UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Classificazione di time-series da immagini Sentinel-2 mediante Multi-variate Functional Principal Component Analysis nell'ambito della Land Surface Phenology

Time-series classification from Sentinel-2 images using the Multi-variate Functional Principal Component Analysis in the context of Land Surface Phenology

Relatore:
Dott. Mancini Adriano

Correlatore:
Dott. Pesaresi Simone

Laureando:
Cristian Colavito

Anno Accademico 2019-2020

Alla mia famiglia, che sempre mi ha supportato.
A mia madre, che nei momenti più bui mi ha saputo motivare.
Ai miei amici, che ogni giorno mi stimolano ad alzare l'asticella.

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 **Land 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 dell'habitat naturale[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 (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 Riccardo Forconi e Deplano Lorenzo e al Dott. Pesaresi Simone del D3A.

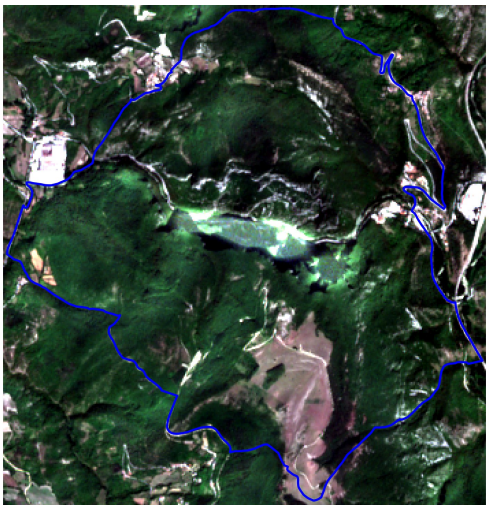

Figura 1.1 – Immagine della Gola di Frasassi rilevata da satellite

Figura 1.2 – Immagine del Monte Conero rilevata da satellite

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 quantificazione 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 mediante la tecnica della *Multivariate Functional Principal Component Analysis* 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*).

Tabella 1.1 – Esempio di formula per la generazione degli indici

Indici	**Formula**	**Operandi**
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 andrà a costruire il modello a partire dal data-set sulla quale è stata applicata una riduzione delle caratteristiche presenti tramite l'MFPCA.

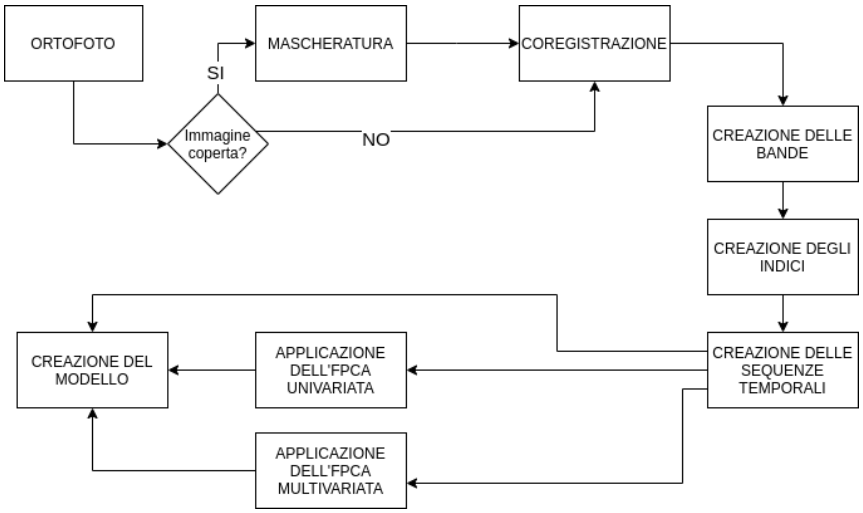

Figura 1.3 – Diagramma del progetto

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 andrà a illustrare il passaggio relativo all'applicazione dell'FPCA multivariata sui migliori indici risultanti dall'FPCA univariata andando a sviluppare il training del modello di predizione.

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.

Nel sesto capitolo si andrà poi a parlare della conclusione del progetto e 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].

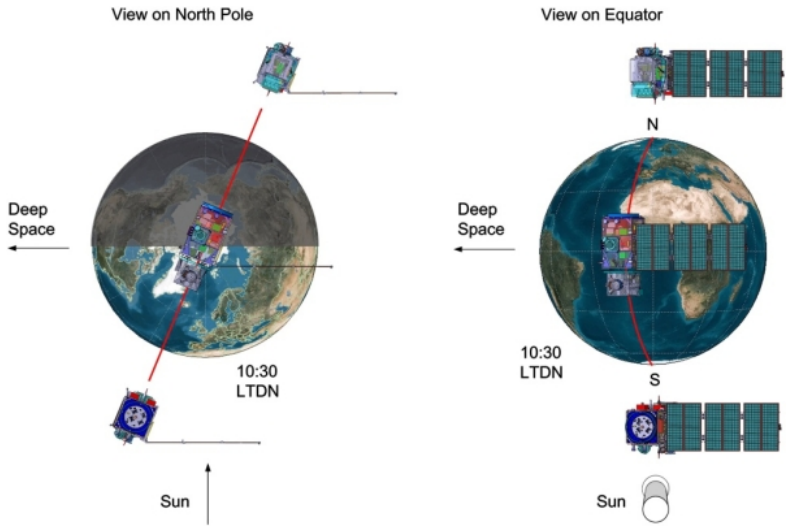

Figura 2.1 – Configurazione orbitale dei satelliti Sentinel-2 [9]

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].

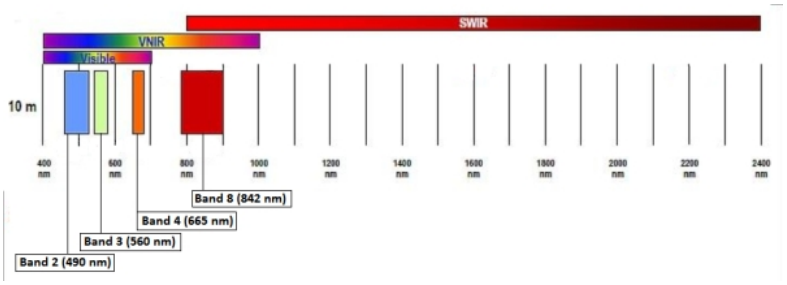

Figura 2.2 – Bande a 10 metri [11]

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].

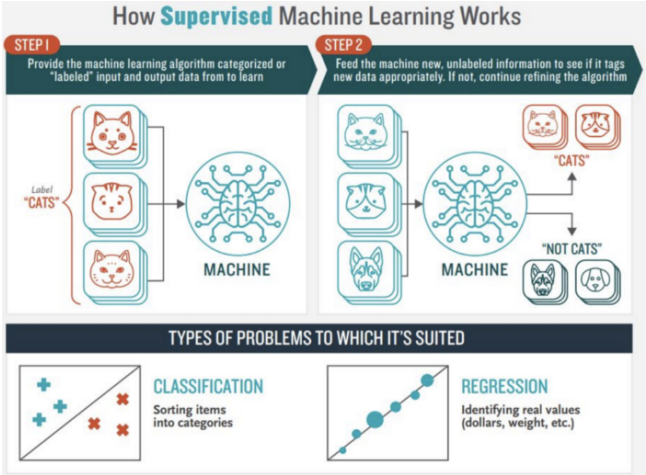

Figura 2.4 – Esempio apprendimento supervisionato [17]

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 4.2).

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.

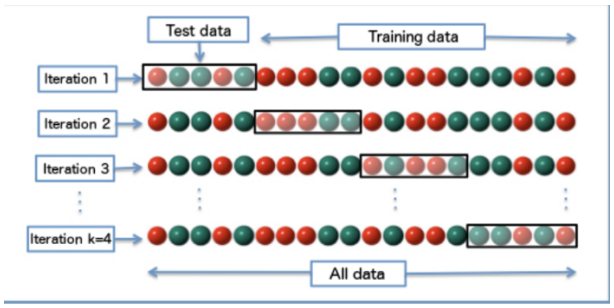

Figura 2.5 – Esempio di divisione del data-set [17]

2.2.2 Random Forest

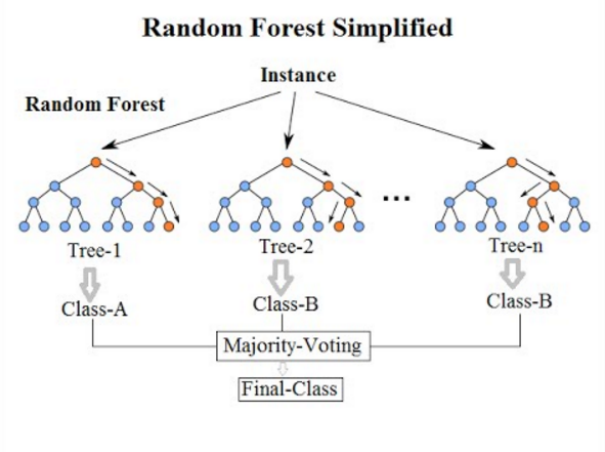

Figura 2.6 – Illustrazione semplificata del funzionamento del random forest [18]

Il ***random forest*** è un classificatore che consiste in un grande numero di alberi decisionali. Ognuno di essi prevederà una classe per ogni determinato oggetto 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'oggetto, effettueranno una predizione corretta [19]. La decisione finale dell'algoritmo si baserà infatti sulla risposta data dalla maggior parte degli alberi decisionali, che nella maggior parte delle volte sarà quella corretta (se il modello è stato costruito correttamente). Gli alberi decisionali che compongono il random forest hanno una bassa correlazione tra loro. Proprio questa loro caratteristica, che rende i risultati degli alberi diversi tra loro, si rivela infine un punto di forza del classificatore, che riesce ad effettuare una predizione a seconda di ciò che hanno predetto la maggior parte degli alberi decisionali. Gli alberi decisionali che compongono il random forest effettuano delle predizioni sui dati basandosi sulle loro caratteristiche. Di conseguenza 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 invece 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 dell'algoritmo.

2.2.3 FPCA

Un concetto principale all'interno del progetto ma in generale anche per quanto riguarda il Data Science è il **PCA (Principal Component Anaysis)**. Il PCA è un metodo attraverso la quale possiamo andare a ridurre, all'interno di un grande data-set, il numero di caratteristiche di quest'ultimo senza andare a perdere di significato [20]. I primi vantaggi dell'utilizzo del PCA sono sicuramente la diminuzione della potenza di calcolo e la riduzione del tempo necessario per la creazione di un modello. Iniziando subito con un esempio di applicazione del PCA possiamo immaginare questo scenario: all'interno di un grafico 3D abbiamo 3 dimensioni: x,y e z. La maggior parte delle variazioni si estendono lungo gli assi y e z, di conseguenza la variazione sull'asse x è minima. Per rappresentare i punti all'interno del grafico possiamo allora escludere la variazione lungo l'asse x, essendo essa di minima importanza per rappresentare i punti, trasformando quindi il nostro grafico in due dimensioni. Questo ci permette di gestire una dimensione in meno senza perdere molte informazioni importanti [20]. La perdita di informazioni sarà ridotta data la bassa variazione dei vari punti all'interno dell'asse x. L'esempio appena fatto ci aiuta a comprendere il funzionamento della PCA e come il suo utilizzo permetta di migliorare le elaborazioni.

L'**FPCA (Functional PCA)** può essere considerato come l'evoluzione della PCA. Il secondo infatti va a diminuire le dimensioni del data-set senza andare ad intaccare la varianza del data-set iniziale mentre il primo fornisce una maggiore importanza allo schema disegnato dall'andamento delle caratteristiche nel lasso di tempo considerato, andando a creare una funzione di riferimento che descrive l'andamento della caratteristica stessa [21].

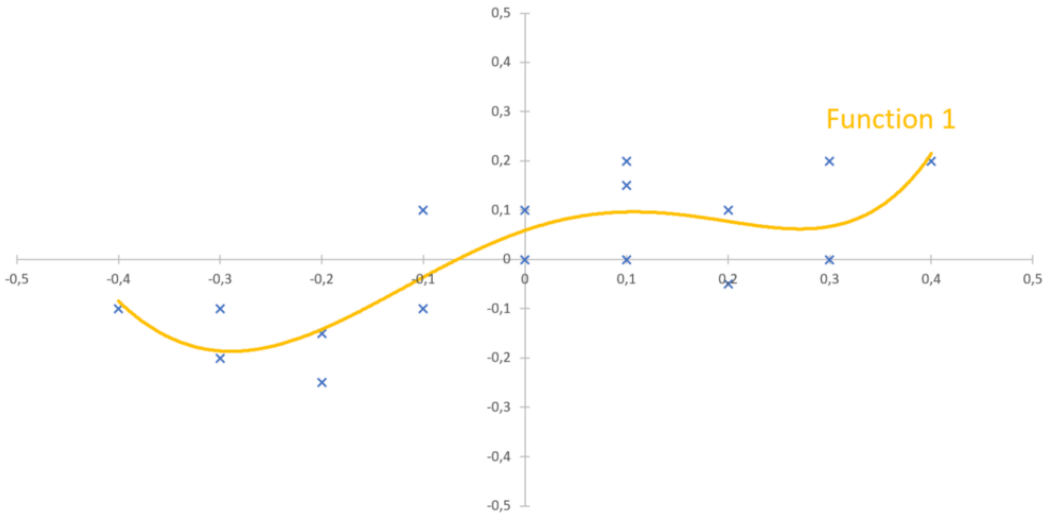

Figura 2.7 – Esempio della creazione della funzione per una caratteristica del data-set [21]

All'interno del progetto in questione è stato utilizzato l'FPCA perché attraverso il PCA la time-series relativa all'andamento delle curve relativi agli indici costruiti nelle bi-settimane sarebbero state considerate come un vettore di dati indipendenti tra loro [2]. Attraverso l'utilizzo dell'FPCA reperiamo:

- autovettori che indicano la variazione spiegata da ogni componente.
- il vettore contenente gli **FPCA scores** che rappresentano la somiglianza tra le istanze nel data-base.
- le **autofunzioni** che rappresentano i principali cambiamenti nei dati [2].

Tra gli elementi prodotti dalla FPCA ci concentreremo maggiormente sugli FPCA scores. Essi svolgono un ruolo centrale all'interno del progetto per le successive analisi effettuate sui dati. In particolare essi verranno utilizzati come *feature* per creare dei modelli per la predizione delle associazioni vegetali nelle zone di studio [2].

2.2.4 MFPCA

Le idee chiave dell' MFPCA (*Multiscale Functional Principal Component Analysis*) è quella di suddividere l'intero dominio in diversi sottodomini in modo che la varianza dei dati all'interno di un sottodominio sia approssimativamente sulla stessa scala, cio significa eseguire una FPCA per ogni sottodominio. Rispetto ai metodi esistenti ad oggi, i tre principali contributi metodologici di questo lavoro sono i seguenti[22]:

- l'approccio con MFPCA è semplice ma potente. In particolare, rispetto all'FPCA su scala singola, l'approccio multiscala allevia il problema della stima delle componenti principali di alto livello e porta ad un miglioramento complessivo dei dati rappresentati, poiché sono necessari meno componenti per avere una sufficiente approssimazione dei dati all'interno di ogni sottodominio.
- L'MFPCA è in grado di scoprire modelli utili nella zona di bassa varianza.
- è Teoricamente dimostriamo che l'MFPCA ha una maggiore capacità di rappresentazione dei dati funzionali e rende di maggior qualità la stima per le componenti principali rispetto all'FPCA su scala singola.

In particolare la MFPCA verrà impiegata sugli indici migliori calcolati con la FPCA.

2.3 Strumenti software

2.3.1 QGIS

QGIS (Quantum GIS) è una applicazione che permette di visualizzare, organizzare, analizzare e rappresentare dati spaziali [23]. 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 [24]

In particolare QGIS supporta formati di file come shapefile [25], e proprio questo sarà il suo utilizzo all'interno di questa tesi. In particolare nelle foto prelevate dai satelliti, sono presenti delle zone in cui ci sono macchie dovute alla correzione topografica. Attraverso gli algoritmi costruiti esse sono sono difficili da escludere, ma proprio attraverso l'utilizzo di QGIS abbiamo la possibilità di effettuare il processo di eliminazione di queste zone 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 [25].

2.3.1.1 Shapefile

2.3.2 R

Figura 2.9 – logo R [26]

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 [26]. 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 [26].

All'interno del software 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 array e matrici;
- Permette l'integrazione di codice C,C++ e Fortran per compiti con alta computazione [26].

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 [27].

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 [28].
- **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 [29].
- **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 [30]. 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 [31]

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.[32]

Figura 2.11 – logo PuTTy [33]

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 per contenere altre proprietà aggiuntive di quest'ultima viene utilizzato uno **shapefile**. Il crop delle immagini viene effettuato sovrapponendole con tale shapefile utilizzando la funzione crop del package Raster[34] 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*)[35], calcolato dalla funzione *cloudMask* del package RStoolBox[28] 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, 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 dove sono state riconosciute le nuvole. Per farlo viene usata la funzione *mask* del *package Raster*[34] di R.

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

Figura 3.2 – Esempio di immagine con nuvole da mascherare

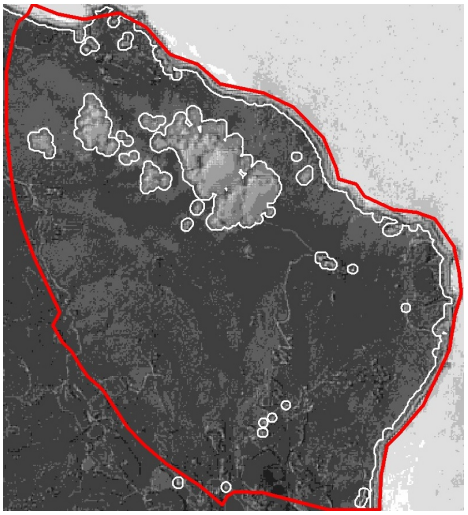

Figura 3.3 – Esempio di maschera con NTDCI filtrato con soglia a 0.2

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

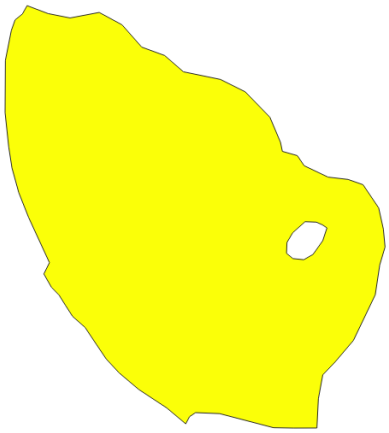

Figura 3.5 – Esempio di shapefile "bucato"

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[34] 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[34] 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, in modo da contenere tutta l'immagine nonostante traslazioni.

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

3.3 Creazione 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

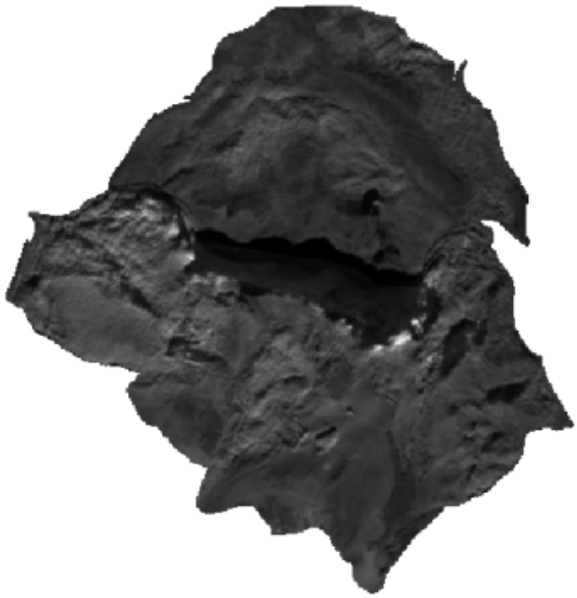

Figura 3.7 – Esempio di selezione della banda NIR

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.

```
1 for (A in bands)
2 {
3   for(B in bands)
4   {
5     if(B >= A){next}
```

```
6     combinazioni <- rbind(combinazioni,c(A,B))
7   }
8 }
```

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.

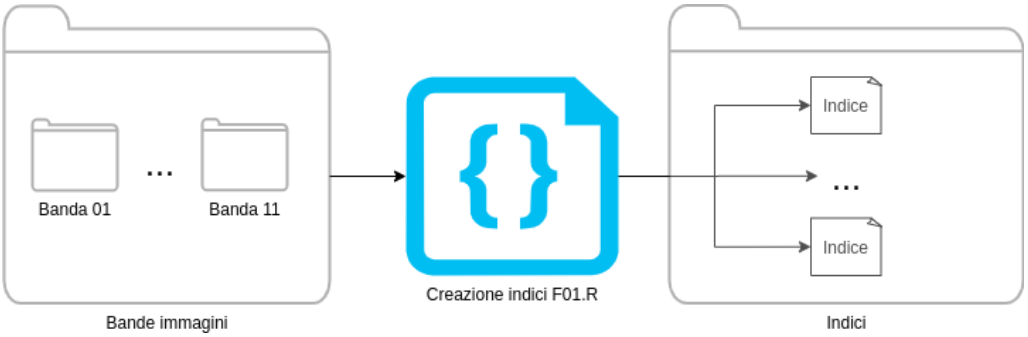

Figura 3.8 – Diagramma dell'esecuzione dello script per la creazione degli indici

`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 terra. 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[34] di R.

```
shapefile_buffer <- buffer(shapefile, 30)#buffer di 30 pixel
...
...
masked_rasters <- mask(rasters, shapefile.buffer)
...
pointsData <- as.data.frame(rasterToPoints(maskedRasters))
```

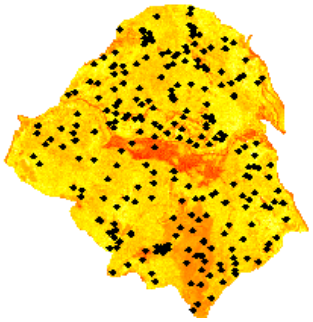

Figura 3.9 – sovrapposizione tra raster di partenza e punti dello shapefile puntuale

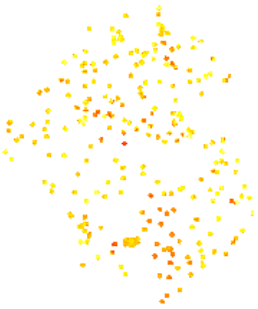

Figura 3.10 – esempio di raster mascherato dopo il buffer sui singoli punti

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[34].

```
1 #aggregazione per settimana, secondo la media (mean), i no-data vengono
      scartati (na.rm = true)
2 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[36] (figura 3.11).

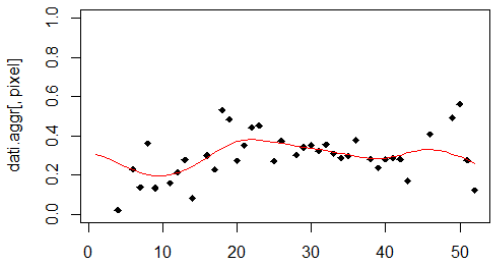

Figura 3.11 – esempio di "lisciatura" dei dati tramite modello GAM

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[37]. Si usa la funzione `extract` del package Raster[34].

```
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')
```

Capitolo 4

Processing del data-set

4.1 Applicazione dell'MFPCA

Grazie al lavoro fornito dal collega Riccardo Forconi sono stati selezionati i modelli migliori dagli **FPCA-scores**, sui quali è stata eseguita la MFPCA.

Il codice in Figura4.1 si riferisce alla creazione del data-set sulla quale è stata applicata l'MFPCA relativa alla funzione F01.
All'interno della cartella `Dataset` si trovano tutte le time-series aventi l'accuracy più alta o con error rate più basso (si veda tabella4.2).

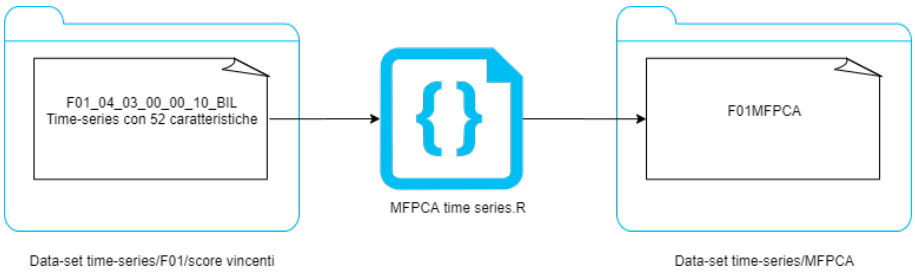

Figura 4.1 – Processo di creazione del data-set con MFPCA

In generale è possibile avere dal data-set preso in esame da 1 a 9 **time series** vincenti, nel caso della F01 ne avremo 5, queste sono:

- 04-02
- 04-03
- 08-06
- 10-06
- 11-05

Di seguito ci saranno i passaggi fondamentali per eseguire lo script della FPCA Multivariata.

Per prendere tutti gli **scores** vincenti è stato usato un ciclo iterativo, all'interno di esso andiamo a togliere le prime quattro colonne della serie temporale, verranno convertiti i numeri in dataframe che poi saranno immagazzinati all'interno di una matrice.

```
ts<- read.delim(var,header = T, sep = " ", quote = "\"", dec = ".")
  ts <- ts[5:nrow(ts),]#tolgo le prime quattro colonne
  ts = as.data.frame(sapply(ts, function(x) as.numeric(as.character(x))))
  assign(paste('ts',i,sep =''), ts) #assegno il nome
  ts <- as.matrix(ts)
```

Si andrà a preparare una lista di funzioni da passare al MFPCA. Molto interessante è l'utilizzo della funzione **scale()**; tale funzione si può utilizzare per normalizzare i valori delle **time-series**. In questo caso si è utilizzato tale parametro per migliorare l'accuratezza della MFPCA. Nel capitolo 6.2 si potranno vedere gli sviluppi futuri di questa funzione.

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

Al termine di tale ciclo si otterà un **Multifuncional data** figura4.2.

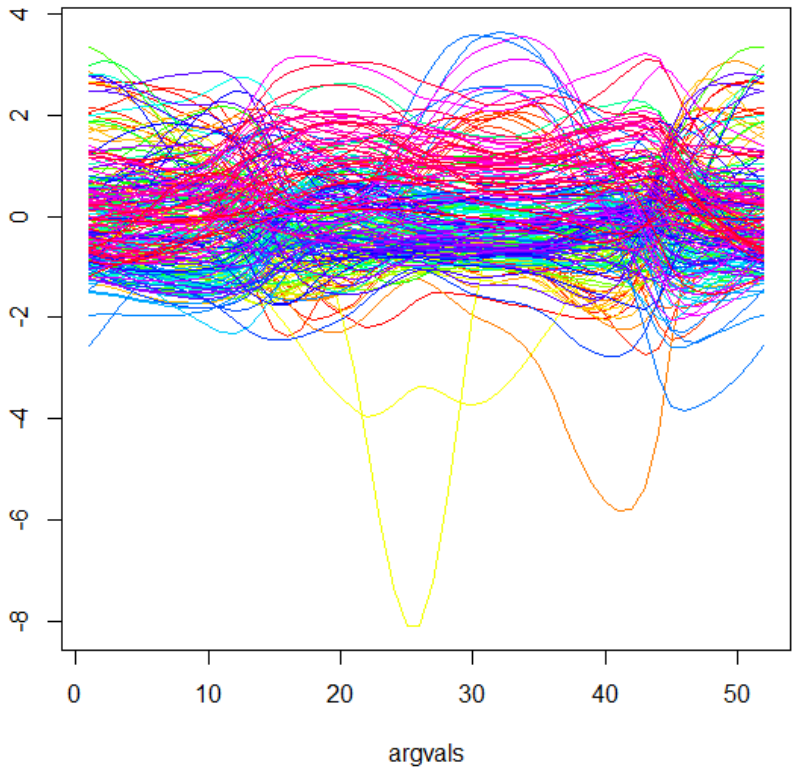

Figura 4.2 – Esempio di grafico di una funzione all'interno del **Multifuncional data**

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

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

Avremo varie proprietà da poter modificare, le principali sono[38]:

- **`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 si andranno a prendere gli **scores** generati e si salveranno attraverso il seguente codice:

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

Al termine della funzione si otterranno gli score che saranno utilizzati nella seguente sezione per andare a creare il modello. Delle righe di esempio del data-set prodotto possono essere osservate nella tabella 4.1. Tale data-set verrà utilizzato nella sezione successiva per andare a creare il modello di predizione delle classi vegetali.

Tabella 4.1 – Esempio di riga del file di **output** F01_MFPCA.txt

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

4.2 Creazione dei modelli

Il codice listato al di sotto di questa sezione si riferisce alla creazione di un modello di predizione della classe vegetale.

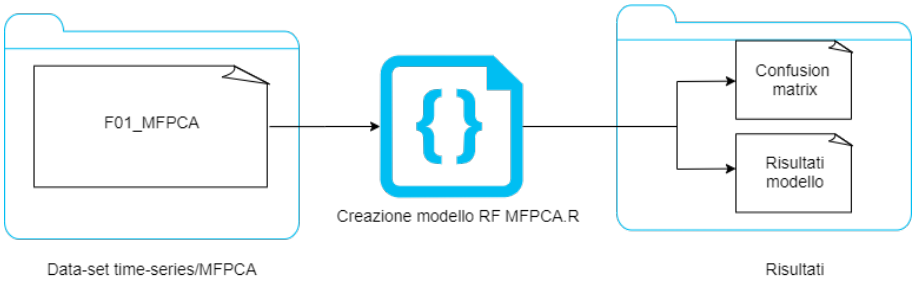

Figura 4.3 – Esempio di creazione del modello e salvataggio dei risultati

Inizialmente si andrà ad istanziare la cartella di lavoro; si effettuerà un ciclo per ogni MFPCA calcolata nella sezione precedente. Attraverso il comando:

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

andiamo a trasformare il data-set in fattore.

Attraverso il comando `trainControl` andiamo a settare i parametri per il training del modello.

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

In questo caso abbiamo utilizzato il **`repeatedcv`** come metodo di training del modello. 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 del modello di predizione che governano tutto il processo di allenamento dello stesso [39]. 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 [40]. A seconda di come l'mtry è stato settato il modello avrà un'accuratezza più o meno alta.

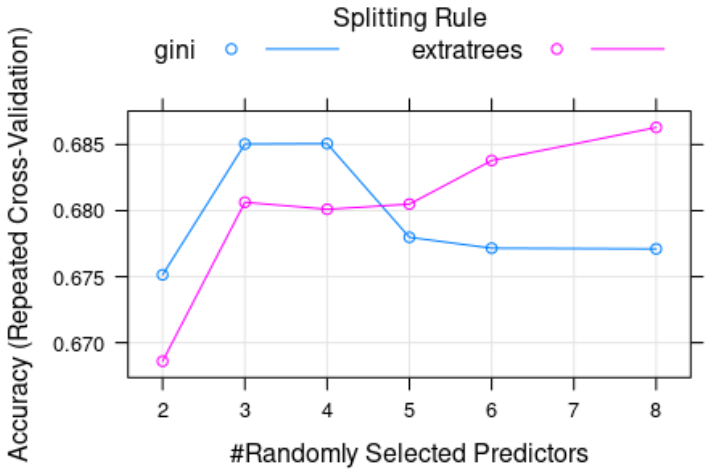

Figura 4.4 – Esempio di differenza di accuratezza a seconda del settaggio degli mtry

Sull'asse delle ascisse possiamo notare dei numeri che vanno da 2 a 8. 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 di riferimento per ogni mtry scelto.

La funzione fondamentale dove andiamo ad allenare 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'allenamento 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 alla 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 farà quindi riferimento a quello in cui è stata trovata la maggior accuratezza generale.

- **`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 0.704169311318 0.638651502682 0.0419160226502 0.0527756758293
2      2 0.767350824028 0.718178736414 0.0335872813995 0.0417588908641
3      3 0.776384981408 0.730198767046 0.0383020418829 0.0468959517819
4      4 0.791312348748 0.748978793072 0.0401251341249 0.0490325824301
5      5 0.795864682890 0.754742586542 0.0355437484943 0.0434458028530
6      6 0.795957346750 0.755140318236 0.0365065805940 0.0446090708099
```

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 0.70. I risultati come quelli scritti sopra vengono inseriti all'interno di file come

`F01_MFPCA_RISULTATI_MODELLO.txt`

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

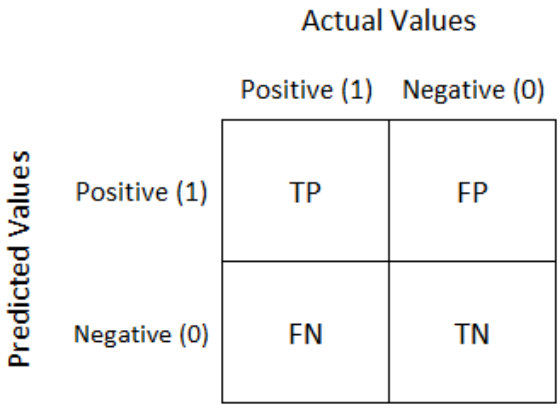

Figura 4.5 – matrice di confusione

La matrice di confusione è uno strumento utile per la misurazione della performance per problemi di classificazione dove le classi sono 2 o più [41]. La matrice di confusione è una matrice che contiene principalmente 4 valori:

- True positive: la classe che è stata predetta è positiva ed è giusta.
- True negative: la classe predetta è negativa ed è giusta.
- False positive (errore di primo tipo): la classe predetta è positiva ed è errata.
- False negativa (errore di secondo tipo): la classe predetta è negativa ed è errata [41].

Nel caso preso in esame, la matrice di confusione basata sulle classi vegetali della tabella 4.2, sarà di questo tipo:

```
   G  J  L  M  O  P  Q  R   class.error
G 11  2  0  1  1  0  1  0  0.3125
J  0 15  0  0  0  0  0  0  0
L  0  0 31  0  2  0  0  1  0.0882352941176471
M  0  1  0 42  0  1  2  0  0.0869565217391305
O  0  0  1  0 44  0 10  1  0.214285714285714
P  1  0  0  1  0 14  0  0  0.125
Q  2  0  0  0  4  0 22  0  0.214285714285714
R  0  0  2  0  1  0  0 28  0.0967741935483871
```

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 J, mentre la classe in cui le predizioni sono state più errate è la G. Le lettere presenti all'interno del listato corrispondono alle diverse classi vegetali presenti nei territori di studio.

Tabella 4.2 – Categorie di classi vegetali

	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

Nella tabella 4.2 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_MFPCA_ConfusionMatrix.txt`.

Capitolo 5

Risultati

Grazie ai risultati del collega Riccardo Forconi ottenuti dall'addestramento tramite **random forest** si sono potuti ricavare:

- l'indice che massimizza l'accuratezza del modello di predizione, prima per ogni formula e poi il migliore in assoluto tra tutte le formule;
- l'indice che minimizza, per ogni classe (associazione) vegetale, il tasso di errore di predizione della classe stessa, prima per ogni formula e successivamente in assoluto tra tutte le formule, nel caso ci fossero 2 indici con lo sesso errore, si è preso quello con accuratezza maggiore;

Ripartendo da quei risultati si sono prese le formule e gli indici migliori con FPCA (vedi tabella 5.1).

Tabella 5.1 – Tabella indici e valori migliori FPCA

BEST ACCURACY OVERALL	$F16/10\_07\_02$	0.747611813
BEST G ERROR RATE OVERALL	$F15/08\_07\_05$	0.1
BEST J ERROR RATE OVERALL	$F22/04\_02$	0
BEST L ERROR RATE OVERALL	$F01/04\_03$	0.058823529
BEST M ERROR RATE OVERALL	$F09/07\_05\_03\_02$	0.043478261
BEST O ERROR RATE	$F01/08\_06$	0.25
BEST P ERROR RATE	$F09/07\_06\_05\_03$	0
BEST Q ERROR RATE OVERALL	$F09/08\_07\_06\_04$	0.142857143
BEST R ERROR RATE OVERALL	$F04/08\_03\_05$	0.064516129

Andando ad effettuare la procedura di addestramento, ma questa volta con MFPCA con il parametro **scale** (vedi capitolo 4.1) sugli indici migliori.
Si è utilizzato uno script che inserisce, all'interno di un file Excel le informazioni dei file contenenti i risultati e le matrici di confusione prodotte nel capitolo 4.2. In questo file Excel, sono stati inseriti i valori più alti di accuratezza per ogni formula dal file contenenti i risultati, mentre nei file contenenti la matrice di confusione, i minori tassi di errore per ogni classe. Possiamo notare i risultati del file Excel nella tabella 5.2.

Tabella 5.2 – Dati presenti nel file Excel relativo alle formule

Formula	01	02	03	04	05
mtry	6	4	6	6	3
Accuracy	0.788201570	0.793405683	0.793054646	0.791791873	0.779053453
Kappa	0.745834326	0.753080750	0.752891393	0.750653940	0.736932870
AccuracySD	0.032565593	0.039855137	0.036931896	0.036931896	0.035167502
KappaSD	0.039274370	0.048271392	0.044539042	0.042903607	0.041326772
G	0.3125	0.3125	0.3750	0.2500	0.1875
J	0	0	0.066666667	0.133333333	0.133333333
L	0.088235294	0.088235294	0.117647059	0.029411765	0.088235294
M	0.086956522	0.021739130	0.065217391	0.195652174	0.065217391
O	0.232142857	0.339285714	0.392857143	0.303571429	0.375
P	0.1250	0.1250	0.0625	0.1250	0.1875
Q	0.214285714	0.285714286	0.285714286	0.285714286	0.214285714
R	0.096774194	0.064516129	0.225806452	0.096774194	0.129032258

Formula	07	08	09	10	11
mtry	6	6	5	3	6
Accuracy	0.795090504	0.784805774	0.760849173	0.777864979	0.778668102
Kappa	0.755030920	0.743117914	0.713771450	0.734732252	0.735366552
AccuracySD	0.034223888	0.038450670	0.031931110	0.046239740	0.040340842
KappaSD	0.041280185	0.046509456	0.038071199	0.055823266	0.049241168
G	0.375	0.3125	0.3125	0.3125	0.25
J	0.066666667	0.133333333	0	0.066666667	0.133333333
L	0.058823529	0.058823529	0.088235294	0.117647059	0.029411765
M	0.086956522	0.217391304	0.065217391	0.108695652	0.152173913
O	0.375	0.285714286	0.392857143	0.303571429	0.321428571
P	0.1875	0.0625	0.0625	0.1875	0.0625
Q	0.107142857	0.214285714	0.285714286	0.214285714	0.107142857
R	0.096774194	0.129032258	0.129032258	0.129032258	0.129032258

Formula	13	15	16	17	19
mtry	6	6	6	5	6
Accuracy	0.807134247	0.788339358	0.775285030	0.786677850	0.758951500
Kappa	0.770028735	0.746500828	0.729882186	0.744439818	0.710939933
AccuracySD	0.032815340	0.037289831	0.040709312	0.031306264	0.032989664
KappaSD	0.039181266	0.045668763	0.050175737	0.038374395	0.040074603
G	0.5	0.3125	0.3125	0.3125	0.375
J	0.066666667	0	0	0.133333333	0.133333333
L	0.117647059	0.058823529	0.058823529	0.029411765	0.029411765
M	0.043478261	0.108695652	0.065217391	0.130434783	0.065217391
O	0.339285714	0.375	0.339285714	0.357142857	0.464285714
P	0.1250	0.1875	0.1875	0.0625	0.1875

Q	0.25	0.285714286	0.321428571	0.178571429	0.25
R	0.129032258	0.129032258	0.064516129	0.129032258	0.161290323

Formula	20	21	22	23
mtry	6	4	5	5
Accuracy	0.792394288	0.794508056	0.802521375	0.755375621
Kappa	0.751614121	0.753693992	0.764191550	0.707338118
AccuracySD	0.043903274	0.041115180	0.042527900	0.048906315
KappaSD	0.053200362	0.050073593	0.051090685	0.059341819
G	0.25	0.1875	0.3125	0.1875
J	0.066666667	0.066666667	0.2	0.266666667
L	0.058823529	0.058823529	0.088235294	0.205882353
M	0.065217391	0.108695652	0.043478261	0.152173913
O	0.339285714	0.375	0.285714286	0.339285714
P	0.1250	0.0625	0.1250	0.0625
Q	0.321428571	0.285714286	0.285714286	0.357142857
R	0.096774194	0.032258065	0.193548387	0.129032258

Questi valori sono stati prelevati effettuando un ordinamento della tabella Excel e i valori trovati ordinandola sono stati salvati all'interno di un altro foglio Excel, dove sono riassunti tutti i migliori risultati.

Tabella 5.6 – Tabella delle formule migliori con MFPCA

BEST ACCURACY OVERALL	$F13$	0.807134247
BEST G ERROR RATE OVERALL	$F21$	0.1875
BEST J ERROR RATE OVERALL	$F02$	0
BEST L ERROR RATE OVERALL	$F04$	0.029411765
BEST M ERROR RATE OVERALL	$F02$	0.021739130
BEST O ERROR RATE	$F01$	0.232142857
BEST P ERROR RATE OVERALL	$F21$	0.0625
BEST Q ERROR RATE OVERALL	$F07$	0.107142857
BEST R ERROR RATE OVERALL	$F21$	0.032258065

Possiamo andare a confrontarla con la tabella 5.7, dove sono riportati i migliori valori con FPCA e MFPCA.

	FPCA		**MFPCA**	
Best Accuracy	**F16/ 10_07_02**	**0.747611813**	**F13**	**0.807134247**
Best G ERROR	**F15/ 08_07_05**	**0.1**	**F21**	**0.1875**
Best J ERROR	**F22/ 04_02**	**0**	**F02**	**0**
Best L ERROR	**F01/ 04_03**	**0.058823529**	**F04**	**0.029411765**
Best M ERROR	**F09/ 07_05_03_02**	**0.043478261**	**F02**	**0.021739130**
Best O ERROR	**F01/ 08_06**	**0.25**	**F01**	**0.232142857**
Best P ERROR	**F09/ 7_06_05_03**	**0**	**F21**	**0.0625**
Best Q ERROR	**F09/ 08_07_06_04**	**0.142857143**	**F07**	**0.107142857**
Best R ERROR	**F04/ 08_03_05**	**0.064516129**	**F21**	**0.032258065**

Tabella 5.7 – Tabella indici e valori migliori del FPCA e MFPCA (con scale)

I risultati del modello di predizione con MFPCA possono considerarsi soddisfacenti in quanto si va a migliorare circa del 6% l'accuratezza. Da notare il fatto che la precisione del modello arriva fino al 81%. Per quanto riguarda il tasso di errore sulla predizione delle classi possiamo osservarle nella tabella 5.8.

Tabella 5.8 – Tabella valutazioni miglioramenti

	Miglioramento %
Best Accuracy	5.95%
Best G ERROR	-8.75%
Best J ERROR	0%
Best L ERROR	2.94%
Best M ERROR	2.17%
Best O ERROR	1.79%
Best P ERROR	-6.25%
Best Q ERROR	3.57%

	Miglioramento %
Best R ERROR	3.23%

Notiamo un netto peggioramento sulle classi G e P, mentre si ha un miglioramento sulle restanti, ulteriori riflessioni sono presenti nel seguente capitolo.

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

L'impiego della MFPCA risulta aver migliorato la predizione rispetto ad un modello con FPCA; detto questo i risultati ottenuti possono ritenersi più che buoni. In base a quanto detto nel capitolo 5 possiamo notare che: Come visto nella tabella5.8, alcune funzioni sbagliano maggiormente con MFPCA. Probabilmente quando si va ad eseguire la MFPCA la funzione MFPCA settando quei parametri,confonderà maggiormente alcune classi vegetali, in particolare dai risultati ottenuti, si può vedere che la classe P, mai sbagliata con la FPCA viene sbagliata con MFPCA.

6.2 Sviluppi futuri

- In futuro potrebbero essere esplorati modelli di predizione utilizzando diversi altri algoritmi. Alcuni esempi di algoritmi che potrebbero essere utilizzati sono:
  - **Adaboost**: tecnica di `boosting` che aiuta a combinare più "classificatori deboli" in un unico "classificatore forte".
  - **ranger**: nuova implementazione dell'algoritmo `random forest` utilizzata nel programma e 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 [42].
- I modelli addestrati potranno essere utilizzati per gestire mappe tematiche nella zona di interesse.
- Come visto nel Capitolo 4.1, si potrà confrontare l'utilizzo della funzione `scale()`, andando a verificare di quanto tale funzione influisce sul risultato finale.
- Si potrà eseguire un modello misto tra MFPCA e FPCA, andando a eseguire la MFPCA solo su alcune classi, in modo da ottimizzare al meglio i punti di forza di tali modelli.

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] René Booysen & Richard Gloaguen & Sandra Lorenz & Robert Zimmermann & Paul A.M. Nex. *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] Pierre-Louis Bescond. Principal components analysis (pca), fundamentals, benefits & insights for industry. *Medium*, 2020.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[35] 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.

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

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

[38] 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.

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

[40] 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.

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

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

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

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

Elenco delle figure

Elenco delle tabelle