable` di R.

```
write.table(dati_finali, 'FXX_B1_B2_B3_B4_10_BIL.txt')
```

Il data-set avrà un formato simile alla tabella 3.2

Tabella 3.2 – Esempio di riga del data-set generato dalle time-series

	X1	X2	...	X52	CLASS
V1	0.669484446716053	0.667264714603539	...	0.669484446716053	Q
V2	0.693001943028295	0.687077625639308	...	0.693001943028295	O

Capitolo 4

Processing del data-set

4.1 Creazione dei modelli

Il codice listato in questa sezione si riferisce alla creazione di un modello di predizione della classe vegetale. Esso è stato creato per ogni indice composto nelle sezioni precedenti. Nonostante il numero di modelli che si andrà a creare sia molto elevato tutto ciò è stato necessario in quanto, a seconda dei risultati relativi a ogni modello, possiamo ricavare quale degli indici produce risultati migliori. Tali indicazioni sono fondamentali per aumentare l'accuratezza della mappatura prodotta ed, eventualmente, andare a comporre gli indici tra loro attraverso l'utilizzo della FPCA multivariata. L'utilizzo della FPCA multivariata potrebbe permettere di aumentare ulteriormente l'accuratezza della predizione delle classi vegetali.

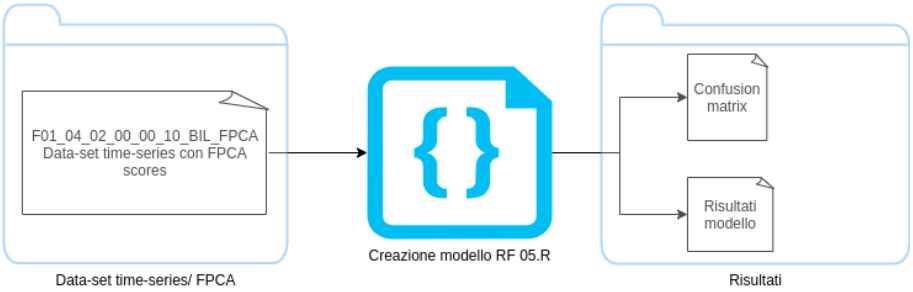

Figura 4.1 – Esempio di creazione del modello e salvataggio dei risultati

In prima istanza si vanno a creare le combinazioni in modo da lavorare soltanto sugli indici effettivamente presenti (codice identico a quello nel capitolo 3.4). Dopodiché si vanno a reperire le *time-series* all'interno di una cartella di riferimento. Dopo aver inserito il contenuto del file di testo all'interno del data frame `dataset`, attraverso il comando `read.delim()` andremo a trasformare i dati contenuti all'interno del data-set in `factor` attraverso:

```
dataset[,ncol(dataset)] <- as.factor(dataset[,ncol(dataset_FPCA)])
```

Questo passaggio è fondamentale in quanto l'algoritmo di classificazione elabora i dati in input soltanto se il loro formato è di tipo `factor`.

```
ctrl <- trainControl(method = "repeatedcv",number = 3, repeats = 10)
```

attraverso questo comando andiamo a configurare i parametri per il *training* del modello. In questo caso abbiamo utilizzato **`repeatedcv`** come metodo di addestramento. Il metodo è molto simile alla *cross-validation*, l'unica differenza è che attraverso la *repeated cross-validation* abbiamo la possibilità di ripetere questo processo un certo numero di volte per aumentare l'accuratezza delle caratteristiche del modello. I parametri passati alla funzione `trainControl` sono:

- `method` che si riferisce al metodo utilizzato per effettuare il *training* (in questo caso *repeated cross-validation*, ma avremmo potuto scrivere *cv* per utilizzare la *cross-validation*)
- `number` numero di cartelle su cui effettuare la *cross-validation*.
- `repeats` numero di volte che la *cross-validation* deve essere ripetuta.

```
tuneGrid2 <- data.frame(
  mtry = c(1:sqrt(ncol(dataset_FPCA)))
)
```

`tuneGrid2` rappresenta la ***tuning grid*** utilizzata per creare il modello. Andando a creare una *tuning grid* personalizzata, è possibile andare ad esplorare diversi **mtry**, cioè diversi **hyperparametri**, del random forest. Gli hyperparametri sono dei parametri inseriti all'interno dell'algoritmo di classificazione che governano tutto il processo di addestramento dello stesso [37]. In questo caso l'hyperparametro del random forest è l'**mtry**; esso rappresenta il numero di variabili scelto in modo random tra quelle presenti ogni volta che si effettua lo split negli alberi decisionali [38]. A seconda di come l'mtry è stato scelto il modello avrà un'accuratezza più o meno alta.

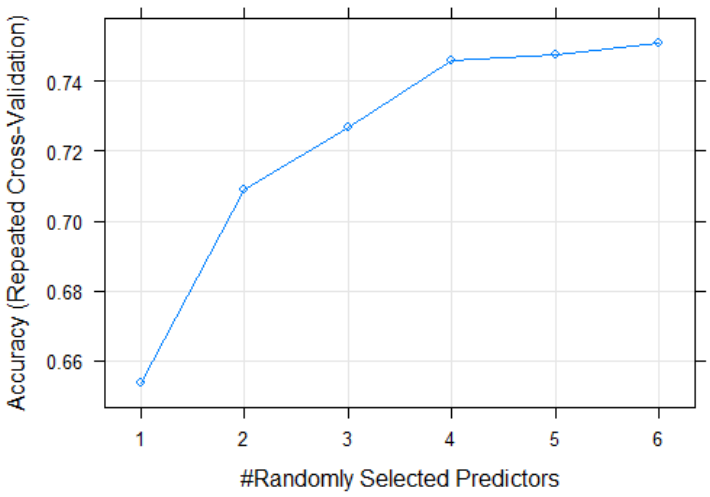

Figura 4.2 – Esempio di differenza di accuratezza a seconda del settaggio degli mtry

Sull'asse delle ascisse della fig4.2 possiamo notare dei numeri che vanno da 1 a 6. Questi numeri si riferiscono agli `mtry` con la quale è stato costruito il modello di predizione. I punti sul grafico fanno poi riferimento all'accuratezza per ogni mtry scelto.

La funzione fondamentale dove andiamo ad addestrare il modello è la seguente:

```
rfDownsampled <- train(Class ~ ., data = dataset_FPCA,
                       method = "rf",
                       ntree = 1500,
                       tuneGrid = tuneGrid2,
                       tuneLength = sqrt(ncol(dataset_FPCA)),
                       metric = "Accuracy",
                       strata = dataset_FPCA$Class,
                       trControl = ctrl,
                       sampsize = campione)
```

- **`Class ~.`**: indica che nell'addestramento del modello andiamo a comparare la caratteristica `Class` rispetto a tutte le altre. In questo modo indichiamo che lo scopo del modello è proprio quello di trovare la classe vegetale di riferimento.
- **`data`**: data-set al quale fare riferimento.
- **`method`** indica il metodo attraverso la quale vogliamo costruire il modello, in questo caso random forest.
- **`ntree`** indica il numero di alberi decisionali da costruire per andare ad effettuare la predizione su ogni singola istanza.
- **`tuneGrid`** indica la *tuning grid* da utilizzare per allenare il modello.
- **`metric`** indica qual è la caratteristica principale del modello, in questo caso l'accuratezza. Il modello finale salvato sarà quindi riferimento quello in cui è stata riscontrata la maggior accuratezza.
- **`trControl`** indica il *train control* da utilizzare.

In seguito al comando `train()` abbiamo a disposizione un modello di predizione delle classi vegetali. In particolare in `rfDownsampled$results` vengono inseriti risultati di questo tipo:

```
mtry  Accuracy     Kappa AccuracySD    KappaSD
1    1 0.5431956 0.4553877 0.04211912 0.04953504
2    2 0.5531215 0.4691002 0.03916910 0.04647552
3    3 0.5503173 0.4660201 0.04291383 0.05027509
```

dove si indicano i principali modelli di predizione che sono stati trovati a seconda dei parametri inseriti. In questo caso l'accuratezza con mtry = 1 è pari a 0.54. I risultati come quelli scritti sopra vengono inseriti all'interno di file come

`F01_11_10_00_00_10_BIL_RISULTATI_MODELLO.txt`

Un'altra caratteristica importante del modello prodotto è la **matrice di confusione**.

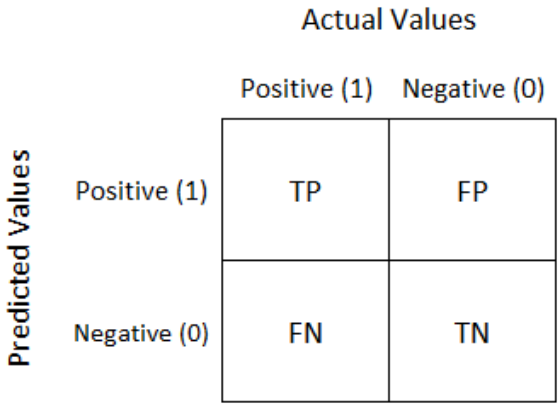

Figura 4.3 – matrice di confusione

La matrice di confusione è uno strumento utile per la misurazione della performance del modello per problemi di classificazione dove le classi sono 2 o più [39]. Per comprendere meglio il contenuto della matrice di confusione facciamo riferimento a un data-set i cui valori su cui effettuare la predizione possono essere di due tipi: vero o falso. La matrice di confusione conterrà principalmente 4 valori:

- ***True positive***: la classe che è stata predetta è vera ed la predizione è giusta.
- ***True negative***: la classe predetta è falsa e la predizione è giusta.
- ***False positive*** (errore di primo tipo): la classe predetta è vera e la predizione è errata.
- ***False negative*** (errore di secondo tipo): la classe predetta è falsa e la predizione è errata [39].

Nel nostro caso la matrice di confusione sarà di questo tipo:

```
  G  J  L  M  O  P  Q  R   class.error
G 9  2  0  2  0  2  1  0   0.4375000
J 0 13  0  2  0  0  0  0   0.1333333
L 1  0 16  3  5  0  4  5   0.5294118
M 7  5  4 17  5  1  5  2   0.6304348
O 3  0  5  2 35  0 11  0   0.3750000
P 0  1  0  0  0 15  0  0   0.0625000
Q 1  0  5  3  5  1 13  0   0.5357143
R 0  0  2  1  2  0  1 25   0.1935484
```

sulla diagonale troviamo su quanti valori è stata effettuata una predizione corretta mentre in tutte le altre posizioni troviamo i valori errati. La colonna finale esprime il tasso percentuale di errore sulla predizione della classe relativa alla riga. In questo caso possiamo notare che la classe su cui è stata effettuata una predizione corretta è la P, mentre la classe in cui le predizioni sono state più errate è la M. Le lettere presenti all'interno del listato corrispondono alle diverse classi vegetali presenti nei territori di studio.

	Categoria fisionomica
Boschi	
L	Boschi a dominanza di Quercus ilex
Q	Boschi a dominanza di Quercus pubescens
O	Boschi a dominanza di Ostrya carpinifolia
R	Rimboschimenti a dominanza di Pinus sp.
Macchia, arbusteti e garighe	
J	Arbusteti a dominanza di Juniperus oxycedrus
G	Arbusteti a dominanza di Spartium junceum
M	Mosaici di vegetazione rupestre, macchia e garighe
Praterie	
P	Praterie a dominanza di Bromus erectus

Tabella 4.1 – Categorie di classi vegetali

Nella tabella 4.1 possiamo notare tutte le caratteristiche delle classi vegetali presenti. Inoltre, per la direttiva Habitat le classi vegetali appena elencate hanno dei precisi habitat in cui si collocano:

- **L**: HABITAT 9340-Foreste di Quercus ilex e Quercus rotundifolia
- **Q** : HABITAT 91AA*-Boschi orientali di quercia bianca.
- **J**: HABITAT 5130: Formazioni a Juniperus communis su lande o prati calcicoli.
- **M**: possono essere presenti i seguenti habitat: HABITAT 8210-Pareti rocciose calcaree con vegetazione casmofitica, HABITAT 6110*-Formazioni erbose calcicole rupicole o basofile, HABITAT 6220*-Percorsi substeppici di graminacee e piante annue.
- **P**: HABITAT 6210*-Formazioni erbose secche seminaturali e facies coperte da cespugli su substrato calcareo

La matrice allegata sopra sarà il contenuto del file:

`F01_11_10_00_00_10_BIL_ConfusionMatrix.txt.`

Capitolo 5

Risultati

5.1 Risultati dei modelli

I modelli di classificazione prodotti nell'elaborazione precedentemente descritta vengono analizzati in questa sezione. Dai file dei risultati sono stati prelevati i valori più alti di accuratezza (*Accuracy*) per ogni indice, mentre nei file contenenti la matrice di confusione sono stati prelevati i minori tassi di errore per ogni classe (*class error*). Queste informazioni che costituiscono il *ranking* dei diversi indici, sono stati salvate attraverso l'utilizzo di uno script apposito, all'interno di un file Excel nel quale sono stati tabulati secondo il formato della seguente tabella 5.1:

Indici	mtry	Accuracy	G	J	L	M	O	P	Q	R
11_10	1	0.510	0.75	0.4	0.676	0.609	0.446	0.125	0.536	0.226
11_08	5	0.643	0.5	0.4	0.235	0.348	0.321	0.1875	0.393	0.516
11_07	5	0.648	0.438	0.467	0.294	0.283	0.411	0.1875	0.25	0.355
...	...	...	...	...	...	...	...	...	...	...

Tabella 5.1 – Formato delle tabelle di *ranking* di ogni funzione. Estratto relativo alla F01. Alcuni valori sono stati arrotondati al terzo decimale per esigenze di spazio

Da questi *ranking* sono state poi estratti gli indici migliori, per ogni funzione, in particolare

- l'indice che massimizza *accuracy* della predizione del modello.
- l'indice che minimizza, per ogni classe vegetale, il *class error*.

I valori migliori prelevati effettuando un semplice ordinamento dei file Excel, per poi essere riassunti tutti su un altro foglio di lavoro. A parità di *class error* è stato privilegiato l'indice con la migliore *accuracy*. Un esempio di risultati migliori, relativi alla funzione F01, sono elencati nella tabella 5.2.

	F01	
	Indice	Valore
Max Accuracy	04_03	0.73108
Min Class Error G	04_03	0.31250
Min Class Error J	10_05	0.20000
Min Class Error L	11_03	0.17647
Min Class Error M	04_03	0.13043
Min Class Error O	08_06	0.26786
Min Class Error P	10_03	0.06250
Min Class Error Q	11_03	0.25000
Min Class Error R	04_03	0.16129

Tabella 5.2 – Esempio di migliori indici e rispettivi valori, tratto dalla formula F01

Sono stati confrontati poi tutti i *ranking* di tutte le funzioni, fino ad arrivare ai valori migliori in assoluto, elencati nella seguente tabella 5.3.

	Indice	Valore
Max Accuracy	F01 - 04_03	0.73107621
Min Class Error G	F09 - 11_04_03_02	0.18750000
Min Class Error J	F23 - 10_07_04	0.06666667
Min Class Error L	F11 - 10_06_05	0.05882353
Min Class Error M	F13 - 10_05_04_03	0.08695652
Min Class Error O	F10 - 11_02_03	0.25000000
Min Class Error P	F01 - 10_03	0.06250000
Min Class Error Q	F17 - 11_05_03_02	0.10714286
Min Class Error R	F23 - 11_10_05	0.09677419

Tabella 5.3 – Migliori indici in assoluto per ogni parametro d'interesse

I risultati dei modelli qui ottenuti considerando l'intero data-set, senza aver applicato particolari elaborazioni alle time-series, possono considerarsi soddisfacenti in quanto la migliore accuratezza rilevata è pari a circa 0.73, tuttavia, come si può osservare dai risultati descritti nella tesi del collega Forconi Riccardo, l'applicazione dell'FPCA univariata (*Univariate Functional Principal Component Analisys*[40]) fornisce un ulteriore miglioramento in questo ambito in termini di *accuracy* e *class error*. Dai *ranking* precedentemente trattati possiamo inoltre osservare che alcune bande ricorrono più di altre tra le combinazioni che identificano gli indici migliori. Qui sotto la classifica nel caso in esame:

1. **banda 3**: 82 volte
2. banda 5: 78 volte
3. banda 4: 77 volte
4. banda 11: 67 volte
5. banda 10: 48 volte

6. banda 2: 46 volte
7. banda 6: 44 volte
8. banda 7: 39 volte
9. banda 8: 32 volte

Alcuni risultati della classifica erano inattesi, come ad esempio la forte presenza della banda 11, che inizialmente era considerata quasi da scartare.

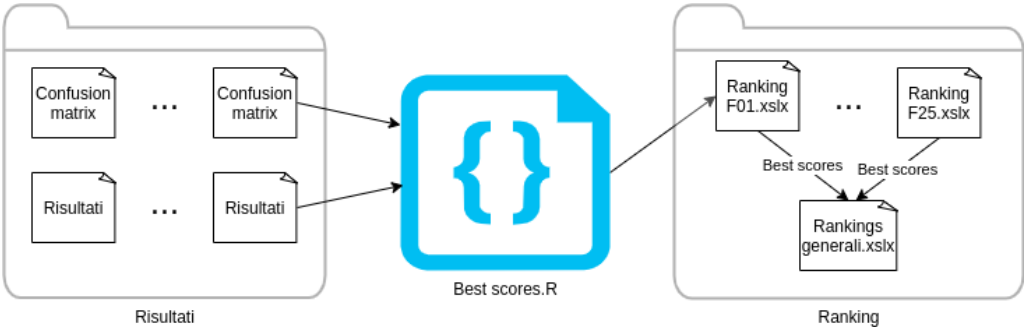

Figura 5.1 – Diagramma per il prelevamento dei risultati

5.2 Applicazione dell'MFPCA

Un ultimo passaggio d'interesse, una volta selezionati per tutte le funzioni gli indici dalle prestazioni migliori in termini di classificazione, è quello di applicare su di essi la MFPCA (*Multivariate Functional Principal Component Analisys*)[41] al fine di migliorare ulteriormente i risultati ottenuti. Per informazioni più dettagliate riguardo la MFPCA, si rimanda alla tesi del collega Colavito Cristian. In questa sezione verrà trattata solo a livello "operativo", illustrando come essa si possa applicare ai dati fin'ora descritti in questa tesi.

Il codice in figura 5.2 si riferisce alla creazione del data-set sul quale è stata applicata la MFPCA relativa alla funzione F01.

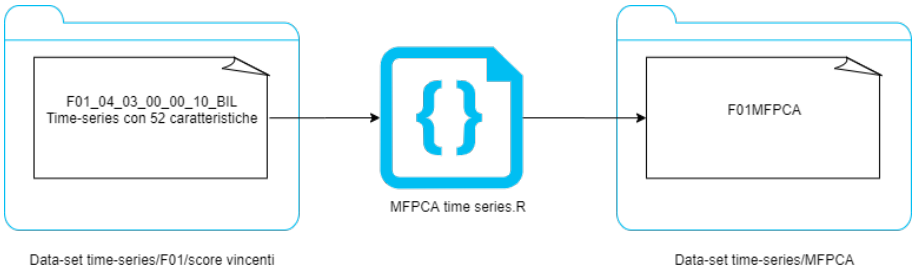

Figura 5.2 – Processo di creazione del data-set con MFPCA

All'interno della cartella `Dataset` si trovano le time-series corrispondenti agli indici migliori. E' possibile averne al massimo 9, 1 che massimizza l'*accuracy* e

8 che minimizzano il *class error*, a meno di eventuali ripetizioni degli stessi.

Gli indici vengono processati all'interno di un ciclo. Molto interessante è l'utilizzo della funzione `scale()`, infatti essa si può utilizzare per normalizzare i valori delle time-series, in questo caso si è utilizzata per migliorare l'accuratezza della MFPCA. Nel capitolo 6.2 si potranno vedere gli sviluppi futuri riguardo questa funzione.

```
ts.f <- funData(argvals = 1:52, X = scale(t(ts)))
```

Al termine del ciclo si ottiene un *multifuncional data* (figura 5.3).

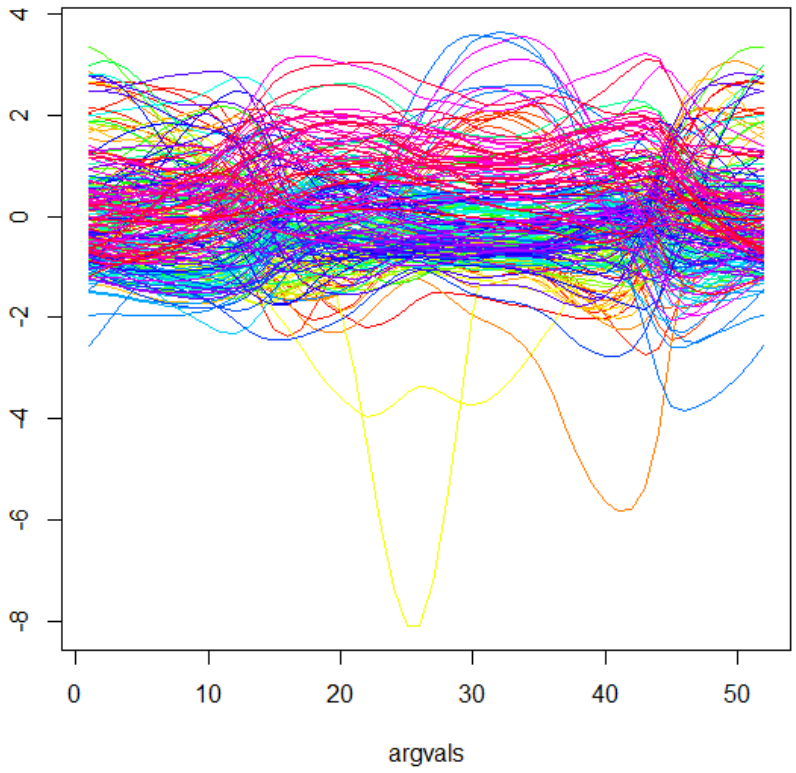

Figura 5.3 – Esempio di grafico di una funzione all' inteno del **Multifuncional data**

La funzione fondamentale dove andiamo ad eseguire la MFPCA è la seguente:

```
MFPCA(mFData, M, uniExpansions,  fit = TRUE, approx.eigen = FALSE)
```

Si hanno vari parametri da poter modificare, i principali sono[42]:

- **`mFData`** va a richiedere un tipo di dato **Multifuncional data**
- **`M`** rappresenta il numero delle componenti principali funzionali multivariate da calcolare.

- **`uniExpansions`** Richiede una lista caratterizzante l'espansione (univariata) che verrà calcolata per ogni elemento.

- **`fit`** rappresenta un valore logico, se impostato a **True**, si otterrà una rappresentazione troncata e i dati vengono calcolati in base ai punteggi stimati e alle autofunzioni.

- **`approx.eigen`**se impostato a **False** la funzione lancia un avviso. Questa è un'impostazione di **default**, altrimenti utilizzerà un'approssimazione utilizzando il pacchetto irlba.

Infine gli **scores** generati si salveranno attraverso le funzioni:

```
fileoutput <- paste('F01_MFPCA.txt', sep = '')
write.table(scores, fileoutput)
```

Al termine della funzione si otterranno gli score che saranno utilizzati, analogamente ai data-set fino ad ora trattati, per generare modelli predittivi con algoritmo random forest.
In questo caso però, ciascun data-set si riferisce ad una singola formula, andandone a comprimere le caratteristiche in 36 componenti. Delle righe di esempio del data-set prodotto possono essere osservate nella tabella 5.4.

	X1	X2	...	X36	CLASS
V1	-10.7334828	-18.75347014	...	0.584822909	Q
V2	-9.4361734	-18.08658785	...	3.262920035	O

Tabella 5.4 – Esempio di riga del file di **output** F01_MFPCA.txt

Di seguito sono listati i risultati relativi ai modelli dopo l'applicazione della MFPCA (5.5).

Formula	mtry	Accuracy	G	J	L	M	O	P	Q	R
F01	5	0.751	0.313	0.200	0.147	0.152	0.286	0.188	0.286	0.194
F02	3	0.777	0.250	0.000	0.118	0.043	0.339	0.063	0.214	0.065
F03	4	0.794	0.250	0.067	0.088	0.043	0.339	0.188	0.250	0.129
F04	3	0.800	0.125	0.000	0.088	0.174	0.339	0.063	0.143	0.129
F05	5	0.792	0.125	0.133	0.029	0.152	0.357	0.125	0.179	0.097
F07	6	0.792	0.125	0.067	0.059	0.043	0.357	0.188	0.250	0.097
F08	6	0.755	0.250	0.133	0.059	0.217	0.429	0.063	0.214	0.161
F09	4	0.793	0.125	0.000	0.088	0.087	0.321	0.125	0.286	0.129
F10	5	0.781	0.250	0.000	0.118	0.087	0.304	0.063	0.321	0.226
F11	6	0.791	0.250	0.067	0.029	0.043	0.482	0.063	0.250	0.161
F13	6	0.781	0.313	0.067	0.147	0.065	0.286	0.125	0.393	0.129
F15	6	0.789	0.125	0.133	0.088	0.109	0.304	0.125	0.250	0.097
F16	5	0.785	0.188	0.000	0.147	0.043	0.321	0.063	0.250	0.097
F17	4	0.785	0.188	0.067	0.088	0.043	0.429	0.125	0.179	0.161
F19	5	0.780	0.188	0.067	0.088	0.022	0.357	0.125	0.286	0.097
F20	6	0.783	0.250	0.200	0.118	0.043	0.321	0.125	0.214	0.129
F21	6	0.797	0.188	0.133	0.059	0.022	0.321	0.188	0.143	0.097
F22	5	0.782	0.188	0.067	0.059	0.065	0.411	0.188	0.286	0.161
F23	5	0.766	0.250	0.200	0.206	0.130	0.321	0.063	0.321	0.161

Tabella 5.5 – Risultati dei modelli di classificazione generati da data-set dopo l'applicazione di MFPCA

Si può notare come le *accuracy* raggiunte siano sopra 0.7, fino ad un massimo di 0.8, più elevate rispetto ai risultati precedenti. Anche i *class error* risultano globalmente migliorati, arrivando anche ad essere zero in alcuni casi.

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

L'impiego della MFPCA risulta aver migliorato le prestazioni dei modelli di classificazione. Ciò detto però, i risultati ottenuti anche solo dalle serie temporali si possono ritenere soddisfacenti.
Inoltre l'analisi esplorativa sui diversi indici ha permesso di trovare combinazioni di bande che fino ad ora non si ritenevano d'interesse ma che si sono rivelate invece in grado di fornire risultati promettenti per il miglioramento della classificazione delle specie vegetali.

6.2 Sviluppi futuri

- In futuro potrebbero essere esplorati modelli di predizione utilizzando diversi altri algoritmi, come ad esempio:
  - **Adaboost**: algoritmo di `boosting` che aiuta a combinare più "classificatori deboli" in un unico "classificatore forte".
  - **ranger**: nuova implementazione dell'algoritmo `random forest` qui trattato, resa disponibile direttamente all'interno del pacchetto `caret`.
  - **KNN**:l'algoritmo KNN (`K-Nearest Neighbors`) assume che elementi della stessa classe si trovino vicini uni con gli altri [43].
- I modelli addestrati potranno essere utilizzati per gestire mappe tematiche nella zona di interesse.
- Come visto nel Capitolo 5.2, si potrà confrontare l'utilizzo della funzione `scale()`, andando a verificare quanto tale operazione influisca sul risultato finale.

Bibliografia

[1] Morisette J.T. Hanes J.M., Liang L. *Biophysical Applications of Satellite Remote Sensing. Springer Remote Sensing/Photogrammetry.* Springer, Berlin, Heidelberg., 2014.

[2] Simone Pesaresi, Adriano Mancini, Giacomo Quattrini, Simona Casavecchia. Mapping mediterranean forest plant associations and habitats with functional principal component analysis using landsat 8 ndvi time series. *Remote Sens.*, 2020.

[3] Ministero dell'ambiente e della tutela del territorio e del mare. Rete natura 2000. https://www.minambiente.it/pagina/rete-natura-2000, 2020.

[4] Ministero dell'ambiente e della tutela del territorio e del mare. Sic, zsc e zps in italia. https://www.minambiente.it/pagina/sic-zsc-e-zps-italia, 2020.

[5] Ministero dell'ambiente e della tutela del territorio e del mare. Direttiva 92/43/cee del consiglio. https://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1992L0043:20070101:IT:PDF, 1992.

[6] ESA. Satellite description. https://sentinel.esa.int/web/sentinel/missions/sentinel-2/satellite-description.

[7] Copernicus. Informazioni su copernicus. https://www.copernicus.eu/it/informazioni-su-copernicus, 2020.

[8] Sandra Lorenz Robert Zimmermann Paul A.M. Nex René Booysen, Richard Gloaguen. *Earth Systems and Environmental Sciences.* Elsevier, 2020.

[9] ESA. Overview. https://sentinel.esa.int/web/sentinel/missions/sentinel-2/overview.

[10] ESA. Sentinel-2. https://sentinel.esa.int/web/sentinel/missions/sentinel-2.

[11] ESA. Spatial resolution. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/resolutions/spatial.

[12] ESA. https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi/processing-levels/level-2.

[13] Gordana Jovanovska Kaplan Ugur Avdan. Object-based water body extraction model using sentinel-2 satellite imagery. https://www.researchgate.net/profile/Gordana_Jovanovska_Kaplan/publication/314119510/figure/tbl1/AS:670480428195846@1536866399263/Sentinel-2-band-characteristics.png.

[14] Wikipedia contributors. Apprendimento supervisionato. https://it.wikipedia.org/wiki/Apprendimento_supervisionato, 2020.

[15] Wikipedia contributors. Ground truth. https://en.wikipedia.org/wiki/Ground_truth, 2020.

[16] ... D. Kaye Mark Ryan M. Talabis. *Information Security Analytics*. Syngress, 2014.

[17] Jorge Leonel. Supervised learning. *Medium*, 2018.

[18] Wikipedia contributors. Random forest. https://en.wikipedia.org/wiki/Random_forest, 2020.

[19] Tony Yiu. Understanding random forest. *Medium*, 2019.

[20] Wikipedia contributors. Serie storica. https://it.wikipedia.org/wiki/Serie_storica, 2020.

[21] Mehrez Zribi Nicolas Baghdadi, Clement Mallet. *QGIS and Generic tools*. Wiley, 2018.

[22] QGIS. https://www.qgis.org/en/site/getinvolved/styleguide.html.

[23] Wikipedia contributors. Qgis. https://en.wikipedia.org/wiki/Ground_truth, 2020.

[24] R-project. What is r? https://www.r-project.org/about.html.

[25] Wikipedia contributors. Rstudio. https://en.wikipedia.org/wiki/RStudio, 2020.

[26] Rproject. Rstoolbox: Tools for remote sensing data analysis. https://cran.r-project.org/web/packages/RStoolbox/index.html, 2019.

[27] Max Kuhn. The caret package. https://topepo.github.io/caret/, 2019.

[28] Project Jupyter. Overview. https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html, 2018.

[29] Wikipedia contributors. File:jupyter logo.svg. https://commons.wikimedia.org/wiki/File:Jupyter_logo.sv, 2020.

[30] chiark greenend. Introduction to putty. https://the.earth.li/~sgtatham/putty/0.74/htmldoc/Chapter1.html#intro, 2020.

[31] PuTTy. https://upload.wikimedia.org/wikipedia/commons/b/b6/PuTTY_icon_128px.png.

[32] CRAN. Raster package di r. https://cran.r-project.org/web/packages/raster/raster.pdf, 2020.

[33] A.Marshak,Y.Knyazikhin,A.Davis,W.Wiscombe,P.Pilewskie. Cloud – vegetation interaction: use of normalized difference cloud index for estimation of cloud optical thickness. http://cybele.bu.edu/download/manuscripts/ndci.pdf, 2000.

[34] Wikipedia contributors. Generalized additive model. https://en.wikipedia.org/wiki/Generalized_additive_model, 2020.

[35] R - smooth a circular periodic time series. https://stats.stackexchange.com/questions/68403/smooth-a-circular-periodic-time-series, 2013.

[36] Wikipedia contributors. Bilinear interpolation. https://en.wikipedia.org/wiki/Bilinear_interpolation, 2020.

[37] Prabhu. Bunderstanding hyperparameters and its optimisation techniques. *Medium*, 2018.

[38] Fortran original by Leo Breiman and Adele Cutler, R port by Andy Liaw and Matthew Wiener. Breiman and cutler's random forests for classification and regression. https://cran.r-project.org/web/packages/randomForest/randomForest.pdf, 2018.

[39] Sarang Narkhede. Understanding confusion matrix. *Medium*, 2018.

[40] Pierre-Louis Bescond. Beyond "classic" pca: Functional principal components analysis (fpca) applied to time-series with python. *Medium*, 2020.

[41] Zhenhua Lin & Hongtu Zhu. Mfpca: Multiscale functional principal component analysis. *Medium*, 2019.

[42] Clara Happ. Multivariate functional principal component analysis for functions on different (dimensional) domains. https://www.rdocumentation.org/packages/MFPCA/versions/1.1/topics/MFPCA, 2018.

[43] Onel Harrison. Machine learning basics with the k-nearest neighbors algorithm. *Medium*, 2018.

[44] Michael A.White, Ramakrishna R.Nemani. *Real-time monitoring and short-term forecasting of land surface phenology.* Elsevier, 2006.

[45] Devin Soni. Supervised vs. unsupervised learning. *Medium*, 2018.

[46] Pierre-Louis Bescond. Principal components analysis (pca), fundamentals, benefits insights for industry. *Medium*, 2020.

Elenco delle figure

Elenco delle tabelle