UNIVERSITÀ POLITECNICA DELLE MARCHE

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA E DELL'AUTOMAZIONE

Tesi di laurea

Analisi di times series da immagini Sentinel-2 nell'ambito della Land Surface Phenology mediante maching learning

Analysis of times series from Sentinel-2 images in the context of Land Surface Phenology using maching learning

Relatore:
Prof. Adriano Mancini

Candidato:
Andrea Langiotti

Anno accademico: 2021/2022

Meglio curiosi
che intelligenti

Il curioso si meraviglia
L'intelligente capisce e basta

Indice

Capitolo 1

Introduzione

1.1 Concetti Introduttivi

Alcune specie vegetali hanno distinti cicli di vita, determinati dalla caduta, dalla crescita del fogliame e da periodi di intensa attività di fotosintesi. In natura ogni elemento che viene "colpito" da radiazione elettromagnetica assorbe, trasmette e riflette. Nel caso in esame si è interessati alla porzione che viene riflessa in quanto i sensori montati a bordo di droni / aerei / satelliti per il tele-rilevamento sono progettati per misurare la riflettanza. Un modo oramai consolidato per mappare la vegetazione è quello di studiare la riflettanza nello spazio e nel tempo (time-series). In molti lavori come [**4**] viene spiegato quali sono i sensori che permettono di misurare la riflettanza ad esempio il LAI-200-Plant-Canopy-Analyzer.

Lo studio di come la riflettanza varia nel tempo ricade nell'ambito della *Land Surface Phenology* (LSP). In molti casi lo scopo è quello di produrre delle mappe tematiche derivanti dalla classificazione con tecniche di *machine learning* spesso con approcci supervisionati. Solitamente il monitoraggio della vegetazione viene fatto da operatori e tale aspetto è molto oneroso dal punto di vista del tempo e dei costi. Oltre ad avere problemi a raggiungere alcune zone, il rilevamento può essere soggetto anche a bias/errori da parte dell'utente. Una soluzione è quella dunque di automatizzare il processo di generazione di mappe fito-sociologiche. Tali mappe si sono dimostrate essere nel tempo uno strumento fondamentale per il monitoraggio degli habitat naturali e seminaturali, poiché necessitano di poche risorse e il rilevamento ottenuto è migliore del precedente. Il lavoro effettuato su queste aree potrà poi essere applicato a diverse aree che necessitano lo stesso tipo di mappatura quindi offre massima flessibilità. Gli studi di seguito citati fanno riferimento al tele-rilevamento della vegetazione[**1**] e alla mappatura degli habitat forestali[**2**] molto utili per capire quali siano i parametri da analizzare. In [**3**] gli autori definiscono una metodologia di analisi dei dati satellitari di tipo Sentinel-2.

Il progetto Copernicus Global Land Service [**5**] fornisce dati utili al nostro progetto.Viene fornito,ad esempio attraverso l'elaborazione dei dati è possibile calcolare indici vegetazionali come ad esempio l'NDVI che forniscono utili indicazioni circa lo stato della vegetazione.

1.2 Obiettivi Progetto

L'obiettivo del progetto è quello di processare un set di immagini satellitari di Sentinel-2 con lo scopo di classificare le specie vegetali a supporto della crazione di mappe fito-sociologiche. Le immagini sono acquisite nell'areale noto come Val Montagnana (provincia di Ancona). Le immagini scelte su diversi anni consentono di coprire il maggior numero di settimane possibili. A partire da tali immagini attraverso l'utilizzo di librerie R e di notebook JupyterLab vengono applicate diverse tecniche con lo scopo di sviluppare dei modelli atti a classificare ogni singola time-series. Viene utilizzato un approccio che consente di ridurre la dimensionalità; tale approccio è noto in letterature come Functional Principal Component Analysis (FPCA); in seguito a partire dagli FPCA score si crea un modello di apprendimento supervisionato a supporto della classificazione. In 1.1 si riporta l'area di studio.

Figura 1.1: Valmontaganna illustrata con mappe geografiche diverse

Le immagine illustrano la stessa zona d'interesse con mappe geografiche diverse. A sinistra abbiamo un immagine satellitare di Google Earth. La seconda è una mappa 2D creata in Qgis. Nella terza immagine sono presenti le classi di vegetazione che sono state rilevate.

Capitolo 2

Software e Strumenti Utilizzati

2.1 R linguaggio

Linguaggio di programmazione open source.Usato per calcoli statistici e di analisi dei dati. Una librerie che abbiamo utilizzato spesso è `raster` che ci permette di leggere,scrivere ,analizzare e modellare dati spaziali.

2.2 Jupiter Notebook

È un ambiente virtuale per operazioni di *Data scientist*. Tale applicazione lavora in local host del tipo: http://localhost:8891/Progetto1.ipynb.Un ambiente virtuale più sofisticato è il JupyterLab ,cioè un IDE facile da usare.

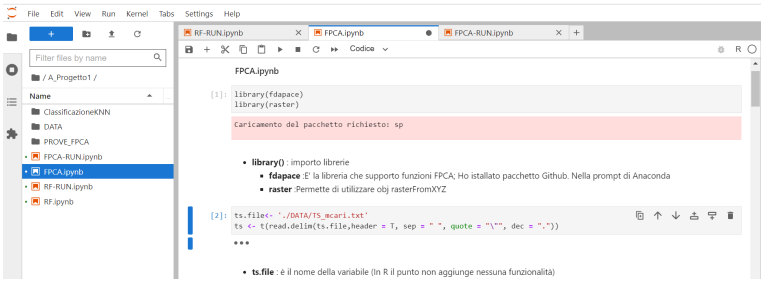


Figura 2.1: ambiente di sviluppo Integrato Jupyter Lab

Tale Notebook è costituito da celle, ognuno dei quali può ospitare codice di qualsiasi linguaggio.Nel nostro caso R. Ogni cella può essere eseguita indipendente dalle altre.Sono messe in esecuzione una dopo l'altra. Per commentare le celle si utilizza il Markdown, che ha una specifica sintassi da rispettare. Un notebook al suo interno contiene un kernel, ovvero un motore operativo che esegue il codice. In questo progetto utilizziamo Rkernel il kernel dedicato al linguaggio R.

Per installare Jupiter Notebook ho utilizzato la distribuzione Anaconda, una piattaforma che offre molte applicazioni per il Data scientist. Per l'installazione di specifici kernel o librerie aggiuntive opero direttamente con il Prompt di Anaconda.

2.3 Anaconda

Distribuzione open source che viene installata nel proprio sistema operativo (Windows o altro)usata nel ambito del *Data Scientist*.Contiene applicazioni come JupyterNoteBook, Oracle, R documentation e molte altre applicazioni in un solo ambiente virtuale. In questo ambiente è molto facile importare librerie che non sono contenute di default in Jupyter ,dalla prompt (terminale) di Anaconda siamo in grado di importare librerie e funzionalità aggiuntive senza problemi. Un'applicazione simile è Visual Studio Code (VSC) ma il progetto avevamo bisogno di librerie R specifiche, ad esempio `raster` e `fpace`, che non potevano essere implementate in VSC. Esiste un altro *framework*simile ad `Anaconda` che occupa meno memoria ed è più performante: `Mamba`.Lo svantaggio è che offre meno funzioni di Anaconda.

2.4 Qgis

QGIS è un Sistema di Informazione Geografica Open Source facile da usare. Rilasciato sotto la GNU General Public License. QGIS è un progetto ufficiale della Open Source Geospatial Foundation (OSGeo).Nel progetto ho utilizzato XYZ-Title per creare una Open Street Map. Sopra a tale mappa andremo a importare un immagine .tif dove sono presenti le varie vegetazioni classificate. Nella figura 2.2 si notano i vari pattern colorati che identificano diverse specie vegetali. Importante è definire in quale emisfero ci troviamo. Con la denominazione EPSG:32633 intendo la nostra zona di interesse.Nel capitolo "Elaborazione del Progetto" vengono forniti maggiori dettagli.

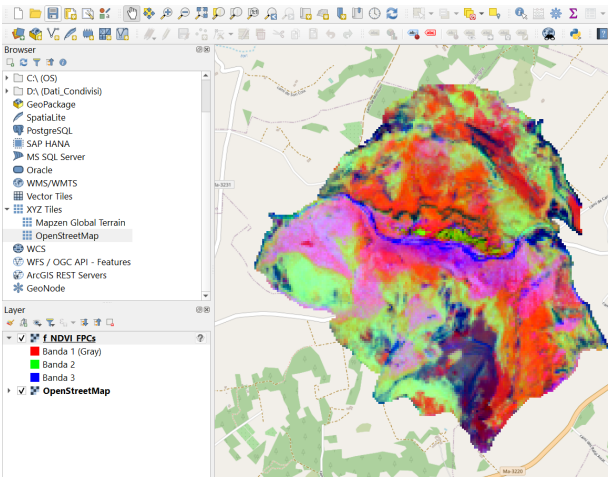


Figura 2.2: Qgis, a sinistra creo una OpeanStreetMap, a destra visualizzo oggetto f-NDVI-FPCs

2.5 Sentinel 2A-2B

La European Space Agency (ESA) ha sviluppato la missione Sentinel2 nel programma spaziale Copernicuous. Tale costellazione orbitale di Satelliti sono volti al monitoraggio delle aree verdi del pianeta Terra. Tali telescopi sono posizionati 180° l'uno rispetto all'altro e hanno entrambe un orbita polare e una detta orbita eliosincrona.L'ESA nel suo sito personale ci fornisce i dettagli tecnici:

- Sentinel 2A è stato lanciato nel 23 Luglio del 2015 e Sentinel2B nel 7 Marzo entrambi con il razzo Vega.
- Ha una risoluzione di 10 m ,possiede 13 bande spettrali con ampia risoluzione pari a 290 km e ogni 5 giorni i dati vengono revisionati.

Tramite le consultazione ufficiale dell'ESA[**4**] abbiamo settato alcuni parametri, ad esempio il valore delle bande.

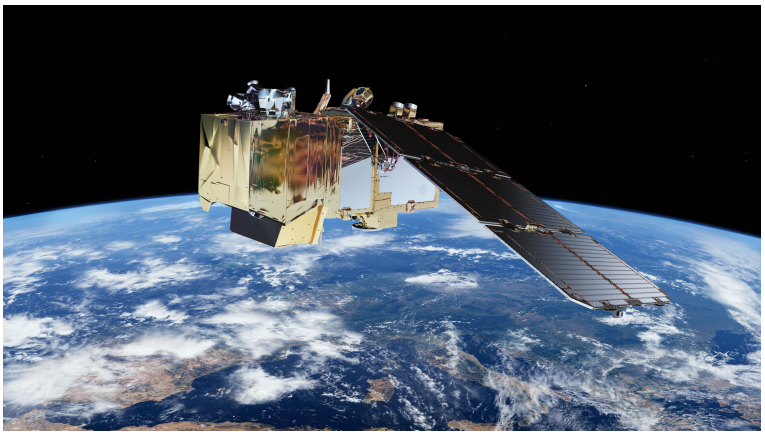


Figura 2.3: Rappresentazione artistica relativa ad un satellite della costellazione Sentinel-2, fonte: ESA

Capitolo 3

Algoritmi Utilizzati

3.1 Riduzione della dimensionalità dei dati

Con riduzione della dimensionalità si intende la trasformazione dei dati da uno spazio ad alta dimensione in uno spazio a bassa dimensione in modo che la rappresentazione a bassa dimensione conservi alcune proprietà significative dei dati originali,riducendo così lo spazio di archiviazione e maggiore velocità di elaborazione dei calcoli. Questa procedura è un operazione di pre-elaborazione del dataset nell'apprendimento automatico non supervisionato. Per parlare di riduzione della dimensionalità dei dati abbiamo bisogno di definire la grande quantità di dati in ingresso.

Nel nostro progetto avremo un dataset di immagini della stessa area, ma acquisiti in tempi differenti . Al fine di avere una rappresentazione compatta della serie temporale è necessario adottare una tecnica di riduzione della dimensionalità che cerchi di preservare anche l'ordinamento temporale dei dati. In particolare si fa riferimento alla tecnica della FPCA.

Definiamo inizialmente un'immagine multi-temporale con 3 coordinate x y e t, il tempo. Costruendo così un layout multiplo di immagine come in figura 3.1. Nella figura 3.2 vengono rappresentati alcuni andamenti temporali dei pixel,rispetto alle 52 settimane.Queste rappresentazioni temporali sono proprio le time series che stiamo cercando,vedi capitolo "Creazione Times Series" per maggiori dettagli.I valori iniziali delle times series sono discreti, li

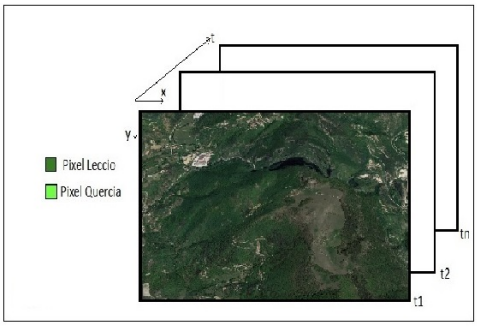


Figura 3.1: Rappresentazione concettuale delle immagini multi-temporali

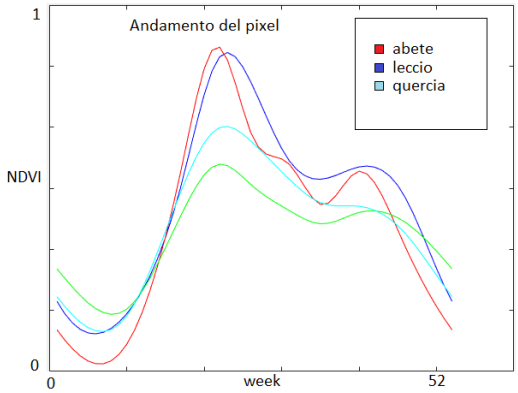


Figura 3.2: Andamento delle times series.Nell'asse x abbiamo le 52 settimane, nell'asse y troviamo i corrispettivi valori di NDVI tra 0 e 1

abbiamo resi continui con operazioni di interpolazione.Ogni valore del pixel, rispetto al tempo, corrisponderà un valore di NDVI.La NDVI è una funzione che associa ad ogni valore dei pixel, un solo valore compreso nell'intervallo [0,1]. Questa funzione è un indice di vegetazione differenziale normalizzato (NDVI),un indicatore del grado di verde dei biomi:

$$NDVI = (NIR - RED)/(NIR + RED) \tag{3.1}$$

Dove NIR e RED, sono rispettivamente la riflettanza misurata nella banda dell'infrarosso vicino e del rosso.

Definito l'andamento temporale dei vari pixel (time-series), attravero la FPCA si ottengono gli FPCA scores. Questi score vengono prodotti tramite delle Funzioni di Analisi Principali delle Componenti. Ogni score è definito in un diagramma cartesiano, come si nota nella figura 3.3. Ogni score è visualizzabile attraverso uno scatter plot scegliendo opportunamente una coppia di componenti(es. FPCA.comp[1] e FPCA.comp[2]). L'utilità di tale trasformazione è di lavorare con dei punti invece che con tanti grafici poiché le procedure di analisi saranno più rapide e la dimensione in spazio di archiviazione sarà minore. C'è da precisare che uno score può essere rappresentato da N-coordinate in funzione del numero di componenti estratte; noi ci limitiamo a usarne solo due. Le altre coordinate FPCA3,FPCA4..etc. saranno ignorate. Verranno scartate perché uno score a due coordinate definisce l'andamento del pixel al 90% e per un approccio ingegneristico può essere accettato.Tale concetto è detto Frazione Varianza Spiegata.

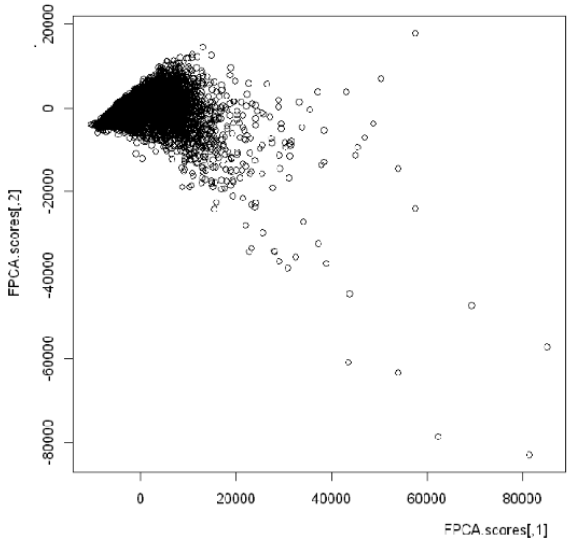


Figura 3.3: Scatter plot di score utilizzando due componenti.

3.1.1 PCA

Per definire il concetto di Analisi Componenti Principali (PCA) dobbiamo introdurre un problema che si ha manipolando *dataset*. Di solito sono complessi e ricchi di dati.Quindi abbiamo bisogno di definire degli algoritmi volti

alla riduzione della dimensionalità dei dati sia nella fase di elaborazione sia nella fase di raccolta dei dati (*storage*). Un algoritmo in grado di ridurre la dimensionalità di un dataset è la Principal Componet Analysis (PCA). In parole semplici si calcola il peso da attribuire a ogni variabile di partenza, in modo da poterle concentrare in una unica serie di nuove variabili (componenti principali) che saranno una combinazione lineare delle variabili di appartenenza.La combinazione lineare PCA non cambia il valore originale, lo ridefinisce con altri valori. In pratica una PCA definisce il dato originale rispetto alla sua variazione in termini probabilistici.Useremo anche la tecnica FPCA cioè un'evoluzione della PCA che fornisce una maggiore importanza all'andamento delle caratteristiche nel lasso di tempo considerato, andando ad applicare un approccio funzionale che preserva l'ordinamento temporale dei dati.

PCA in statistica

Questa tecnica di riduzione dei dati viene applicata nell'ambito Informatico ma ha una solita definizione in Statistica.Per definire questi concetti vedi [**9**] [**10**]. Parleremo di variabile aleatoria o casuale $X = (X1, X2, X3, ..., Xd)$ di una popolazione con media $\mu = E[X]$ e varianza $\sigma = \sqrt{(}\sum_{n=1}^{N}(X_i + \mu)/N)$.

In realtà, nel ambito Probabilistico, si lavora con una variabile detta normalizzata: z ; legata dalla variabile x , media e varianza. Z = (X*μ)/σ

Definita una altra variabile aleatoria Y costruisco una matrice di covarianza Cov(x,y) e i coefficienti di correlazione delle variabili ρ(x,y).

Date p variabili aleatorie Y1,..., Yp cioè le componenti principali sono particolari combinazioni lineari delle variabili originali X1,...,Xn . Tali combinazioni lineari devono descrivere una trasformazione lineare delle variabili che proiettano quelle originarie in un nuovo sistema cartesiano nel quale le variabili vengono ordinate in ordine decrescente di varianza. La riduzione della complessità del sistema avviene limitandosi ad analizzare le componenti principali.

Sia X = [X1, ..., Xp] , un vettore aleatorio, insieme alla matrice di covarianza con autovalori $\lambda 1 \leq \lambda 2 \leq ... \leq \lambda p \leq 0$ andiamo a costruire le componenti principali cioè costruiamo la combinazione lineare Y rispetto alle variabili originali X:

Y1 = a11*X1 + a12*X2 + ... + a1p*Xp

Y2 = a21*X1 + a22*X2 + ... + a2p*Xp

Yp = ap1*X1 + ap2*X2 + ... + app*Xp

Var(Yi) = ai^T *Σ ai ,i=1,...,p

Cov (Yi,Yk)= ai^T Σ ak

3.1.2 FPCA

Definito il concetto di componente principale (PCA), si vuole dare più importanza alle curve che al valore delle componenti principali stesse. Per definire le curve FPCA si faccia riferimento alla pubblicazione [**11**].

FPCA cerca di effettuare una riduzione della dimensionalità dei dati ma rispetto alla PCA si utilizzano delle auto-funzioni cercando di preservere l'ordine temporale dei dati. Con l'analisi PCA lavoriamo con i vettori, invece con le FPCA lavoriamo con delle funzioni. Questo significa che le componenti principali diventano funzioni curve. Qui descriviamo quali sono i passaggi relativi all'analisi basata su FPCA:

1. Sia X(t) = [X1(t),...,Xn(t)] il nostro campione, la prima curva $\phi 1$ è detta zero mean curve
2. La prima FPCA score sarà:

$$s_{1_i} = \int \phi_1 \cdot X_i(t)\, dt$$

3. La prima curva $\phi_1(t)$ è stata trovata massimizzando la varianza dello score di varianza $var(s_1)$ rispettando tale vincolo: $\int \phi_1^2\, dt = 1 \equiv \|\phi_1^2\|$

Quindi la prima FPCA si trova massimizzando la varianza del primo valore di FPCA-scores. Ora troveremo le n-curve successive, aggiungendo un altro vincolo. Costruisco le successive curve in questo modo:

1. Sia X(t) = [X1(t),...,Xn(t)] la seconda curva $\phi 2$ è detta zero mean curve
2. La seconda FPCA score sarà:

$$s_{2_i} = \int \phi_1 \cdot X_i(t)\, dt$$

3. La seconda curva $\phi 1$ è trovata massimizzando la varianza $Var(s2)$ tale che : $\int \phi_2^2\, dt = 1$ e $\int \phi_1 \cdot \phi_2\, dt = 0$, il secondo vicolo

Abbiamo trovato il modo per costruire tutte le altre curve:

$$X_i(t) = \sum_{k=1}^{\infty} S_{i_k} \cdot \phi_k(t)$$

A questo punto si applica lo stesso procedimento a tutte le successive curve. Solo che, a differenza della prima curva devo rispettare un vincolo in più, cioè la seconda curva FPCA2 deve essere ortogonale alla prima FPCA1. In effetti, tutte le FPCA devono essere ortogonali tra loro. Ora ogni curva può essere rappresentata come una combinazione lineare delle FPCA. Quindi la FPCA-score ci dice in che misura una curva è costituita dalla rispettiva componente principale funzionale.

Ora andiamo a visualizzare tali oggetti:

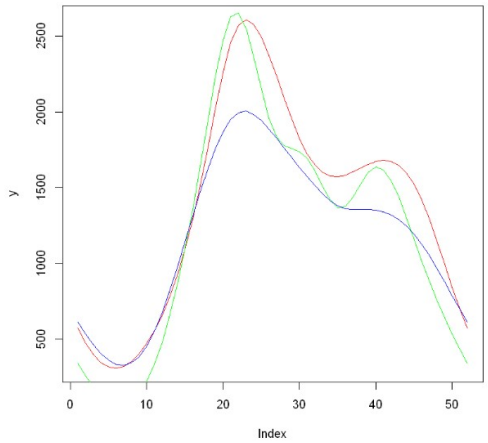


Figura 3.4

- Plottiamo (*plotting*) alcune delle tantissime times series 3.4, cioè le variabili originali da trasformare. Realizzate dal comando:

```
y <-ts[1:52] y1 <-ts[1:52,2] y2 <-ts[1:52,3]
plot(y,type = "l", col= "red")
lines(y1,col="green") #unisce grafici in uno solo
lines(y2,col="blue")
```

- Nella figura 3.5 vediamo le curve $\phi 1$, $\phi 2$ e $\phi 3$.Realizzato dal comando:

```
plot(fpcaObjTS)
```

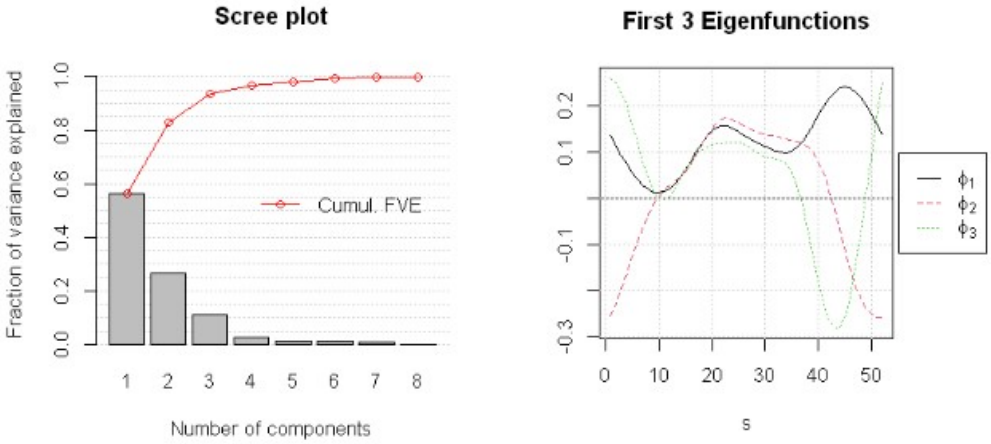


Figura 3.5

- Nella figura 3.6 troviamo gli FPCA score ottenuti:

```
plot(FPCA.scores)
```

Per vedere come implementare o visualizzare una PCA (o FPCA) si faccia riferimento al capitolo "Elaborazione del Progetto".

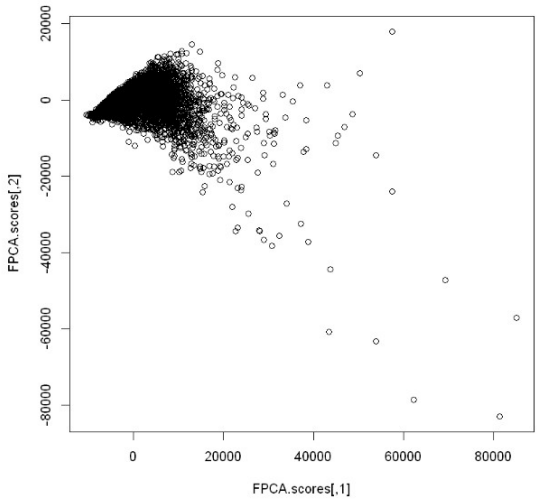


Figura 3.6

3.1.3 Multivariate Functional Principal Analisis - MFPCA

Inizialmente si definisce una "Multivariate Functional Principal Analisis" [**14**] in statistica [**13**] [**14**];cosi da implementarla successivamente nell'ambiente software con linguaggio R.

Questa funzione calcola un'analisi funzionale multivariata delle componenti principali (MFPCA) basata su n-osservazioni $x1, ..., xN$ di una popolazione $X = (X(1), ...X(p))$ con elementi $X(j) \in L^2(Tj)$ e $Tj \subset IR^d$.

In particolare, gli elementi possono essere definiti su domini (dimensionali) diversi. I risultati contengono la funzione media, la stima delle componenti principali funzionali multivariate $\psi^1, \ldots, \psi^M$; gli autovalori associati: $\lambda^1 \geq \ldots \geq \lambda^M > 0$ e gli scores: $\rho_{i_m} = \langle x_i, \psi_i \rangle$.

Inoltre,le traiettorie stimate per ogni osservazione, sono basate sulla rappresentazione troncata di Karhunen-Loeve; se vogliamo tale troncatura usiamo (fit=TRUE): $x_i = \sum \rho_{i_m} \cdot \psi$.

Le curve ψ^i, usano la classe multiFunData , il quale è definita nel *package* funData. Prima abbiamo dato un significato statistico alla MFPCA ora vediamo come usarlo per linguaggio R;Vedi articolo [**10**] ,per definire meglio la libreria:

```
MFPCA(
mFData,   #una multiFunData object contiene N-osservazioni
M,        #numero delle MFPCA da calcolare
uniExpansions,
weights = rep(1, length(mFData)),
fit = FALSE,
approx.eigen = FALSE,
```

)

Un interessante lavoro circa l'utilizzo delle tecniche presentante in questo capitolo è quello presente in [**15**].

```
#0_Librerie utili
library(fdapace),library(raster,library(funData),library(MFPCA)
#1_Costruisco funzione manualmente
set.seed(2)
sim <- simMultiFunData(type = "weighted",
argvals = list(),
M = list(c(4,5), 20), eFunType = list(c("Fourier", "Fourier"), "Poly"),
eValType = "exponential", N = 150)
#2_Visualizzo i risultati
pca <- MFPCA(sim$simData, M = 10,
uniExpansions = list(list(type= "splines2D",
,k = c(10,12)),list(type = "uFPCA")))
summary(pca)
plot(pca)
scoreplot(pca)
```

Le immagini, 3.7, visualizzano le componenti principali derivanti dall'analisi funzionale multi-variabile, in due modi diversi. L'immagine a sinistra summary(pca) mostra le componenti principali rispetto al suo valore e la sua frequenza, cioè nella zona blu la cardinalità delle componete principale è minore rispetto alla zona rossa. Ci fornisce un'idea immediata sui valori che ripetono più spesso e la cardinalità delle componenti principali. L'immagine a destra scoreplot(pca) mostra gli scores, che vengono creati uno dopo l'altro, perciò troviamo il numero corrispondente per ogni punto.

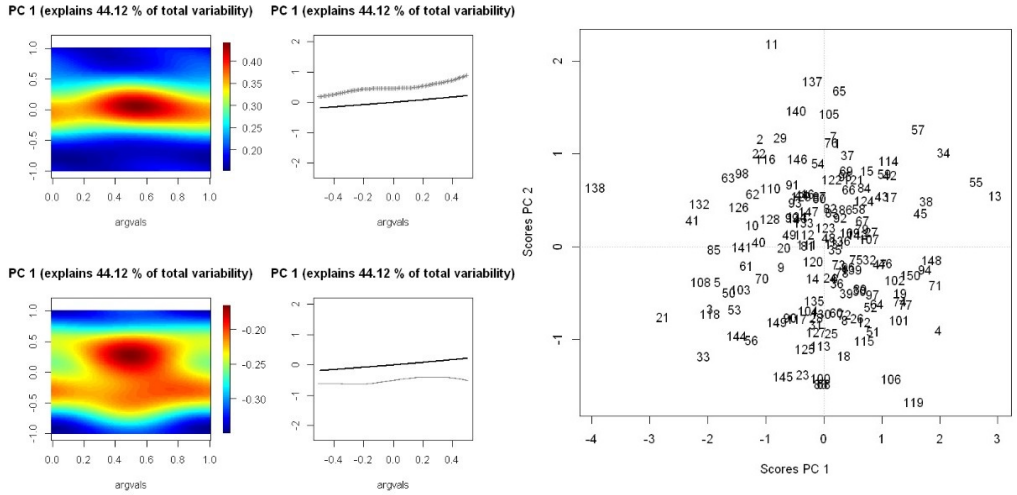


Figura 3.7:

3.2 Algoritmi di classificazione

3.2.1 Alberi decisionali

Nell'ambito del machine Learning applichiamo algoritmi di classificazione uno di questi sono gli alberi. Un albero (tree) è diviso da nodi.Ogni nodo può contenere un dato o un espressione. Possiamo suddividere i nodi in livelli.I nodi sono collegati da rami (link)e possiamo definire diversi percorsi (path).Si può definire la profondità del albero (deep). Possiamo lavorare anche con più alberi. Con albero decisionale si intende un particolare albero che prende in input una oggetto , ne verifica la condizione e ne restituisce la decisione.

3.2.2 K-Nearest Neighbors

L'algoritmo k-nearest neighbors, è un algoritmo per classificazione usato dal modello predittivo, ha lo scopo d dividere dati simili in classi di appartenenza.Tale algoritmo utilizza la prossimità per effettuare la classificazioni. Dopodiché costruisce le varie classi rispettando i neighbors.Questi k-neighbors verranno settati all'inizio, cioè nella fase di creazione del modello predittivo

Nel dettaglio andiamo a spiegare un esempio che modo di implementare un semplicissimo modello predittivo. L'esempio si avvale dei dati del progetto e ha lo scopo di classificare i vari ceppi di vegetazione. Il programma è stato svolto in JupyterLab con linguaggio python.Lo svantaggio del modello è di adoperare una classificazione binaria.

Le fasi di lavorazione sono le seguenti:

- Definire dataframe (media e varianza delle times series).Un dataframe per il train e l altro per la predizione.La varianza e la media sono state calcolate derivanti dalle times series.
- Creazione modello e preparazione dei dati di apprendimento. Definendo Training e test set.
- Predizione del modello. Nell'ultima riga vediamo che fornendo un esempio, il modello lo assegnerà al gruppo 1 (gruppo dell'abete)

```
import pandas as pd
print(pd.__version__)
pwd #Definisco area di lavoro
#Dati rielaborati in matlab derivanti dalle times series
d_train={"ceppo"   : ['abete','leccio'],
        "devianza": [700.8056 , 360.7349],
        "media"   : [ 1.3732  ,  1.1265 ],
        "target"  : [0,1]}
d1 = d_train
```

```
df = pd.DataFrame(data=d1,)
df #dataframe
X = df[df.columns[1:3]] #features
y = df["target"]          #target
X.head()
#libreria sklearn volta a implementare algoritimo KNN,RF etc...
import sklearn
from sklearn.neighbors import KNeighborsClassifier
modello = KNeighborsClassifier(1)
modello.fit(X,y)
modello.predict([[340.7349 , 0.866]])#prova del modello predittivo
```

Il risultato finale sarà:

```
In [14]: modello.predict([[340.7349 , 0.866]])

C:\ProgramData\Anaconda3\lib\site-packages
sClassifier was fitted with feature names
  warnings.warn(
C:\ProgramData\Anaconda3\lib\site-packages
tions (e.g. `skew`, `kurtosis`), the defau
is behavior will change: the default value
e eliminated, and the value None will no l
  mode, _ = stats.mode(_y[neigh_ind, k], a

Out[14]: array([1], dtype=int64)
```

3.2.3 Random Forest

Per parlare di un RF abbiamo bisogno di definire i metodi *ensemble.*[**6**].Solitamente non si usa solo un algoritmo per produrre modelli predittivi ma ne usiamo più di uno, con lo scopo di esaltare le prestazioni complessive. Con metodi ensemble combiniamo tali algoritmi per migliorare la generalità e la robustezza rispetto uno stimatore.Tale stimatore è l'elemento principale per definire una classificazione rispetto un altra.Un modello *ensemble* a singolo stimatore viene chiamato *weaK-learner*, se combina vari stimatori si chiamerà *strong-learner*. Ogni weaklearner sarà addestrato su una parte dei dati di addestramento estratta in maniera casuale dal dataset, tale tecnica è chiamata *bootstrap*. Il *bagging* è una tecnica di ensemble learning in cui si addestrano diversi modelli di machine learning su campioni di dati diversi ottenuti tramite il campionamento bootstrap. Ogni modello viene addestrato su un sottoinsieme dei dati di addestramento generato casualmente con sostituzione. Successivamente, i risultati dei modelli vengono combinati tramite votazione o media per ottenere una predizione finale.La loro combinazione produrrà un modello finale robusto, cioè produce un modello in grado di ridurre l'overfitting e l'underfitting. Il metodo Random Forest è uno dei metodi più diffusi, appartenente proprio alla specifica categoria dei metodi di bagging. Random Forest in R

Programming è un insieme di alberi decisionali [**7**]. Crea e combina più alberi decisionali per ottenere previsioni più accurate. È un algoritmo di classificazione non lineare. Viene effettuata una stima dell'errore nei casi che non viene utilizzata mentre si costruisce l albero. Sono chiamati casuali (random) perché scelgono i predittori in modo casuale al momento dell'addestramento. Sono chiamati foresta (forest) perché prendono l'output di più alberi per prendere una decisione. La foresta casuale supera gli alberi decisionali poiché un gran numero di alberi (modelli) non correlati operano in maniera efficiente rispetto ai singoli modelli costituenti. Vedi anche [**8**].

Per capire e visualizzare un Albero decisionale, abbiamo trovato un software che ci offro la possibilità di andare a vedere le foreste decisionali che si generano, senza scrivere righe di codice. Questo software open source si chiama RapidMinerStudio; invece di programmare in senso classico vengono usati dei blocchi che contengono i modelli già settati e dunque pronti all'uso.

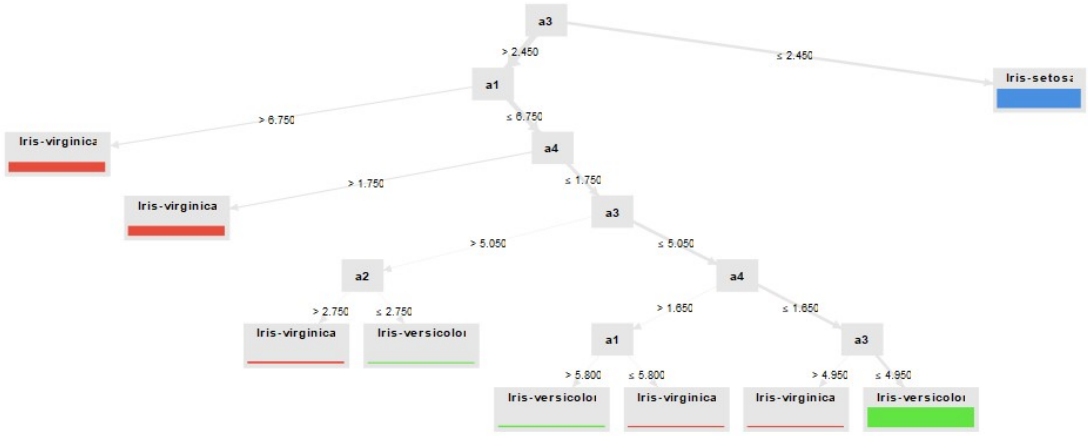


Figura 3.8: Albero decisionale che viene generato per la classifica dei petali del Dataset Iris

Con 3 semplici blocchi possiamo produrre la classificazione con algoritmo Random Forest

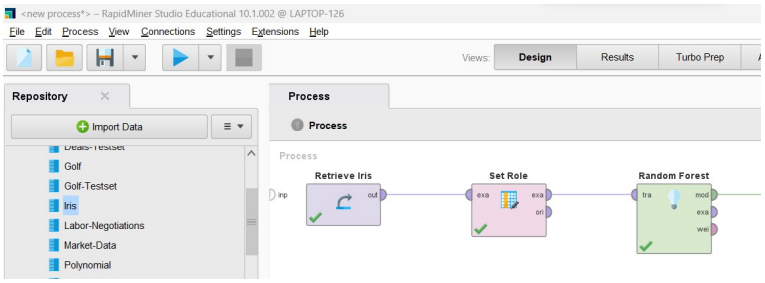


Figura 3.9:

3.3 Supervised learning

La capacità di apprendere dai dati è un aspetto fondamentale soprattutto in tutti quei contesti dove l'uomo assegna ad un insieme dei dati delle classi di appartenenza attraverso una legenda. Questo aspetto richiama gli algoritmi di tipo supervisionato tipici del mondo del machine learning .Applicare un algoritmo supervisionato vuol dire analizzare il dato in ingresso e cercare di estrapolare proprietà e caratteristiche di interesse. Nel nostro progetto vogliamo mappando i pixel definendo cosi la stagionalità della vegetazione (Land Surface Phenology) così da classificare le varie specie vegetali; si utilizzerà un algoritmo supervisionato di per finalità di classificazione visto che utenti esperti hanno definito un data-set da usare sia per la fase di apprendimento che di validazione.

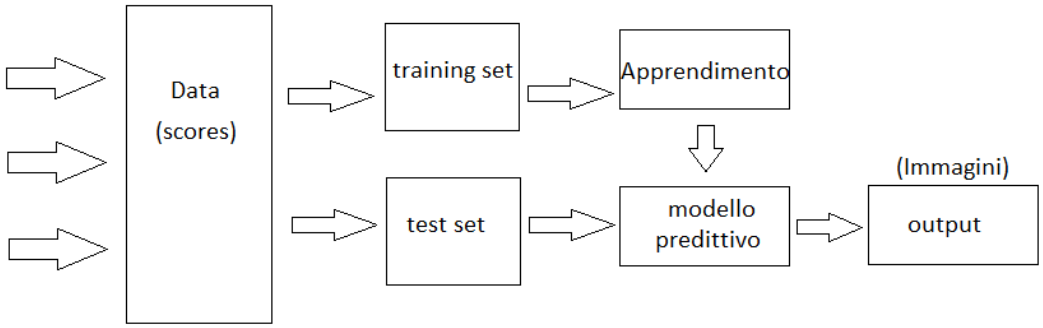


Figura 3.10: Diagramma flusso che spiega come lavora un algoritmo di apprendimento supervisionato.Si nota ce il diagramma adotta un logica ricorsiva.

Un algoritmo di machine learning nelle prima fase separa i dati, di input, in due gruppi: traning-set , utile nella fase di addestramento, e test-set, usato nella fase di test del modello predittivo. Si definisce, solitamente, un 80 % dei dati al training-set e un 20% al test-set. Il diagramma di flusso mostrato in figura 3.10 delinea le vari fasi alla base della creazione di un modello supervisionato. In Figura 3.13 invece si mostra come un modello posso avere diversi comportamento in funzione della qualità dei dati in ingresso e della tipologia di algoritmo utilizzato. In particolare si possono verificare le seguenti situazioni:

- Over-fitting: si riferisce a un modello che lavora troppo bene i dati di addestramento. L'over-fitting è più probabile con i modelli non parametrici e non lineari e non offrono flessibilità nell'apprendimento.
- Under-fitting: si riferisce a un modello che non è in grado di modellare i dati di addestramento né di generalizzare a nuovi dati.L'under-fitting è facile da individuare con una buona metrica delle prestazioni, ad esempio la matrice di confusione.

- Optimal-fit: Idealmente, si vuole selezionare un modello che si trovi al punto giusto tra l'under-fitting e l'over-fitting.Questo è l'obiettivo, ma è molto più difficile da raggiungere nella pratica.

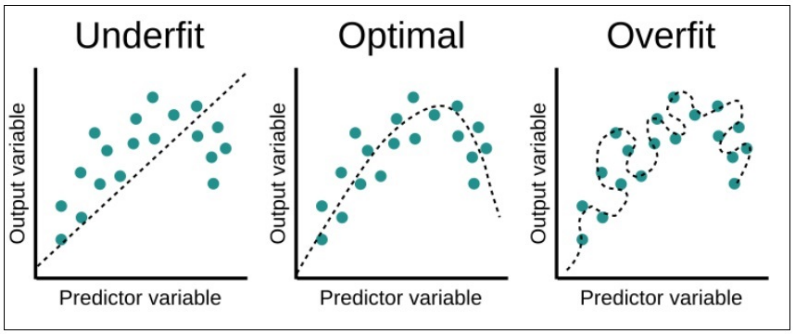


Figura 3.11: https://jashrathod.github.io/2021-09-30-underfitting-overfitting-and-regularization/

Ci sono delle tecniche particolari per ovviare a questi problemi. Un buon approccio per evitare l'over-fitting è quella di utilizzare una parte dei dati (validation set) per valutare la qualità del processo di apprendimento (training). La tecnica del validation set prevede la suddivisione del dataset disponibile in tre parti: training set, validation set e test set. Il modello viene addestrato sul training set e il valore dell'errore viene valutato sia sul validation set che sul test set. Questo aiuta a identificare se il modello è in overfitting sui dati di addestramento. Analizzando il grafico delle curve di apprendimento del training set e del validation set, è possibile quantificare le metriche di qualità del modello e prendere decisioni su eventuali modifiche da apportare. In figura 3.12 possiamo individuare l'andamento temporale del training e della validazione. La situazione migliore si ha quando gli andamenti sono lineari e l'errore è piccolo. Successivamente le due curve divergono. Individuare queste due fasi è fondamentale per capire se modello è robusto.

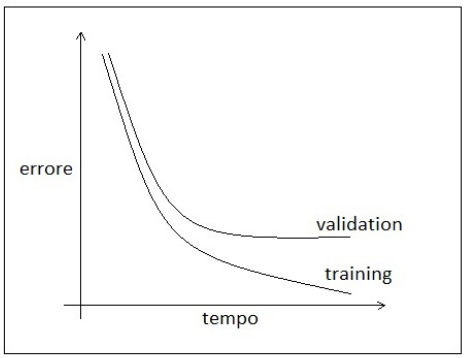


Figura 3.12: Andamento temporale delle metriche durante la fase di training e validazione.

3.4 Cross validation

E' una tecnica per prevenire l'over-fitting [**16**].In questo modello predittivo suddividiamo i dati in K-insiemi. Nella fase di training gli insiemi da 1 a k-1 verranno assegnati al training-set e il k-esimo lo assegniamo al test-set. Nella successiva iterazione assegneremo i valori diversi al training e test. Un esempio di cross validazione viene mostrato della figura. Il modello verrà iterato k volte cosi lavorerà con combinazioni diversa di dati.Questa tecnica verrà applicata al modello predittivo di tipo Random Forest.

Figura 3.13: Esemplificazione della cross-validation:
https://stats.stackexchange.com/questions/290919/what-is-cross-validation-for

Capitolo 4

Elaborazione del Progetto

4.1 Creazione delle times series

4.1.1 Introduzione

Nella prima fase del progetto si cerca di estrapolare le times series, derivanti dalle immagini satellitari. Le times series sono gli andamenti temporali dei pixel rispetto alle 52 settimane che vengono estrapolate dalle immagini. Ogni time series viene poi aggiunta ad un dataframe. Come si vede in figura 4.1 la prima times series corrisponde alla prima riga del dataframe. Si nota che il valore della prima (X1) ed ultima settimana (X52) coincidono per creare una continuità nel tempo.

	1	2	3	4	5	6	7	8	9	10	···	18622	18623	18624	18625	18626	18627	18628	18629	18630	18631
X1	574.4215	340.78534	612.0931	839.4898	813.4939	345.56855	403.7100	629.8092	523.2339	668.1970	···	934.1996	701.7522	728.4072	5385.3165	593.8575	1064.3986	1133.4944	700.9078	1433.4891	1382.3374
X2	473.3139	257.91823	532.4417	763.9165	724.7197	264.38911	346.4919	562.6649	499.2386	687.4736	···	914.9744	729.0427	758.1881	5148.6487	578.0390	1054.8218	1089.4465	739.7769	1261.0409	1287.9546
X3	397.2122	186.01573	462.0782	690.2571	641.7346	192.42144	299.3087	501.5964	470.9228	698.6876	···	886.7335	758.7821	787.4146	4688.4075	580.1456	1021.4816	1039.4802	777.9475	1092.3926	1162.6757
⋮																					
X51	700.0292	432.33031	698.3268	913.9683	904.8583	432.08565	470.0819	700.5915	541.2999	640.6929	···	943.2695	678.6301	699.3855	5357.8421	630.5582	1047.0684	1170.0834	663.3940	1603.4187	1437.4421
X52	574.4215	340.78534	612.0931	839.4898	813.4939	345.56855	403.7100	629.8092	523.2339	668.1970	···	934.1996	701.7522	728.4072	5385.3165	593.8575	1064.3986	1133.4944	700.9078	1433.4891	1382.3374

Figura 4.1: Dataframe contente le times series

Per estrapolare le times serie si devono trasformare le immagini in oggetti *raster*. Gli oggetti raster ci permettono di visualizzare facilmente i valori dei pixel. Ora si cerca di creare un *dataframe* con i dati di interesse che saranno trasformati in curve e ridotti di dimensionalità attraverso l'applicazione dell'analisi FPCA. Lo scopo è di mostrare le times series in base alle bande.

```
#1_LIBRERIE
library(raster),library(sf)
library(rgdal),library(terra)
library(rnaturalearth),library(lubridate)
library(reshape2),library(tictoc)
library(data.table),library(rlang)
#2_UPLOAD DEI DATO DI INPUT
percorso <-'C:/Users/PC/Bande'
percorso_dati <-'C:/Users/PC/Bande/Dati/S2B01'
percorso_shape <- 'C:/Users/PC/Bande/ForesteConero.shp'
setwd(percorso_dati)
file_list <- list.files(percorso_dati,pattern = ".tif")
```

```
file_shp<- readOGR(percorso_shape)
bands <- c('11', '10','09','08','07','06','05','04','03','02','01')
lambda <- c(490,560,665,705,740,783,842,1610,2190)
#3
dati2fit <-data.frame()
n<-length(file_list)
for(i in 1:n){
#3.1_OBJECT RASTER
file<- file_list[i]
rst <- raster(file)#banda 1
lambda <- c(2190) #valore della banda 1
mylayer <- stack(rst)
#3.2_VOGLIO DEFINIRE CORDINATE E VALORI PIXEL
p <- buffer(file_shp, 20)
mylayers.crop <- crop(mylayer, p)
mylayers.mask <- mask(mylayers.crop, p)
mylayers.point <- as.data.frame(rasterToPoints(mylayers.mask))
d <- dim(mylayers.point) #(X,Y,VALUE)
d1 <- d[1]
d2 <- d[2]
#3.3_TRASFORMO NOME FILE IN DOY
nome<-colnames(mylayers.point)[3]
date <- substr(nome, 8, 15)
nomi.date <- ymd(date)
DOY <- as.numeric(strftime(as.Date(nomi.date),"%j"))
doy <-DOY
DOY <- c('x','y',DOY)
colnames(mylayers.point) <- DOY
#3.4_ORDINAMNETO DECRESCENTE DEI DATI
dati <- mylayers.point[,3:d2] #valori pixel
dati <-as.numeric(dati)
dati <-ceiling(dati)
dati <- order(dati)
dati.order <- dati #ordine decrescente
dati.ordered.t <- as.data.frame(t(dati.order))#V1,...,V70429
#3.5_MERGE DEI DUE DATAFRAME
dati.ordered.t$DOY <- as.numeric(doy)
doy <- dati.ordered.t$DOY
m<-matrix(data = NA, nrow = d1+1, ncol = 365, byrow = FALSE)
dati2fit<-as.data.frame(m)
col<- colnames(dati2fit)[doy]
dati2fit[,col]<-t(dati.ordered.t)#dati2fit}
#4_INTERPOLAZIONE CON GAM
tempi <- 1:365
DOY <- dati$DOY
```

```
dt <- data.frame(DOY = seq(1, max(tempi), 1), se = FALSE)
f1 <- function(x) { return(
    predict(gam(as.vector(x) ~ s(DOY, bs = c("cc"), k = -1),
        na.action = na.exclude), dt)) }
dati.tempi.fit <- cbind(tempi, apply(dati[,2:ncol(dati)], MARGIN = 2, f1))
dati.tempi.fit <- dati.tempi.fit[,2:ncol(dati.tempi.fit)]
#5_Setto i parametri interni del dataframe
punti <- ncol(dati.tempi.fit) #numero di time series
bande <- length(seq(490,2130, by = 35))
tempi <- length(seq(4,365, by = 7))
myArray <- array(0, dim=c(punti,bande,tempi))
#6_Vado a riempire l'array vuoto
    for(i in 1:ncol(dati.tempi.fit)){
       myArray[i,,] <- t(as.matrix(dati.tempi.fit[,i]))}
#7_Trasfomrmo l'array in una serie di immagini bidimensionali funzionali
g1 <- new("funData", argvals = list( seq(490,2130, by = 35),1:tempi),
    X = myArray)
sim.m <-multiFunData(g1)
#8_Visualizzo Output
plot(sim.m[[1]], obs = 7000)
#9_Costuiamo MFPCA dell'arrya bidimensionale
pca <- MFPCA(sim.m, M = 35,uniExpansions = list(list(type = "splines2D", k =
#10_Viualizzazione delle componenti principali...
plot(pca$functions[[1]], obs = 1,  main = 'FPC1')
```

In output andremo a visualizzare le Componenti Principali delle times series prodotte dalle immagini satellitari , riferite ai valori delle bande.

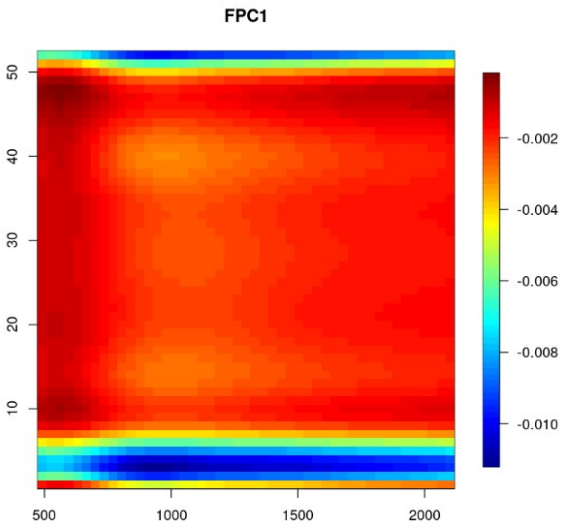


Figura 4.2: Le componenti principali

4.2 Riduzione dimensinalità tramite FPCA

4.2.1 Introduzione

In questo script preleveremo le times series per ridurre la loro dimensionalità e costruire le curve FPCA associate. L'uscita di tale script rappresenta la mappa FPCA.comp.rast della zona di studio.

```
#1
library(fdapace)
library(raster)
#2
ts.file <- './DATA/TS_mcari.txt'
ts <- t(read.delim(ts.file,header = T, sep = " ", quote =
         "\"", dec = "."))
coord.file <- './DATA/coordinate_ts.txt'
pixel.coord <- read.delim(coord.file,header = T,
                 sep = " ",quote = "\"", dec = ".")
#3
tp<- dim(ts)
tp1 <- tp[1]
tp2 <- tp[2]
dati<- as.vector(as.matrix((ts)))
time <- as.numeric(rep(1:tp1, tp2))
ID <- rep(1:tp2, each = tp1)
per.fpca <- as.data.frame(cbind(ID, dati, time))
#4
library(fdapace)
ts.f <- MakeFPCAInputs(per.fpca$ID, per.fpca$time, per.fpca$dati)
fpcaObjTS <- FPCA(ts.f$Ly, ts.f$Lt,list(methodXi='IN',
methodMuCovEst = 'smooth',
userBwCov = 2, kernel= 'gauss',FVEthreshold=0.999))
#5
plot(fpcaObjTS)
CreateScreePlot(fpcaObjTS)
round((fpcaObjTS$lambda/sum(fpcaObjTS$lambda))*100,3)
round(cumsum(fpcaObjTS$lambda/sum(fpcaObjTS$lambda))*100,3)
CreateModeOfVarPlot(fpcaObjTS,k = 1, rainbowPlot = TRUE,
                    col= c( "blue","white","red"))
#6
library(raster)
FPCA.scores <- fpcaObjTS$xiEst
FPCA.scores.coord <- cbind(pixel.coord,FPCA.scores)
FPCA.comp.rast <-rasterFromXYZ(FPCA.scores.coord)
plot(FPCA.comp.rast[[1]])
k=1
```

```
par(mfrow=c(1,2))
CreateModeOfVarPlot(fpcaObjTS,k = k, rainbowPlot = TRUE,
                        col= c( "blue","white","red"))
plot(FPCA.comp.rast[[k]],
    col=colorRampPalette(c("blue", "white", "red"))(255))
#7
crs(FPCA.comp.rast) <- "EPSG:32633"
writeRaster(FPCA.comp.rast, "./DATA/Predictors/tmpc_MCARI_FP)
```

1: si definiscono le librerie da utilizzare. Inizialmente Jupyter Lab non le contiene di default ma dovremmo aggiungerle manualmente.

2: in questa fase definiamo l'input,cioè Le times series (ts). Queste ts sono il risultato finale dell'analisi delle immagini satellitari. Estrapolate nel capitolo "Creazione delle times series".Nella figura si visualizzano i risultati: Si nota che

	1	2	3	4	5	6	7	8	9	10	...	18622	18623	18624	18625	18626	18627	18628	18629	18630	18631
X1	574.4215	340.78534	612.0931	839.4898	813.4939	345.56855	403.7100	629.8092	523.2339	668.1970	...	934.1996	701.7522	728.4072	5385.3165	593.8575	1064.3986	1133.4944	700.9078	1433.4891	1382.3374
X2	473.3139	257.91823	532.4417	763.9165	724.7197	264.38911	346.4919	562.6649	499.2386	687.4736	...	914.9744	729.0427	758.1881	5148.6487	578.0390	1054.8218	1089.4465	739.7769	1261.0409	1287.9546
X3	397.2122	186.01573	462.0782	690.2571	641.7346	192.42144	299.3087	501.5964	470.9228	698.6876	...	886.7335	758.7821	787.4146	4688.4075	580.1456	1021.4816	1039.4802	777.9475	1092.3926	1162.6757
⋮																					
X51	700.0292	432.33031	698.3268	913.9683	904.8583	432.08565	470.0819	700.5915	541.2999	640.6929	...	943.2695	678.6301	699.3855	5357.8421	630.5582	1047.0684	1170.0834	663.3940	1603.4187	1437.4421
X52	574.4215	340.78534	612.0931	839.4898	813.4939	345.56855	403.7100	629.8092	523.2339	668.1970	...	934.1996	701.7522	728.4072	5385.3165	593.8575	1064.3986	1133.4944	700.9078	1433.4891	1382.3374

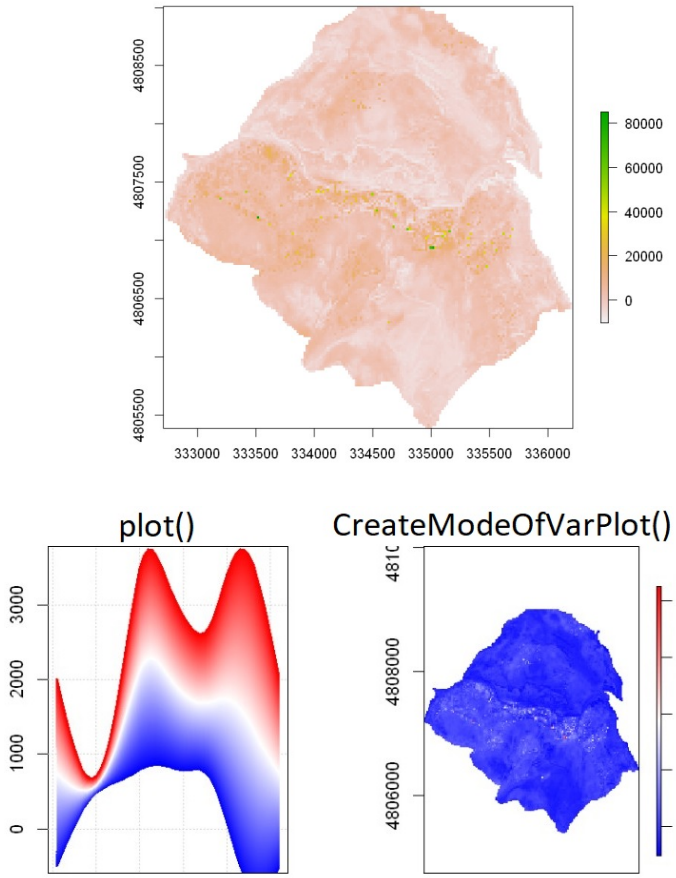
il risultato iniziale t1 è uguale a quello finale tf finale: particolare caratteristica della vegetazione definita come textITLand Surface Phenology.Ciò ci definisce che il valore di riflattanza di una superficie vegetata dipende dalla stagionalità. Oltre a leggere le times serie, definiamo anche le loro corrispettive coordinate (coord.file) poiché stiamo lavorando con immagini "spaziali".

3: si costruisce il dataframe (per.fpca) rispettando le dimensioni opportune. Non sempre si ha il numero esatto degli elementi che costituisce un dataframe;

4: una volta definito il dataframe costruisco le corrispettive curve funzionali delle componenti principali di analisi FPCA. Una volta definite tali curve costruisco un oggetto fpcaOBJTS dove mettere tali curve.

5: si va a visualizzare l'oggetto fpca con plot(fpcaObjTS) e successivamente cerchiamo di arrotondare il dato con funzione round().

6:Ora andiamo ad individuare gli scores relative alle curve FPCA.Dobbiamo anche ricordarci di legare gli scores alle proprie coordinate spaziali (FPCA.score.coord).Ora andiamo a visualizzare plot(FPCA.comp.rast[[1]]): Successivamente andiamo a visualizzare i risultati; plot() e CreateModeOfVarPlot() offrono due modi diversi di visualizzare lo stesso oggetto: **7**: QGIS è stato utilizzato per visualizzare i dati ottenuti in uscita che hanno caratteristiche spaziali QGIS consente agli utenti di definire un sistema di riferimento CRS, Coordinate Reference System, globale o a livello di singolo progetto per i layer privi di un sistema di riferimento. Consente inoltre di definire sistemi di coordinate personalizzati e supporta anche la modifica dei raster. Tutte

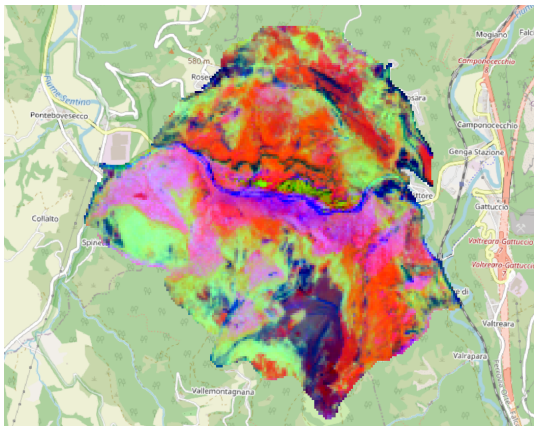


queste funzionalità permettono all'utente di visualizzare layers con sistemi di riferimento diversi e di sovrapporli correttamente. I sistemi di riferimento disponibili in QGIS sono basati su quelli definiti dall'European Petroleum Survey Group (EPSG) e dall'Institut Geographique National francese (IGN) e sono ricavati essenzialmente dalle tabelle di riferimento spaziale usate da GDAL. I codici EPSG sono presenti nel database e possono essere utilizzati per specificare un sistema di riferimento in QGIS.

4.3 Creazione modello predittivo con Random Forest

In questa fase del progetto definiamo un modello predittivo del tipo Random Forest con lo scopo di classificare le specie vegetali. In output restituiamo un immagine geo-referenziata (VegetationMap.tif) che può essere visualizzata in QGIS. Questa immagine contiene tutte le classi vegetali in accordo con la legenda definita dagli esperti all'atto della creazione del training-set.

```
#1_DEFINISCO DATI
set.seed(1000)
RD.files <- './Data/Reference_Data.shp'
```



```
RD <- shapefile(RD.files)
data.folder <- './Data/Predictors'
file.names <- list.files(data.folder, pattern='f_')
file.paths <- paste(data.folder, file.names, sep="/")
predictors <- stack(file.paths)
crs(predictors) <- crs(RD)
#2_DEFINISCO DATAFRAME
pred.val <- extract(predictors,RD, method = 'bilinear')
dataset <- as.data.frame(pred.val)
dataset$CLASS <- RD$Class
#3_DEFINISCI TRANING
set.seed(1000)
ctrl <- trainControl(method = "repeatedcv",number = 10, repeats = 5)
dataset <- dataset[,c(pr,'CLASS')]
mtry <- 1:sqrt(ncol(dataset))
tunegrid <- expand.grid(.mtry=mtry)
rfDownsampled <- train(as.factor(CLASS) ~ ., data = dataset,
                       method = "rf",
                       ntree = 1500,%compromesso-buon modello
                       tuneGrid = tunegrid,
                       metric = "Accuracy",
                       strata = as.factor(dataset$CLASS),
                       trControl = ctrl,
                       sampsize = sample)
#4_PLOT DELLA MAPPA
map <- predict(disaggregate(predictors, c(2,2),
method = 'bilinear'),rfDownsampled)
plot(map)
writeRaster(map, './Data/Results/VegetationMap.tif')
```

Capitolo 5

Conclusioni e sviluppi futuri

Il progetto si è concentrato sull'analisi di immagini satellitari provenienti da Sentinel-2 applicando tecniche di riduzione della dimensionalità dei dati in ingresso (time-series dense). Successivamente è proposto un un modello basato su Random Forest per classificare le varie time-series a partire dai FPCA score che hanno consentito di ridurre la dimensionalità del dato generando in uscita mappe raster che consentono per ogni pixel di associare una determinata classe.

I punti di forza alla base del progetto di tesi sono i seguenti:

- Utilizzo di tecniche di riduzione dei dati PCA (Principal Component Analysis), FPCA (Functional Principal Component Analysis) e MFPCA (Multi-Band Functional Principal Component Analysis),in conformità alle documentazioni ufficiali che spiegano i comandi utilizzati.
- Creazione di mappe fito-sologiche dettagliate che forniscono informazioni sulla vegetazione, riducendo la necessità di rilevamenti sul campo.

Alcune criticità sono emerse durante lo svolgimento del lavoro di tesi ed in particolare:

- si nota come la documentazione circa alcune funzioni relative a pacchetti R sia frammentata ed in alcuni casi incompleta. Tuttavia, R è conosciuto per essere un linguaggio molto adatto per implementare funzioni di analisi dei dati.
- L'approccio di programmazione a "blocchi"/celle di codice che comporta la verifica dei comandi uno alla volta prima di procedere ulteriormente.Il lavoro per trovare il comando giusto richiede tempo.

Complessivamente gli obiettivi iniziali del progetto si ritengo raggiunti con la possibilità di ottimizzare gli script al fine di includere anche differenti indici vegetazionali (es. GNDVI, NDRE,...) che consentiranno di accrescere la precisione ed accuratezza del modello di classificazione.

Bibliografia

[1] Tree Cover Change Mask 2015-2018,https://land.copernicus.eu/pan-european/high-resolution-layers/forests/tree-cover-density/change-maps/tree-cover-change-mask-2015-2018.

[2] Functional Analysis for Habitat Mapping in a Special Area of Conservation Using Sentinel-2 Time-Series Data, https://www.mdpi.com/2072-4292/14/5/1179, Adriano Mancini, Simone Passarini,Giacomo Quartini,Simona Casavecchia.

[3] Recognition and Characterization of Forest Plant Communities through Remote-Sensing NDVI Time Series,https://www.mdpi.com/1424-2818/12/8/313,Adriano Mancini, Simone Passarini, Simona Casavecchia

[4] Analisi della vegetazione naturale in aree vulnerabili alla certificazione mediante telerilevamento,Dott.ssa Claudia Trotta

[5] https://land.copernicus.eu/global/products/ndvi

[6] https://www.geeksforgeeks.org/random-forest-approach-in-r-programming/,2021

[7] Machine Learnng con Python e Scikit-learn,M.di Nunzio ,2022

[8] Data Science con python,Giuseppe Magi,2020

[9] Pca,Analisi Componenti Principali: Algoritmi e Applicazioni,Tesi università di Matematica,Bruno Farabegoli,2015-206

[10] Pca, Analisi delle componenti principali in spazi di Hilbert e applicazioni,Mariantonietta Di Giglio,2013-2014

[11] Fpca,YujianHong,https://fda.readthedocs.io/en/latest/auto_examples/plot_fpca.html

[12] FPCA statistica, https://towardsdatascience.com/functional-principal-component-analysis-and-functional-data-91d21261ab7f.,2021,Johannes Wohlenberg

[13] Mfpca libreria,Multivariate Functional Principal Component Analysis for Data Observed on Different Dimensional Domains,2022

[14] Mfpca statistica,Multivariate Functional Principal Component Analysis for Data Observed on Different (Dimensional) Domains,Clara Happ,2017

[15] https://cran.r-project.org/web/packages/MFPCA/MFPCA.pdf,2022

[16] https://www.fastaireference.com/overfitting

[17] https://en.wikipedia.org/wiki/Cross-validation%28statistics%29

Alla mia famiglia che mi ha sempre supportato e sopportato..