UNIVERSITÁ POLITECNICA DELLE MARCHE

FACOLTÁ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Classificazione di time-series da immagini Sentinel-2 mediante rete InceptionTime nell'ambito della Land Surface Phenology

Time-series classification from Sentinel-2 images using InceptionTime network in the context of Land Surface Phenology

Relatore:
Dott. Mancini Adriano
Correlatore:
Galdelli Alessandro

Laureando:
Compagnoni Marco

Anno Accademico 2020-2021

Indice

1 Introduzione **5**
- 1.1 Obiettivo . . . 5
- 1.2 Progetto . . . 6
- 1.3 Struttura della Tesi . . . 7

2 Metodi e tecniche utilizzate **9**
- 2.1 Telerilevamento . . . 9
  - 2.1.1 Sentinel-2 . . . 9
- 2.2 Classificazione . . . 11
  - 2.2.1 Machine Learning . . . 11
  - 2.2.2 Deep Learning . . . 15
- 2.3 Strumenti software . . . 19
  - 2.3.1 R . . . 19
  - 2.3.2 JupyterLab . . . 19
  - 2.3.3 Putty . . . 20
  - 2.3.4 Matlab . . . 20
  - 2.3.5 Colaboratory . . . 21

3 Creazione del dataset **23**
- 3.1 Pre-processing . . . 23
- 3.2 Creazione Bande e Indici . . . 23
  - 3.2.1 Funzione F03 . . . 24
  - 3.2.2 Funzione F08 . . . 24
- 3.3 Creazione time-series . . . 25
- 3.4 Derivazione time-series . . . 27
- 3.5 Descrizione classi . . . 28

4 Processing del dataset **29**
- 4.1 Random forests . . . 29
  - 4.1.1 Applicazione FPCA univariata . . . 29
  - 4.1.2 Creazione modello . . . 31
- 4.2 InceptionTime . . . 33
  - 4.2.1 Creazione dataset univariate . . . 33

4.2.2 Creazione modello 34

5 Risultati **39**
5.1 Risultati classificazione Random forest 39
5.1.1 F03 39
5.1.2 F08 40
5.2 Risultati classificazione InceptionTime 42
5.2.1 F03 43
5.2.2 F08 46

6 Conclusioni e sviluppi futuri **49**
6.1 Conclusioni 49
6.2 Sviluppi futuri 50

Bibliografia **51**

Elenco delle figure **54**

Elenco delle tabelle **55**

Capitolo 1

Introduzione

La classificazione delle comunità vegetali ricopre un ruolo fondamentale nella definizione delle strategie di monitoraggio e conservazione della biodiversità degli habitat [1]. Attualmente la fitosociologia è il metodo di riferimento per la classificazione vegetale ed essa si basa su dati raccolti direttamente sul campo. Lo sviluppo di piattaforme di rilevazione a distanza multi-spettrali, come i satelliti, ha determinato la nascita di software orientati alla classificazioni di aree geografiche mediante l'utilizzo di dati telerilevati (immagini satellitari). I processi biologici che caratterizzano la vita delle piante alterano la riflettanza della radiazione elettromagnetica incidente dalla superficie terrestre generando cosi una firma spettrale. La variazione nel tempo dei valori di riflettanza prende il nome *Land Surface Phenology*. Combinando tra loro misure della riflettanza di uno o più canali spettrali di uno specifico territorio è possibile ottenere degli indici ambientali [2]. Valutando tali indici in un intervallo temporale, essi forniscono informazioni su quella che è la comunità vegetale presente in quel determinato territorio.

1.1 Obiettivo

Nella fitosociologia classica, il processo di classificazione delle comunità vegetali viene effettuato attraverso uno studio sul luogo. Questo significa che delle persone devono recarsi fisicamente sul luogo da classificare. Questa metodologia risulta essere particolarmente dispendiosa in termini di tempo e anche poco efficace se si considera l'impossibilità di poter valutare ogni punto di un territorio. Lo scopo di questo studio è quello di illustrare e valutare diversi metodi di classificazione, basato sulle immagini telerilevate, che vada ad ovviare le problematiche evidenziate nel metodo classico, ottimizzando così il processo di classificazione. Lo studio riportato in questo elaborato propone un processo di classificazione del territorio relativo al Monte Conero mediante l'utilizzo di algoritmi di apprendimento (Random forest[3]) e rete neurale InceptionTime.

Figura 1.1: Immagine satellitare del Monte Conero.

1.2 Progetto

Nella prima fase di questo studio, partendo dalle immagini satellitari del Monte Conero, si vanno a generare delle strutture dati, chiamate time-series, dell'indice NDVI *(Normalized Difference Vegetation Index)*[4] relativi a 175 luoghi, localizzati all'interno dell'area ZSC (zona speciale di conservazione) del Monte Conero, di cui la classificazione è nota(ground truth). In questo progetto i valori dell'indice NDVI sono stati ottenuti combinando tra loro due o quattro componenti spettrali. I file generati sono caratterizzati da una tabella costituita da 175 colonne, ognuna della quali è associato a uno dei 175 luoghi citati in precedenza. In ogni colonna troveremo i valori di latitudine, longitudine, classe vegetale presente(Class), sotto-classe vegetale presente(Class1) e l'evoluzione temporale dell'indice NDVI valutata su un intervallo temporale di 52 settimane. Nella seconda fase si procede con il calcolo della derivata prima e seconda delle time-series. La terza fase è quella associata alla classificazione. Il classificatore Random forest è stato testato sia sulle time-series normali sia su quelle derivate. Mentre, per la rete neurale InceptionTime il processo di classificazione è stato effettuato esclusivamente sulle time-series normali. Nella figura 1.2 sono illustrati, a livello grafico, i passaggi chiave del progetto descritto in questo elaborato.

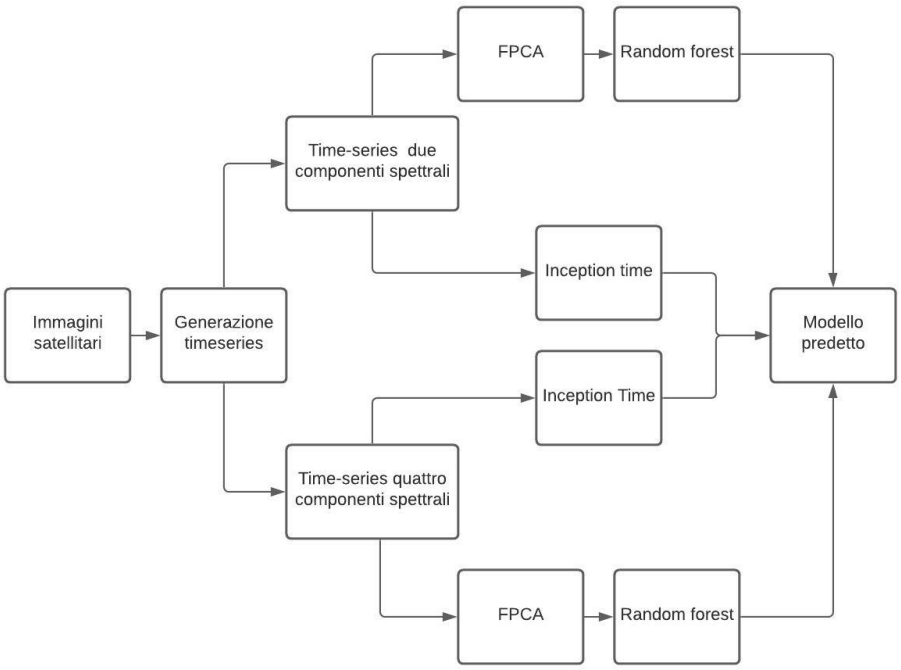

Figura 1.2: Flowchart dei metodi proposti

1.3 Struttura della Tesi

Nel secondo capitolo sono illustrati i metodi e le tecniche utilizzati in questo progetto. In particolare vengono descritti le due tipologie di classificazione proposte e i vari software e linguaggi di programmazione utilizzati. Nel terzo capitolo sono illustrati nel dettaglio i passaggi necessari alla generazione delle time-series e delle loro derivate. Nel quarto capitolo sono descritti i classificatori utilizzati, mettendo in evidenza i parametri che li caratterizzano. Nel quinto capitolo sono illustrati i risultati dei processi di classificazione. Nel sesto capitolo è riportata una valutazione dei risultati ottenuti e delle proposte per eventuali progetti futuri.

Capitolo 2

Metodi e tecniche utilizzate

2.1 Telerilevamento

Il Telerilevamento (*Remote Sensing* in inglese) è un approccio multidisciplinare allo studio del territorio. Esso consente di raccogliere da remoto informazioni per il riconoscimento e la caratterizzazione delle comunità vegetali presenti in un luogo. L'acquisizione dei dati in Telerilevamento avviene grazie a speciali sensori che registrano le informazioni trasportate dall'energia elettromagnetica emessa, riflessa o diffusa dai corpi osservati. Il Telerilevamento sfrutta il diverso modo in cui le superfici naturali interagiscono con l'energia elettromagnetica proveniente da una sorgente per ottenere informazioni sulle loro caratteristiche. Nel suo percorso questa energia interagisce con l'atmosfera e viene modificata in funzione della concentrazione delle sue componenti spettrali prima di raggiungere il bersaglio. L'iterazione tra l'energia incidente e la superfici colpite genera una "traccia" chiamata firma spettrale. Le diverse firme spettrali delle varie superfici naturali vengono captate e misurate da sensori montati su satelliti[5]. In questo studio verranno utilizzate le immagini raccolte dal progetto Copernicus, nello specifico dal sistema satellitare Sentinel-2.

2.1.1 Sentinel-2

Il sistema satellitare Sentinel-2 è costituito da una costellazione di due satelliti: Sentinel-2A e Sentinel-2B. Nella figura 2.1 è illustra la struttura del satellite Sentinel-2A. Questo sistema satellitare fornisce immagini multi-spettrali (13 bande) ad alta e media risoluzione spaziale a seconda della specifica banda. I satelliti pesano circa 1200 kg e sono progettati per una vita utile di circa 7 anni, anche se le batterie e i propellenti sono stati caricati per 12 anni di operazioni, incluse le manovre orbitali di fine vita dello strumento. I sensori MSI (*Multispectral INstrument*) montati sulla piattaforma lavorano passivamente, i raggi di luce incidenti vengono divisi e filtrati su piani focali separati, uno per le bande del visibile (VIS), uno per la banda dell'infrarosso vicino(*NearInfraredNIR*) e uno per l'in-

frarosso "a onde corte" (*Short Wave Infrared SWIR*). I sensori MSI sono in grado di acquisire 4 bande nel visibile e vicino infrarosso con risoluzione spaziale 10 m, 6 bande nell'infrarosso con risoluzione spaziale 20 m e 3 bande con risoluzione 60 m di cui una nel blu e due nell'infrarosso. Una descrizione dettagliata delle bande è illustrata nella figura 2.2. Un apposito meccanismo, una sorta di otturatore, evita che i sensori vengano colpiti dai raggi provenienti direttamente dal sole. Lo stesso meccanismo funziona anche da strumento di calibrazione raccogliendo i raggi del sole dopo la riflessione attraverso un diffusore [6].

Figura 2.1: Immagine sentinel-2A [6].

I due satelliti operano sulla medesima orbita inclinata di 98,62° (rispetto all'equatore), eliosincrona, ad una elevazione media di 786 km, sfasati di 180°. La scelta dell'orbita eliosincrona è stata fatta per minimizzare l'impatto delle ombreggiature sul terreno.

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 - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

Figura 2.2: Bande Sentinel [6].

2.2 Classificazione

2.2.1 Machine Learning

Machine Learning è un sottoinsieme dell'Intelligenza Artificiale che si occupa dello studio e l'ottimizzazione della capacità di un sistema informatico di apprendere informazioni ed effettuare previsioni su di esse utilizzando dei dati in autonomia , cioè senza essere programmati per farlo. Un sistema informatico, analizzando dei dati, può ricavare dei modelli di apprendimento. Questi modelli permettono di costruire algoritmi di apprendimento per effettuare una predizione.

2.2.1.1 Apprendimento supervisionato

L'apprendimento supervisionato è una tecnica che mira a istruire un sistema informatico in modo da consentirgli di elaborare automaticamente previsioni sui valori di uscita rispetto ad un set di dati in input. In questa metodologia il set di dati fornito in input, che prende il nome di *training-set*, è costituito da coppie di valori input-output. Il sistema informatico, utilizzando il set di dati in input, ha l'obbiettivo di riconoscere il legame delle variabili che lo costituiscono, apprenderlo e successivamente utilizzarlo per andare a generare un modello matematico che gli permetta di calcolare l'output corretto in funzione di ogni input. In questa tecnica il supervisore, tipicamente un essere umano, ricopre un ruolo fondamentale in quanto l'affidabilità del modello generato è fortemente dipendente dal training set. Fornendo in ingresso un training-set eccessivamente piccolo, il modello generato potrebbe non avere abbastanza esperienza per fornire un output corretto , mentre fornire in ingresso un training-set troppo grande po-

trebbe rendere il modello generato estremamente complesso. Le prestazioni del modello generato vengono valutate e testate mediante l'utilizzo di un altro set di dati che prende il nome di test-set. Nell'apprendimento supervisionato possiamo effettuare una suddivisione delle tipologie dei modelli di predizione, si parla di:

- **Classificazione** , quando il processo di predizione è effettuato su delle variabili qualitative;
- **Regressione**, quando il processo di predizione è effettuato su delle variabili quantitative.

2.2.1.2 FPCA *(Functional principal component analysis)*

Uno dei principali elementi nello studio dei big data è il **PCA (Principal Component Analysis)**. Il PCA è un metodo con la quale è possibile ridurre il costo computazionale dell'analisi di un grande dataset, riducendo le sue caratteristiche limitando la perdita di informazioni. Un esempio con la quale possiamo illustrare il metodo PCA è il seguente: supponiamo di conoscere la funzione di moto di una particella all'interno di uno spazio di 3 dimensioni (x , y, z). Nel caso in cui la maggior parte delle variazioni di questa particella avvengano principalmente lungo x e y, con variazioni lungo z minime, potremmo pensare di non considerare la componente lungo z trasformando così la legge di moto in una funzione dipende solo da due parametri: x e y. In questo modo è possibile lavorare con una dimensione in meno senza perdere le informazioni più importanti. L'FPCA (Functional PCA) rappresenta un'evoluzione del metodo PCA. Mentre il secondo va a diminuire la dimensione del data-set escludendo quelle caratteristiche poco rilevanti, il primo analizza lo schema generale descritto dal contribuito di tutte le caratteristiche, generando una funzione che le vada a descrivere[7]. Il metodo FPCA può generare tre tipologie di risultati:

- **autovettori** che indicano la variazione associata ad ogni componente;
- il vettore contenente gli **FPCA scores** che rappresentano la somiglianza tra le istanze nel database;
- le **autofunzioni** che rappresentano i principali cambiamenti nei dati.

2.2.1.3 Cross Validation

La cross validation (o validazione incrociata) è una tecnica statistica usata nel machine learning per ottimizzare l'efficacia del set di dati disponibili per il processo di addestramento. La cross validation dispone di diversi metodi, il più diffuso è il k-fold validation. In questo metodo il set di dati iniziale viene suddiviso in k parti di uguale dimensione. Successivamente si seleziona 1 delle k parti disponibili per utilizzarla come test-set. Le restanti k-1 parti vengono utilizzate come training-set[8]. Il processo di addestramento viene ripetuto k volte, in ognuna delle quali si considera come test-set una delle k parti in cui il set di dati iniziale è suddiviso. In questo modo è possibile generare un modello con un livello di accuratezza maggiore rispetto a quello generato mediante un semplice suddivisione del set iniziale in test-set e training-set in quanto si riduce la dipendenza del modello generato dal set di dati utilizzato per la fase di addestramento. In seguito nella figura 2.3 è possibile osservare il come viene suddiviso un determinato dataset:

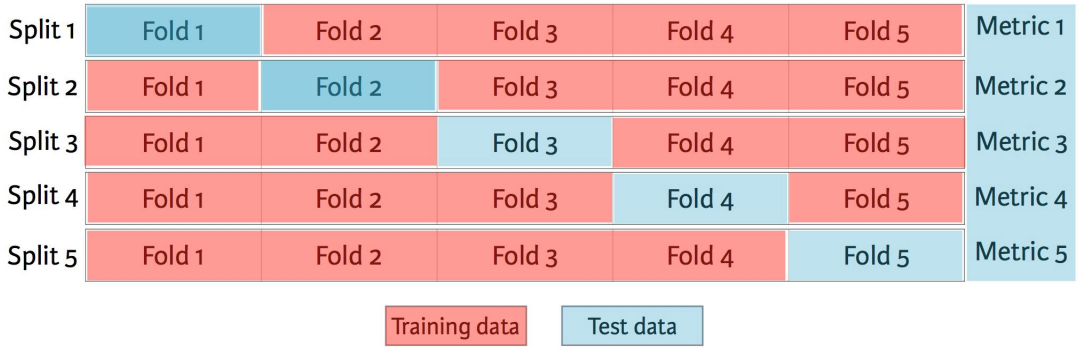

Figura 2.3: Schema suddivisione dataset mediante k-fold [9].

2.2.1.4 Random forest

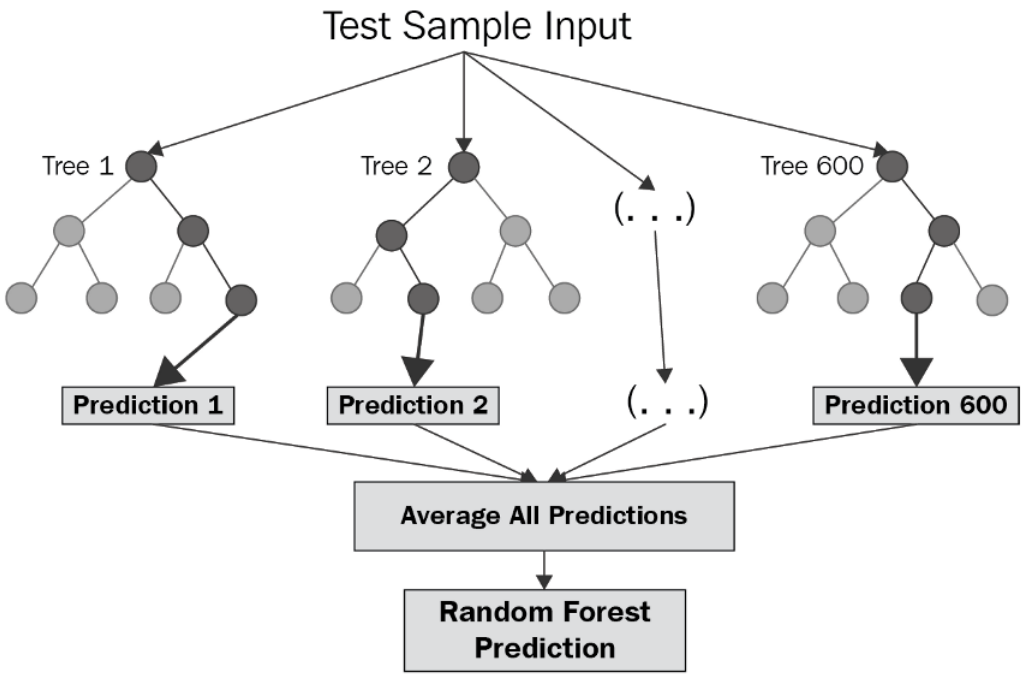

Figura 2.4: Struttura del metodo Random forest [10].

Il Random forest è un metodo per la classificazione o la regressione di big data, che opera costruendo una moltitudine di alberi decisionali durante la fase di addestramento. Il principale obbiettivo di questa tecnica è quello di andare ad ovviare al problema dell'eccessivo adattamento di un modello di predizione al suo set di allenamento(training set). Ciascun albero decisionale viene generato mediante un processo di addestramento basato su campione del training-set casuale, mediante la tecnica di **bagging**[11]. Questa tecnica prevede che ogni campione possiede la medesima dimensione del training-set ma in esso è possibile trovare zero o più copie delle stesso elemento. Questo significa che nel caso in cui il nostro ipotetico training-set sia costituito da dai seguenti elementi (1,2,3,4,5), un ipotetico campione potrebbe avere la seguente forma (1,3,3,4,4). Questa tecnica permette di ridurre la correlazione, in termine di valore o classe predetta, per ogni albero decisionale. Per le attività di classificazione, il valore in output della Random forest è la classe predetta dalla maggior parte degli alberi presenti. Per le attività di regressione , il valore in output della Random forest si ottiene effettuando una media dei valori predetti da ciascun albero[12].

2.2.2 Deep Learning

Il Deep Learning è un una branca del Machine Learning caratterizzata dall'utilizzo, per i processi di classificazione, di reti neurali artificiali. Nonostante la richiesta di capacità computazionali elevate, il punto di forza del Deep Learning è la scalabilità prestazionale, il quale rappresenta anche l'elemento distintivo rispetto al Machine Learning. Infatti aumentando i dati disponibili nei sistemi Deep Learning è possibile migliorare le prestazioni. Questa considerazione non è valida invece per i sistemi del tipo Machine Learning, in quanto, una volta raggiunto un determinato livello di prestazioni non è possibile migliorarle nemmeno aggiungendo ulteriori dati di training[13].

2.2.2.1 Reti Neurali

Le reti neurali si basano sull'idea di poter ricreare il processo di apprendimento presente all'intero dei cervelli biologici. Un cervello biologico utilizza la rete di neuroni per elaborare e per modellare l'ambiente che lo circonda. Nello specifico,

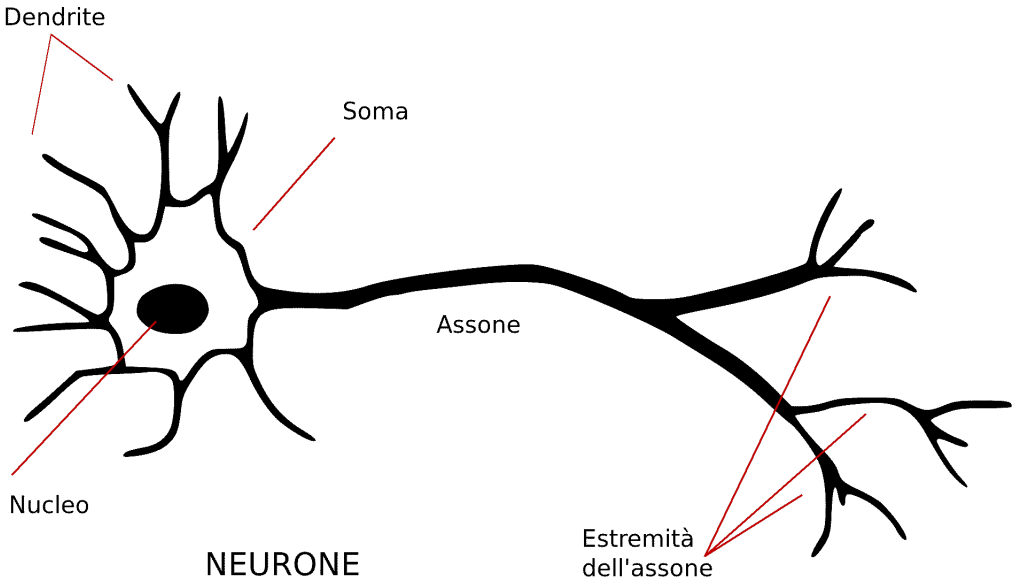

Figura 2.5: Struttura rete neurale biologica [13].

un neurone prende in input le informazioni di altri neuroni attraverso i dentriti, ovvero le fibre minori che si ramificano a partire dal neurone. Successivamente il neurone valuta gli input ricevuti e genera un output che propaga verso gli altri neuroni attraverso l'assone[13]. La struttura delle rete neurali artificiali risulta essere analoga e la possiamo suddividere in:

- **Input:**In questo livello i dati forniti in input alla rete vengono sottoposti ad una prima funzione matematica;

- **Hidden:** In questo livello viene effettuata una combinazione lineare(somma pesate) dei valori in output forniti dall'Input Layer. In una rete neurale possono essere presenti molteplici Hidden Layer;
- **Output:** In questo livello viene svolto lo stesso compito previsto dal Hidden Layer, ma in questo caso l'output coincide con l'uscita dell'intera rete;

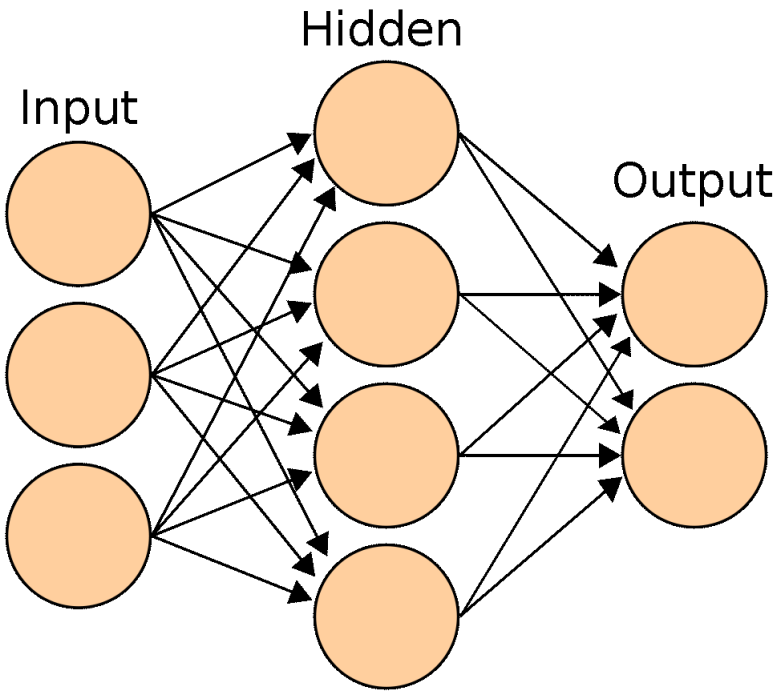

Figura 2.6: Struttura rete neurale artificiale [13].

Il processo di addestramento delle reti neurali è di tipo **"Trials-and-errors"**, cioè le reti ottimizzeranno la loro capacità di classificazione in funzione degli errori commessi. Risulta essere evidente che l'affidabilità dell'output generato da un rete risulta essere dipendente dalla quantità di dati forniti in input.

2.2.2.2 InceptionTime

InceptionTime è una rete neurale, introdotta di recente, caratterizzata da una buona precisione e da un'ottima scalabilità. La sua architettura è simile a quella delle *Convolutional Neural Network* ma si diversifica dalla presenza di blocchi chiamati Inception Modules.

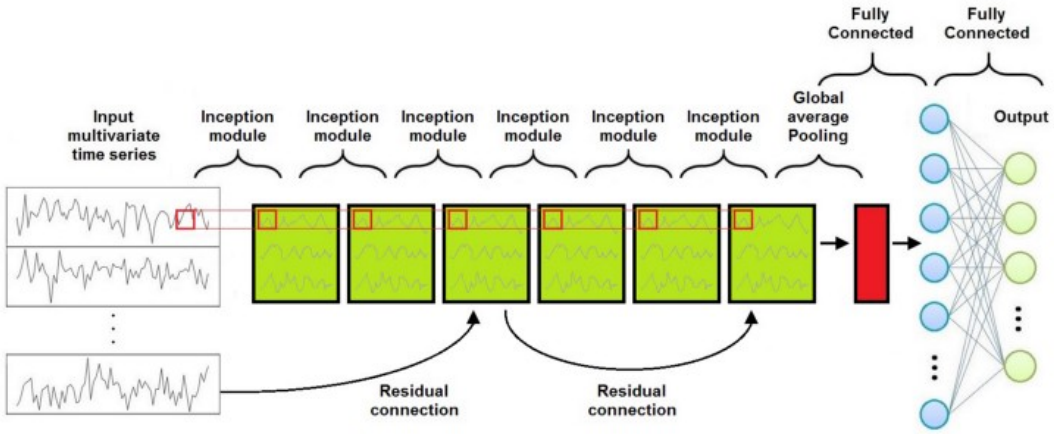

Figura 2.7: Architettura InceptionTime [14].

Ciascun modulo Inception può essere suddiviso in 4 blocchi:

- **1 blocco:**(*Bottleneck*) Questo blocco si occupa di ridurre la dimensione degli input. Lo scopo è quello di ridurre il costo computazionale, velocizzando l'addestramento e migliorando la generalizzazione [14];
- **2 blocco:** Il secondo blocco è costituito da un insieme di processi convoluzionali paralleli che agiscono sulla stessa mappa delle caratteristiche di input. Nella figura 2.8 sono rappresentati 3 processi convoluzionali con dimensione di filtro 10, 20 e 40[14];
- **3 blocco:** Il terzo blocco è un *MaxPooling*, che introduce la possibilità di avere un modello invariante a piccole perturbazioni[14];
- **4 blocco:** Il quarto blocco è un Depth Concatenation Layer, in cui l'output di ciascuna convoluzione parallela indipendente e del MaxPooling viene concatenato per formare la serie temporale di output dell'attuale Inception Module [14].

In questo progetto è stata utilizzata un implementazione della rete InceptionTime presente su Github. Il link con la quale è possibile accedervi è posto in fondo alla pagina. [1] [15].

[1] "https://github.com/hfawaz/InceptionTime"

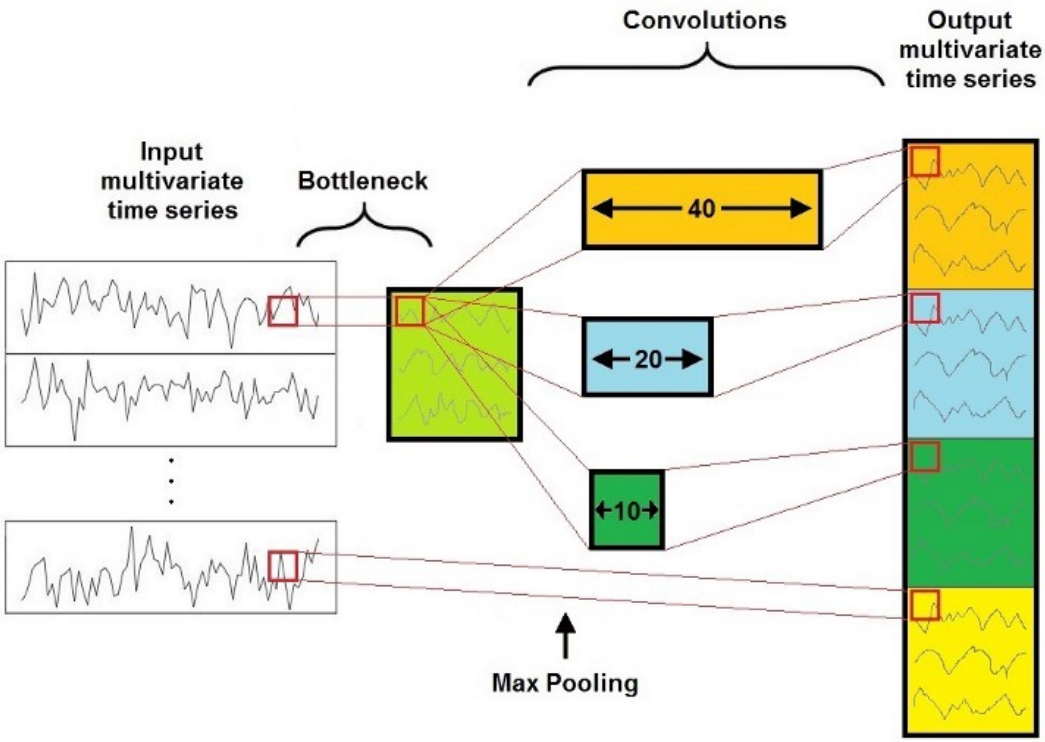

Figura 2.8: Architettura Modulo Inception [14].

2.3 Strumenti software

2.3.1 R

Figura 2.9: Logo di R [16].

R è un linguaggio e un ambiente per il calcolo statistico e la grafica. È un progetto GNU simile al linguaggio e all'ambiente S che è stato sviluppato presso i Bell Laboratories [16]. R fornisce un'ampia varietà di tecniche statistiche (per l'analisi e la classificazione di serie temporali) e grafiche. Il linguaggio S è spesso il veicolo di scelta per la ricerca nella metodologia statistica e R fornisce un percorso Open Source per la partecipazione a tale attività. R è una suite integrata di funzionalità software per la manipolazione dei dati, il calcolo e la visualizzazione grafica. Include:

- una raccolta ampia, coerente e integrata di strumenti intermedi per l'analisi dei dati[16];
- funzionalità grafiche per l'analisi dei dati e la visualizzazione su schermo[16];

Tutti gli script, utilizzati in questo progetto, per la creazione delle time-series, calcolo delle loro derivate , calcolo dei modelli di predizione e la valutazione dei risultati sono stati scritti in R.

2.3.2 JupyterLab

JupyterLab è un'interfaccia utente web-based rilasciata nel Febbraio del 2018. JupyterLab consente di lavorare con documenti e attività che prendono il nome di Jupyter Notebook[17]. Ciascun Notebook può essere suddiviso in celle,permettendo così l'esecuzione di una specifica sezione di un script. In questo progetto, JupyterLab è stato utilizzato per lavorare con gli elaboratori remoti forniti dall'Università Politecnica delle Marche.

Figura 2.10: Logo di JupyterLab [17].

2.3.3 Putty

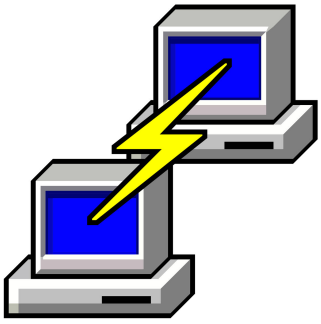

Figura 2.11: Logo di Putty [18].

PuTTY è un client SSH, Telnet e rlogin combinato con un emulatore di terminale per la gestione in remoto di sistemi informatici. È una suite di software libero, originariamente disponibile solo per sistemi Microsoft Windows, Unix e Unix-like[18]. In questo progetto è stato utilizzato per connettersi a JupyterLab attraverso il protocollo SSH.

2.3.4 Matlab

MATLAB (abbreviazione di Matrix Laboratory) è un ambiente per il calcolo numerico e l'analisi statistica scritto in C, che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks. MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi. MATLAB è un ambiente molto utilizzato per via dei suoi numerosi strumenti a supporto dei più disparati campi di studio applicati e funziona su diversi sistemi operativi, tra cui Windows, Mac OS, GNU/Linux e Unix [19]. In questo progetto è stato utilizzato per generare i dataset utilizzato per la classificazione mediante InceptionTime.

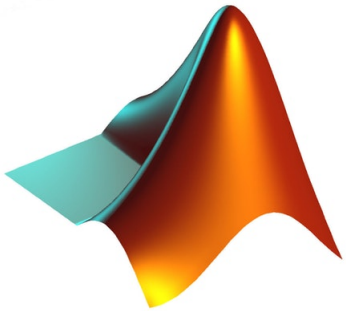

Figura 2.12: Logo di Matlab [19].

2.3.5 Colaboratory

Figura 2.13: Logo di Colaboratory [20].

Colaboratory, o "Colab" in breve , è un prodotto di Google Research. Consente di scrivere ed eseguire codice Python direttamente nel proprio browser con i seguenti vantaggi:

- Nessuna configurazione necessaria sul proprio dispositivo;
- Accesso gratuito alle GPU;
- Condivisione semplificata;

L'ambiente di lavoro generato da Colab è di tipo iterativo e permette di combinare,in un singolo documento, codice eseguibile con immagini,HTML, LaTex e altro ancora. La particolarità di questo ambiente è la sua diretta connessione con Google Drive , infatti è possibile importare dati da un drive per poi memorizzare in esso i risultati delle esecuzione. Colab è particolarmente utilizzato per quanto

riguarda il Data Science ed il Machine Learning. Nell'ambito del Data Science , Colab utilizza la potenza delle librerie Python per analizzare e visualizzare i dati. Nell'ambito del Machine Learning , Colab permette di importare set di immagini , addestrare un classificatore di immagini e valutare le sue performance, il tutto con poche righe di codice. I blocchi note Colab eseguono il codice sui server cloud di Google, il che significa che è possibile utilizzare la potenza dell'hardware Google, tra cui GPU e TPU, a prescindere dalla potenza del proprio dispositivo. In questo progetto le funzionalità di Colab sono state utilizzate per effettuare l'addestramento delle rete neurale InceptionTime per effettuare il processo di classificazione delle immagini satellitari.

Capitolo 3

Creazione del dataset

3.1 Pre-processing

Le immagini satellitari della regione di studio di questo progetto (Monte Conero), prima di essere utilizzate per il processo di classificazione, devono essere sottoposte ad un *pre-processing* iniziale, caratterizzato dalle seguenti fasi:

- **Mascheratura:** Le immagini satellitari acquisite dal satellite(nel periodo di tempo che va dall'Aprile 2017 al Marzo 2020) possono presentare delle imperfezioni dovute alla presenza di nuvole. Quelle zone che sono coperte direttamente dalle nuvole oppure oscurate dall'ombra che esse possono generare non forniscono informazioni veritiere sullo stato della vegetazioni. Queste aree devono essere sottoposte ad un processo di mascheratura con la quale si va a sostituire il colore dei pixel che li compongono con il valore "NO DATA".
- **Ritaglio (Crop) dell'immagine:** Le immagini satellitare rappresentano una area geografica molto più vasta rispetto a quella della regione di studio, quindi devono essere opportunamente ritagliate;
- **Coregistrazione:** Per le fasi successive del progetto è necessario che tutte le immagini siano allineate in modo tale che le coordinate di ciascun pixel di ciascuna immagine corrispondano alla stessa area geografica.

3.2 Creazione Bande e Indici

Come già introdotto nel capitolo 1 le time-series utilizzate in questo progetto vengono generate combinando 2 o 4 componenti spettrali. Ciascuna componete è associata a un determinato intervallo di frequenze(banda). Le bande utilizzate in questo progetto sono state generate nel progetto di tesi proposto dai miei colleghi Forconi Riccardo, Deplano Lorenzo e Colavito Cristian. Le bande in

questione sono le seguenti: "11", "10", "08", "07", "06", "05", "04", "03", "02". Le combinazioni di queste bande(indici), necessarie per la generazione delle time-series, è prodotta da delle funzioni che prendono il nome di F03 e F08.

3.2.1 Funzione F03

La funzione F03 genera degli indici combinando 2 diverse bande tra quelle disponibili. Lo script della funzione F03 è il seguente:

```
bands <- c('11','10','08','07','06','05','04','03','02')
for (A in bands)
 {
  for(B in bands)
   {
    if(B >= A){next}
    combinazioni <- rbind(combinazioni,c(A,B))
   }
 }
```

La variabile "bands" è un vettore contenente tutte le bande disponibili. La variabile "combinazioni" è una matrice, nella quale ciascuna riga rappresenta una combinazione di due bande, cioè un indice. Il suo contenuto è possibile rappresentarlo nel seguente modo:

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

3.2.2 Funzione F08

La funzione F08 genera degli indici combinando 4 diverse bande tra quelle disponibili. Lo script della funzione F08 è il seguente:

```
bands <- c('11','10','08','07','06','05','04','03','02')
for (A in bands)
 {
  for(B in bands)
   {
    for(C in bands)
     {
```

```
  for(D in bands)
   {
        if(B >= A || C >= D || D >= B|| C >= A|| C >= B){next}
    combinazioni <- rbind(combinazioni,c(A,B,C,D))
       }
}
  }
}
```

Anche in questo caso la variabile "combinazioni" è una matrice, nella quale ciascuna sua riga rappresenta una combinazione di quattro bande. Il suo contenuto è possibile rappresentarlo nel seguente modo:

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

3.3 Creazione time-series

In questa fase le informazioni contenute nelle immagini vengono convertiti in dati numerici. Questi dati numerici vengono estrapolati da ciascun immagine attraverso l'utilizzo di un shapefile puntuale, che contiene le coordinate geografiche e spaziali e la classe di appartenenza dei punti sottoposti alla verifica a terra. I dati estrapolati vengono successivamente suddivisi in settimane, facendo una media tra le informazioni contenute in foto che appartengono alla stessa settimana. Per effettuare l'estrazione dai dati viene utilizzata inizialmente la funzione *mask* che ritaglia le immagini in prossimità dei punti specificati dallo shapefile, per poi in seguito applicare la funzione *rasterToPoints* convertendo così il risultato in un dataframe. Tutte le funzioni utilizzate appartengono al package **Raster** di R [16]. I passaggi appena descritti sono svolti dal seguente codice:

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

A questo punto i dati generati vengono ordinati cronologicamente in base ai giorni dell'anno. Essi verranno poi aggregati e per ciascun punto 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. Le righe di codice che svolgono questo compito sono le seguenti:

```
#aggregazione per settimana,secondo la media(mean),i no-data
#vengono scartati (na.rm = true)
dati_aggregati<-aggregate(pointsData,by=pointsData[,"WEEK"]
,FUN = mean,na.rm =TRUE)
```

Infine è necessario interpolare i valori dei pixel del buffer per ogni punto considerato. In questo progetto si è scelto di utilizzare l'interpolazione bilineare. Si usa la funzione *extract* del package Raster. Il codice per eseguire questo compito è il seguente:

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

A questo punto le time-series hanno preso forma, la loro struttura la possiamo osservare nella figura 3.1.

L'ultimo elemento da aggiungere è quello associato alla classe cioè il valore ottenuto attraverso la verifica sul luogo. Le time-series sono state salvate nel formato (.txt) e nominate:

```
FXX_B1_B2_B3_B4_10_BIL.txt
```

	V1	V2	...	V175
X	"388465"	"388530"	...	"387468"
Y	"4822491"	"4822455"	...	"4823601"
CLASS	"L"	"L"	...	"L"
CLASS1	"L3"	"L3"	...	"L3"
X1	"2400.02"	"2645.45"	...	"3803.74"
X2	"2400.02"	"2639.98"	...	"3714.34"
...	...	...	...	...
X52	"2400.02"	"2645.45"	...	"3803.74"

Tabella 3.1: Esempio struttura time-series

3.4 Derivazione time-series

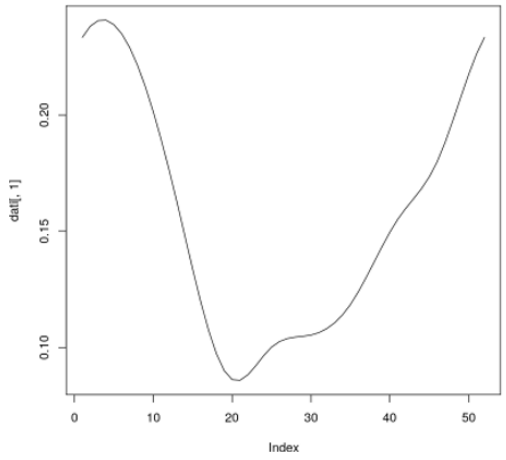

Figura 3.1: Esempio di time-series

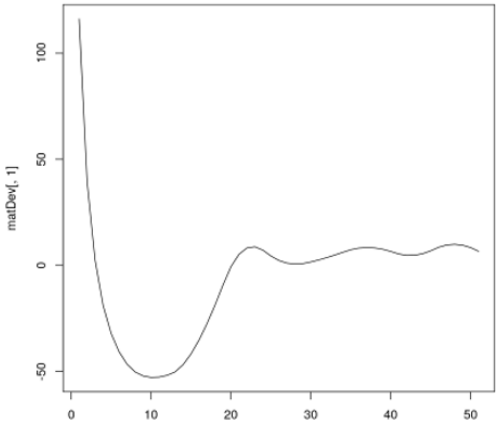

Figura 3.2: Esempio di time-series derivata

La derivazione delle time-series è stata effettuato utilizzando la funzione **grad** del pacchetto R NumDeriv[21]. Il metodo *grad* prevede tre tipologie di metodi derivazione: Simple, Complex e Richardson. In questo progetto il metodo utilizzato è il Richardson in quanto fornisce un grado accuratezza, in termini di calcolo, maggiore. In seguito sono riportate le righe di codice utilizzate per effettuare la derivazione.

```
time_series_der1<-grad(time_series,a, method="Richardson" )
time_series_der2<-grad(time_series_der1,a,method="Richardson")
```

Nelle variabili "time_series_der1" e "time_series_der2" sono memorizzate rispettivamente la derivata prima e seconda della time-series sottoposta al processo derivazione. Effettuando il processo di derivazione, utilizzando il metodo appena descritto, si ha la perdita di una valore per ogni punto per cui si conosce la verità a terra. In pratica per ogni punto, nelle time-series derivate, non avremo più un vettore di 52 indici NDVI ma di 51. Per la derivata seconda tale vettore sarà costituito da 50 elementi. Nelle figure 3.1 e 3.2 è illustrato il confronto tra una time-series normale e la sua rispettiva derivata prima. Le time-series derivate sono state salvate nel formato (.txt) e nominate:

```
F03_B1_B2_00_00_10_BIL-DER1.txt
F03_B1_B2_00_00_10_BIL-DER2.txt
```

3.5 Descrizione classi

Le classi vegetali presenti all'interno della regione di studio (Monte Conero) sono illustrate nella tabella 3.2.

Classi vegetali	**Categoria fisionomica**
L	Boschi a dominanza di Quercus ilex e Quercus rotundifolia
O	Boschi a dominanza di Ostrya carpinifolia
R	Rimboschimenti a dominanza di Pinus sp.

Tabella 3.2: Categorie di classi vegetali

All'interno dei file contenenti le time-series, nello specifico nella riga *Class1*, la classe "L" è stata suddivisa in due sottoclassi e sono denominate nel seguente modo:

- **L3**, per indicare i boschi a dominanza di *Quercus ilex*;
- **L4**, per indicare i boschi a dominanza di *Quercus rotundifolia*.

La classe **"O"** è stata ridefinita come **"O3"**.

Capitolo 4

Processing del dataset

In questo capitolo sono illustrate in dettaglio i due metodi di classificazione utilizzati:

- Random forests
- InceptionTime

4.1 Random forests

Tutti i processi descritti in questa sezione sono stati eseguiti sulle time-series F03 e F08 e poi ripetuti per quanto riguarda le loro time-series derivate(DER1 e DER2).

4.1.1 Applicazione FPCA univariata

Utilizzando il classificatore Random forests, le time-series, prima di essere classificate, devono essere sottoposte ad un processo di FPCA. Lo script a cui facciamo riferimento in questa sezione, nella prima fase, si occupa di estrapolare dalla time-series F03 e F08 quelle informazioni necessarie al processo di FPCA e per i processi successivi. Le righe di codice che svolgono questa mansione sono le seguenti:

```
dati<- as.vector(as.numeric(as.matrix(ts1[5:nrow(ts1), ])))
time <- as.numeric(rep(1:tp1, tp2))#da controllare
ID <- rep(1:tp2, each = tp1)
per.fpca <- as.data.frame(cbind(ID, dati, time))
```

Nello specifico:

- **dati:** in questa variabile vengono memorizzati l'evoluzione dell'indice NDVI nelle 52 settimane;

- **time:**questa variabile conterrà invece la settimana dell'anno all'interno della quale una determinata immagine si trova;
- **ID:** sarà un numero che farà riferimento al pixel all'interno della ground truth;
- **per.fpca:** è un dataset contenente le informazioni di *dati , time e ID*

Il seguente comando si occupa di preparare il dataset ad essere elaborato attraverso il comando FPCA():

```
ts.f<-MakeFPCAInputs(per.fpca$ID,per.fpca$time,per.fpca$dati)
```

Il comando che effettivamente esegue la riduzione del dataset è il seguente:

```
fpcaObjTS <- FPCA(ts.f$Ly, ts.f$Lt,list(methodXi='IN',
methodMuCovEst = 'smooth', userBwCov = 2, kernel= 'gauss'))
```

I parametri forniti al comando FPCA contengono le seguenti informazioni:

- **ts.fLy:** contiene i valori osservati per ogni punto relativo alle settimane;
- **ts.fLt:** contiene le settimane alla quale si riferiscono i valori di ts.fLy;
- **methodXi='IN':**si riferisce il metodo per stimare gli scores, in questo caso è stato inserito IN, cioè relativo a una serie di dati densi;

```
scores <- fpcaObjTS$xiEst
```

Attraverso questo metodo si vanno ad estrarre gli FPCA score generati dal comando FPCA() e vengono inseriti all'interno della variabile scores. All'interno delle variabile scores andiamo ad aggiungere una colonna il cui contenuto è la classe vegetale , ottenuta mediante ground truth. Tutti gli FPCA scores vengo successivamente memorizzati in un dataset, la cui struttura è analoga a quella mostrata nella tabella 4.1. Questo processo ovviamente viene ripetuto per ciascun indice previsto dalla funzione F00.

	X1	**X2**	**X3**	**...**	**X11**
V1	"-694.87"	"155.68"	"-262.21"	...	"L3"
V2	"-538.76"	"101.21"	"-555.56"	...	"L3"
...	...	...	...	...	...
V175	"-595.91"	"-703.53"	"639.25"	...	"L3"

Tabella 4.1: Esempio struttura time-series FPCA

4.1.2 Creazione modello

Una volta ottenuti i dataset FPCA è possibile intraprendere il processo di classificazione. Qui in seguito saranno descritti i passaggi chiavi dello script generato per la creazione dei modelli.

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

Con il metodo trainControl è possibile andare a definire i parametri per il processo di *training* del modello. In questo caso il metodo di addestramento utilizzato è il **repeatedcv**. Questo metodo consiste nel ripetere molteplici volte la classica cross-validation migliorando cosi l'accuracy del modello generato. I parametri forniti al metodo trainControl sono:

- **method:** questo parametro definisce il metodo utilizzato per definire in dati che saranno utilizzati per l'addestramento;
- **number:** numero di cartelle su cui effettuare la cross-validation;
- **repeats:** è un parametro che viene specificato solo nel caso si utilizzi il metodo repeatedcv, indica il numero di ripetizione dell cross-validation;

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

TuneGrid2 indica la tuning grid utilizzata per creare il modello. Definendo una tuning grid personalizzata è possibile andare a valutare il comportamento del Random forest in funzione della variazione dei suoi hyperparametri. Gli hyperparametri del modello sono le proprietà che governano l'intero processo di addestramento [22]. In questo progetto l'accuracy dei modelli generati sono stati valutati solo in funzione della variazione di un singolo hyperparametro : **mtry**.

Il metodo con la quale si effettua l'addestramento del modello è il seguente:

```
rfDownsampled <- train(Class1 ~ ., data = dataset_FPCA,
                       method = "rf",
                       ntree = 1500,
                       tuneGrid = tuneGrid2,
                       tuneLength = sqrt(ncol(dataset_FPCA)),
                       metric = "Accuracy",
                       strata = dataset_FPCA$Class,
                       trControl = ctrl,
                       sampsize = campione)
```

I parametri della funzione **train** sono i seguenti:

- **Class :** definisco la caratteristica da comparare con tutte le altre. In pratica specifico che lo scopo è quello di effettuare la predizione della classe vegetativa;
- **data:** indico il dataset da considerare;
- **method:** definisco il classificatore da utilizzare;
- **ntree:** definisco il numero di alberi da creare per effettuare la predizione per ogni istanza;
- **tuneGrid:**definisco la tune-grid da utilizzare;
- **metric**:indica qual è la caratteristica principale del modello, in questo caso l'accuratezza. Il modello finale salvato sarà quindi riferimento quello in cui è stata riscontrata la miglior accuratezza;
- **trControl:**indica il train control da utilizzare.

I risultati del processo di classificazione sono illustrati dalla figura 4.1. In essi è possibile osservare le performance dei modelli generati in funzione della variazione del parametro mtry.

```
"mtry" "Accuracy" "Kappa" "AccuracySD" "KappaSD"
"1" 1 0.418042866761683 0.112090861602301 0.0562666941090522 0.0826447719861933
"2" 2 0.419232947457285 0.117573746015509 0.0579278132032558 0.0855308134863378
"3" 3 0.424305594006446 0.126944589969528 0.0594601762569933 0.0857738131286831
```

Figura 4.1: Esempio tabella performance RF.

Un altro elemento da considerare nella fase delle valutazione delle performance è la matrice di confusione, illustrata nella figura 4.2. La matrice di confusione è costituita da 4 tipologie di valori:

- **true positive:** la classe che è stata predetta è vera ed la predizione è giusta;
- **true negative:** la classe predetta è falsa e la predizione è giusta;
- **false positive** la classe predetta è vera e la predizione è errata.
- **false negative** la classe predetta è falsa e la predizione è errata;

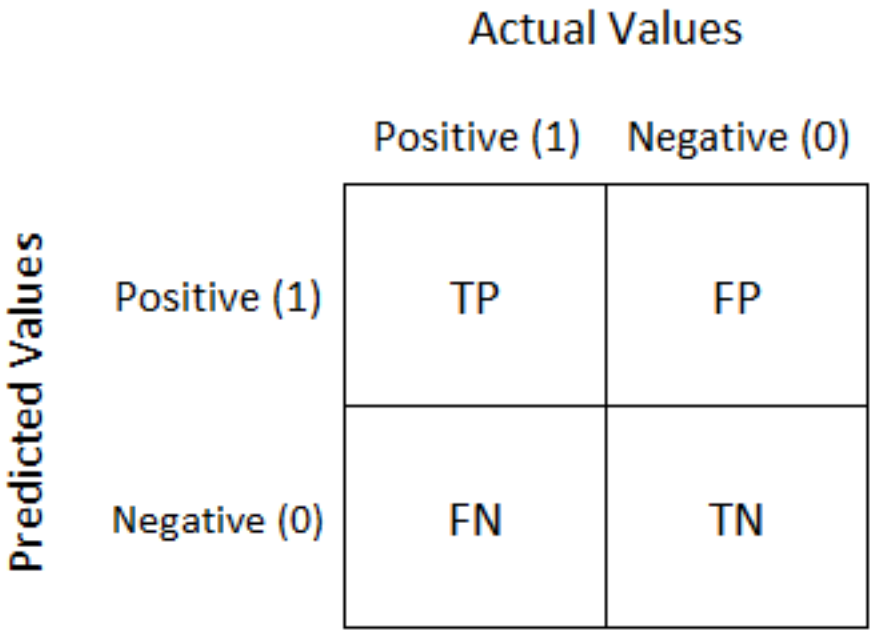

Figura 4.2: Esempio matrice di confusione.

4.2 InceptionTime

4.2.1 Creazione dataset univariate

La rete neurale InceptionTime è stata testata sulle time-series generate dalle funzione F03 e F08. La loro struttura non è immediatamente utilizzabile per il processo di classificazione. Attraverso l'utilizzo di Matlab, la loro struttura è stata modificata per renderla idonea al processo di classificazione, essa è rappresentata nella tabella 4.2. Nei dataset relativi all' Inceptiontime e nei sui risultati le classi sono state indicate attraverso l'utilizzo di semplici numeri: 1, 2, 3 e 4.

	X1	X2	...	Classe
V1	0.210867525935422	0.903981486611507	...	1
V2	2.995751845610290	3.211795413625440	...	1
...	...	...	...	...
V175	-0.464593337289954	-0.208296736950631	...	3

Tabella 4.2: Esempio struttura time-series InceptionTime

La nuova struttura si ottiene effettuando un processo di trasposizione della matrice degli indici NDVI, presente nelle time-series classiche, ottenendo cosi una matrice di dimensione 52×175. Successivamente si aggiunge una colonna in

cui sono riportati i valori della classi ottenute mediante ground truth. I nuovi dataset devono essere successivamente suddivisi in train-set e test-set. Lo script che esegue questa suddivisone è il seguente:

```
c = cvpartition(B,'Holdout',0.2);
istrain=training(c);
istest=test(c);
train = X(istrain,:);
test = X(istest,:);
```

Il metodo utilizzato per effettuare la suddivisione è **Holdout**, che prevede nell'utilizzare l'80% del dataset disponibile per effettuare l'addestramento e il restante 20% per la fase di test.

4.2.2 Creazione modello

In questo progetto è stata utilizzata un'implementazione della rete InceptionTime presente su Github[15]. Questa implementazione è scritta in Python e utilizza la libreria open source Keras. Il codice è suddiviso in questo modo:

- il file ***main.py*** , che contiene il codice necessario per eseguire un esperimento di classificazione;
- la cartella ***utils*** contenente le funzioni per la lettura dei dataset e visualizzare i grafici. Per la visualizzazione dei dati utilizza la libreria **numpy**;
- la cartella ***classifiers*** che contiene due file python:
  - **inception.py** che contiene la rete InceptionTime;
  - **nne.py** che contiene le informazioni per raggruppare un gruppo di reti Inception.

Il codice utilizzato per avviare la rete prende il nome di start-file.ipynb. Il primo compito di questo codice è quello di andare a caricare su Colab i dataset, da classificare, opportunamente caricati in un Google Drive. Questo compito viene svolto dalle righe di codice sotto riportate:

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
%cp -av /content/drive/MyDrive/InceptionTime
/content/InceptionTime
```

Successivamente si procede al download e all'istallazione delle librerie necessarie:

```
%tensorflow_version 1.x
!nvidia-smi
!pip install h5py==2.8.0
```

Arrivati a questo punto è possibile avviare il processo di classificazione. La riga di codice con la quale è possibile avviare il main della rete Inception è la seguente:

```
!python3 main.py InceptionTime
```

Durante l'esecuzione del main, si generano 5 reti Inception diverse. Queste 5 reti si diversificano in quanto presentano differenze nei valori dei loro iperparametri, nello specificano possiedono *pesi* diversi. I pesi sono generalmente inizializzati con piccoli numeri casuali per prevenire la morte dei neuroni, ma non troppo piccoli per evitare il gradiente zero [14]. Ciascuna rete quindi esegue un processo di classificazione diverso fornendo così diversi risultati. Lo scopo di questa suddivisione della rete è quello di trovare quella configurazione dei pesi che ottimizzi il processo di classificazione.
I risultati dal processo di classificazione possono essere valutati attraverso l'utilizzo di tre diversi strumenti:

- *Metriche*, la cui struttura è descritta nella tabella 4.3;
- *tabella delle performance*, la cui struttura è descritta dall'immagine 4.3 ;
- *Matrici di confusione*;

precision	accuracy	recall	duration
...	...	...	...

Tabella 4.3: Esempio struttura risultati InceptionTime

	precision	recall	f1-score	support
0	0.80	0.86	0.83	14
1	0.70	0.88	0.78	8
2	1.00	0.67	0.80	3
3	1.00	0.80	0.89	10
accuracy			0.83	35
macro avg	0.88	0.80	0.82	35
weighted avg	0.85	0.83	0.83	35

Figura 4.3: Esempio struttura tabella delle performance

I parametri che vengono riportati nelle metriche e nelle tabelle delle performance sono i seguenti:

- **Accuracy:** questo parametro indica il numero di istanze correttamente predette rispetto al numero totale di istanze di dati. In seguito viene illustrata la formula per calcolare l'accuracy;

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{4.1}$$

  Questo parametro potrebbe non essere una buona metrica nel caso in cui il set di dati risulti non essere bilanciato in quanto potrebbe fornire una descrizione fuorviante dei risultati;

- **Precision:** è definito come il rapporto tra i true positive e la somma dei true positive e false positive. Osservando la formula possiamo dedurre che la precision assume valore 1 solo quando il valore del numeratore è pari a quello del denominatore cioè nel momento in cui i Falsi Positivi(FP) risultano essere nulli;

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

- **Recall:**ovvero la capacità di individuare i campioni positivi tra tutti i di campioni positivi del dataset.E' definita com il rapporto tra le istanze positive correttamente individuate dal sistema e la somma dei true positive e false negative ;

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

- **F1 score:** è la media armonica di precisione e richiamo ed è una misura migliore dell'accuratezza .

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4.4}$$

Il processo di classificazione con la rete InceptionTime proposto in questo elaborato prevede due diverse fasi. In prima istanza, la classificazione è stata eseguita considerando le 3 classi riportate all'interno delle time-series nella riga *Class*. Successivamente, per quei indici che hanno ottenuto i risultati migliori, il processo di classificazione è stato ripetuto considerando le 4 classi riportate all'interno delle time-series alla riga *Class1*.

Capitolo 5

Risultati

5.1 Risultati classificazione Random forest

Partendo dalle tabelle delle performance, la cui struttura è descritta dalla figura 4.1, sono stati generati dei file Excel per andare a valutare quali indici restituiscono valori di performance migliori e sopratutto in quale configurazione (normale, derivata prima o derivata seconda). La struttura del file Excel è illustrata nella tabella 5.1. Nel file Excel sono stati riportati esclusivamente i valori di performance migliori ed è questa la motivazione per la quale nella colonna "mtry" troveremo valori diversi.

Indici	mtry	Accuracy	Kappa	AccuracySD	KappaSD	L3	L4	O3	R
11-10	3	*	*	*	*	*	*	*	*
11-10-DER1	2	*	*	*	*	*	*	*	*
11-10-DER2	1	*	*	*	*	*	*	*	*
...	...	...	...	...	...	...	...	...	...

Tabella 5.1: Struttura file Excel per valutare la performance Random forest

5.1.1 F03

Le performance migliori ottenute utilizzando le time-series F03 sono descritte nella tabella 5.2. In essa è possibile osservare che la prestazione migliore è stata ottenuta con la time-series di 07-05, con un valore di accuracy del 80%. Valutando invece i risultati in termini di configurazione, possiamo affermare che solo per alcuni indici i risultati ottenuti con le time-series derivate risultano essere migliori rispetto a quelli ottenuti con le time-series normali.

Indice migliore	Indice peggiore	Migliori 10 indici	Valori di accuracy
07-05	11-02-Der2	07-05	0.8040
*	*	08-03	0.7997
*	*	06-05	0.7995
*	*	10-06	0.7994
*	*	05-04-Der1	0.7979
*	*	11-10	0.7948
*	*	11-06	0.7938
*	*	11-08	0.7924
*	*	08-04	0.7829
*	*	05-04	0.7810

Tabella 5.2: Tabella performance F03 con Random forest

Nelle figure 5.1 e 5.2 sono riportate le performance e la matrice di confusione associata alla time-series F03-07-05.

```
"mtry" "Accuracy" "Kappa" "AccuracySD" "KappaSD"
"1" 1 0.794378787412717 0.693078628940237 0.0392508028005546 0.0572074929095586
"2" 2 0.802924215858173 0.706462191891795 0.032634902493917 0.0469337719131787
"3" 3 0.804044076683038 0.709297018242902 0.0357216774622241 0.0514761756004865
```

Figura 5.1: Performance F03-07-05 RF

```
"L3" "L4" "O3" "R" "class.error"
"L3" 47 16 8 0 0.338028169014085
"L4" 6 30 0 1 0.189189189189189
"O3" 2 0 11 0 0.153846153846154
"R" 0 5 2 47 0.12962962962963
```

Figura 5.2: Matrice di confusione F03-07-05 RF

5.1.2 F08

Le performance migliori ottenute utilizzando le time-series F08 sono descritte nella tabella 5.3. In essa è possibile osservare che la prestazione migliore è stata ottenuta con la time-series di 10-05-03-04, con un valore di accuracy del 78%. Anche in questo caso valutando invece i risultati in termini di configurazione, possiamo affermare che solo per alcuni indici i risultati ottenuti con le time-series derivate risultano essere migliori rispetto a quelli ottenuti con le time-series normali.

Indice migliore	Indice peggiore	Migliori 10 indici	Valori di accuracy
10-05-03-04	08-07-02-04-Der1	10-05-03-04	0.7794
*	*	10-07-03-06	0.7706
*	*	10-07-04-06	0.7681
*	*	06-05-03-04	0.7646
*	*	07-05-03-04	0.7645
*	*	11-10-06-07	0.7629
*	*	08-05-03-04	0.7588
*	*	10-05-03-04-Der1	0.7577
*	*	11-07-03-06-Der1	0.7577
*	*	11-10-03-08	0.7576

Tabella 5.3: Tabella performance F08 con Random forest

Nelle figure 5.3 e 5.4 sono riportate le performance e la matrice di confusione associata alla time-series F08-10-05-03-04.

```
"mtry" "Accuracy" "Kappa" "AccuracySD" "KappaSD"
"1" 1 0.779407975910944 0.663659392842245 0.0443920428784896 0.0694532713142252
"2" 2 0.773639983457572 0.656617654051626 0.0476689666200343 0.0737356581434322
"3" 3 0.773092494044384 0.657346504246645 0.0454159954094507 0.0697191980984218
```

Figura 5.3: Performance F08-10-05-03-04 RF

```
"L3" "L4" "O3" "R" "class.error"
"L3" 45 11 13 2 0.366197183098592
"L4" 7 28 0 2 0.243243243243243
"O3" 3 0 10 0 0.230769230769231
"R" 5 4 0 45 0.166666666666667
```

Figura 5.4: Matrice di confusione F08-10-05-03-04 RF

5.2 Risultati classificazione InceptionTime

L'analisi dei risultati è stato effettuata sia in termini di valore assoluto, sia in termini di valore medio(robustezza). Come descritto nella sezione 4.2.2, il processo di classificazione mediante InceptionTime restituisce 5 diverse tabelle di performance associate alle 5 reti che si vanno a generare durante il processo di classificazione per ogni indice. I valori medi di accuracy, che sono illustrati in seguito, si ottengono effettuando una media dei 5 valori di accuracy restituiti dalla rete. I valori assoluti si riferiscono al valore massimo di accuracy generato dalle 5 reti.
I valori di performance sono stati riportati in un file Excel, in cui per ogni indice sono stati riportate le informazioni illustrate nella tabella 5.4.

Indice:	F03-11-10	F03-11-08
Itr0	0.5429	0.5714
Itr1	0.6571	0.4
Itr2	0.4857	0.5429
Itr3	0.5143	0.5714
Itr4	0.4286	0.6571
Valore massimo	0.6571	0.6571
Valore medio	0.5257	0.5486

Tabella 5.4: Tabella descrittiva per la performance di un indice

In seguito utilizzando le funzioni di Excel abbiamo ricavato una tabella in cui sono riportate:

- indice migliore;
- indice peggiore;
- migliori 10 indici;

5.2.1 F03

5.2.1.1 Risultati classificazione 3 classi

La tabella che descrive le prestazione dei modelli generati in termini di valore assoluto è la tabella 5.5:

Indice migliore	Indice peggiore	Migliori 10 indici	Valori di accuracy
F03-07-04	F03-11-04	F03-07-04	0,8286
*	*	F03-06-03	0,8286
*	*	F03-10-08	0,8
*	*	F03-10-07	0,8
*	*	F03-11-06	0,7714
*	*	F03-11-05	0,7714
*	*	F03-08-05	0,7714
*	*	F03-08-04	0,7714
*	*	F03-11-07	0,7429
*	*	F03-10-03	0,7429

Tabella 5.5: Tabella performance indici in termini di valore assoluto F03

La tabella che descrive le prestazione dei modelli generati in termini di valore medio è la tabella 5.6.

Indice migliore	Indice peggiore	Migliori 10 indici	Valori di accuracy
F03-07-04	F03-11-04	F03-07-04	0,7886
*	*	F03-10-07	0,7771
*	*	F03-06-03	0,7771
*	*	F03-10-08	0,7543
*	*	F03-11-05	0,7314
*	*	F03-08-05	0,7257
*	*	F03-08-04	0,7257
*	*	F03-06-02	0,7257
*	*	F03-11-06	0,72
*	*	F03-11-07	0,7086

Tabella 5.6: Tabella performance indici in termini di valore medio F03

In questo caso l'indice F03-07-04 risulta essere il migliore, sia in termini di valore assoluto sia in termini di valore medio, con un accuracy massima del 83%. Nelle figure 5.5 e 5.6 è illustrata la matrice di confusione e la tabella delle performance dell'iterazione migliore per F03-07-04.

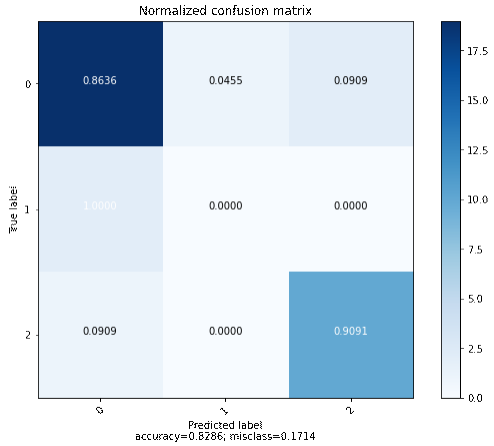

Figura 5.5: Matrice di confusione della F03-07-04(3 classi)

	precision	recall	f1-score	support
0	0.86	0.86	0.86	22
1	0.00	0.00	0.00	2
2	0.83	0.91	0.87	11
accuracy			0.83	35
macro avg	0.57	0.59	0.58	35
weighted avg	0.80	0.83	0.82	35

Figura 5.6: Performance della F03-07-04(3 classi)

Per completezza andiamo ad inserire anche i risultati della seconda combinazione migliore,in termini di robustezza, la F03-10-07. Anche in questo il valori di accuracy raggiunto è del'83% ma si diversifica dalla precedente in quanto leggermente peggiore in termini di valore medio. Nelle figure 5.5 e 5.8 è illustrata la matrice di confusione e la tabella delle performance della migliore iterazione per F03-10-07.

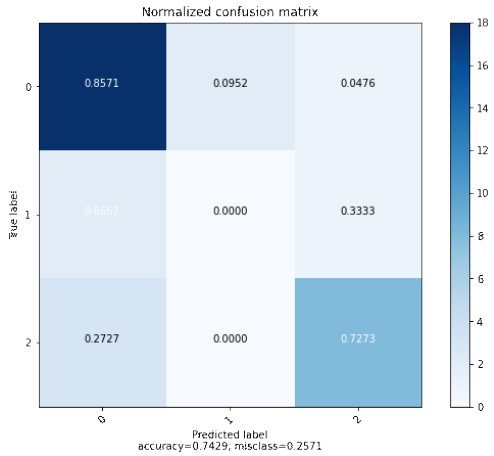

Figura 5.7: Matrice di confusione della F03-10-07(3 classi)

	precision	recall	f1-score	support
0	0.86	0.86	0.86	22
1	0.00	0.00	0.00	2
2	0.83	0.91	0.87	11
accuracy			0.83	35
macro avg	0.57	0.59	0.58	35
weighted avg	0.80	0.83	0.82	35

Figura 5.8: Performance della F03-10-07(3 classi)

5.2.1.2 Risultati classificazione 4 classi

Estendo il processo di classificazione da 3 a 4 classi si ottengono i seguenti risultati a parità di indice. Nelle figure 5.9 e 5.10 sono illustrate la matrice di confusione e la tabella delle performance della migliore iterazione di F03-07-04. In questo caso è stato raggiunto un livello di accuracy del 89%.

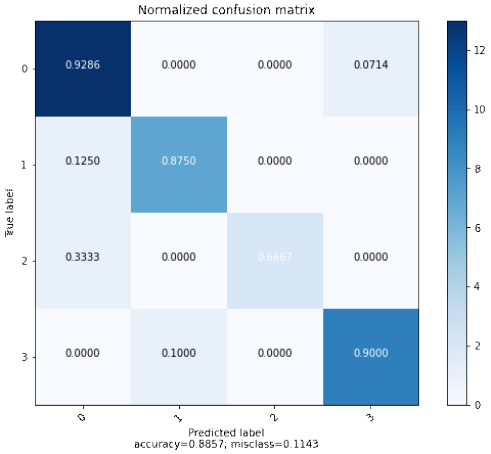

Figura 5.9: Matrice di confusione della F03-07-04

	precision	recall	f1-score	support
0	0.87	0.93	0.90	14
1	0.88	0.88	0.88	8
2	1.00	0.67	0.80	3
3	0.90	0.90	0.90	10
accuracy			0.89	35
macro avg	0.91	0.84	0.87	35
weighted avg	0.89	0.89	0.88	35

Figura 5.10: Performance della F03-07-04

Nella figure 5.12 e 5.11 sono illustrate la matrice di confusione e la tabella delle performance dell'iterazione migliore per F03-10-07. In questo caso è stato raggiunto un livello di accuracy pari al 86%.

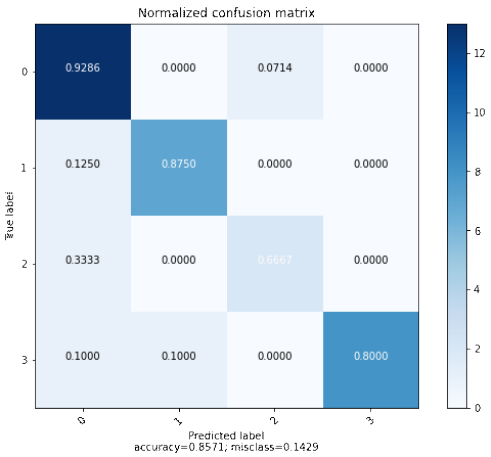

Figura 5.11: Matrice di confusione della F03-10-07

	precision	recall	f1-score	support
0	0.81	0.93	0.87	14
1	0.88	0.88	0.88	8
2	0.67	0.67	0.67	3
3	1.00	0.80	0.89	10
accuracy			0.86	35
macro avg	0.84	0.82	0.82	35
weighted avg	0.87	0.86	0.86	35

Figura 5.12: Performance della F03-10-07

5.2.2 F08

5.2.2.1 Risultati classificazione 3 classi

Per valutare i risultati della F08 è stato utilizzato lo stesso metodo introdotto per la F03. La valutazione delle prestazioni in termini di valore assoluto sono riportate nella tabella 5.7. La valutazione delle prestazioni in termini di valore medio sono riportare nella tabella 5.8.

Indice migliore	Indice peggiore	Migliori 10 indici	Valori di accuracy
F08-07-06-03-04	F08-08-07-05-06	F08-07-06-03-04	0,8571
*	*	F08-11-10-05-07	0,8286
*	*	F08-11-08-03-07	0,8286
*	*	F08-11-05-02-03	0,8286
*	*	F08-07-05-03-04	0,8286
*	*	F08-11-10-02-05	0,8
*	*	F08-11-05-03-04	0,8
*	*	F08-11-08-02-03	0,8
*	*	F08-11-07-03-06	0,8
*	*	F08-10-05-03-04	0,8

Tabella 5.7: Tabella performance indici in termini di valore assoluto F08

Indice migliore	Indice peggiore	Migliori 10 indici	Valori di accuracy
F08-07-06-03-04	F08-11-08-04-07	F08-07-06-03-04	0,7943
*	*	F08-11-08-03-07	0,7886
*	*	F08-07-05-03-04	0,7829
*	*	F08-10-08-02-03	0,7771
*	*	F08-11-10-02-05	0,7657
*	*	F08-11-05-02-03	0,7657
*	*	F08-11-05-03-04	0,76
*	*	F08-11-10-05-07	0,7543
*	*	F08-10-05-03-04	0,7486
*	*	F08-10-08-02-07	0,7314

Tabella 5.8: Tabella performance indici in termini di valore medio F08

Anche in questo caso l'indice migliore in termini di valore assoluto e medio è il medesimo, cioè F08-07-06-03-04, che ha raggiunto un livello massimo di accuracy del 86%. La tabella delle performance e la matrice di confusione sono riportare nelle figure 5.13 e 5.14.

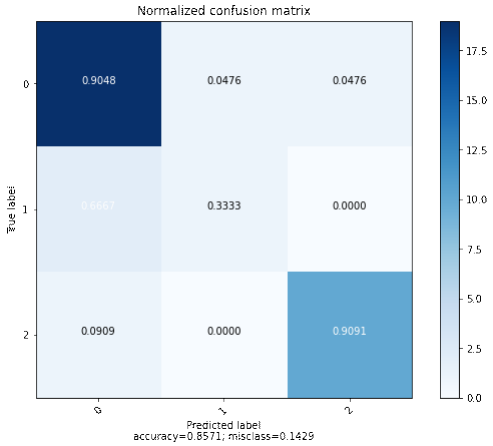

Figura 5.13: Matrice di confusione della F08-07-06-03-04(3 classi)

	precision	recall	f1-score	support
0	0.86	0.90	0.88	21
1	0.50	0.33	0.40	3
2	0.91	0.91	0.91	11
accuracy			0.86	35
macro avg	0.76	0.72	0.73	35
weighted avg	0.85	0.86	0.85	35

Figura 5.14: Performance della F08-07-06-03-04(3 classi)

Per completezza andiamo ad inserire anche i risultati della seconda combinazione migliore, in termini di robustezza, la F08-11-08-03-07. In questo caso è stato raggiunto un livello di accuracy pari al 83%. Nelle figure 5.15 e 5.16 sono illustrate la matrice di confusione e la tabella delle performance della migliore iterazione per F08-11-08-03-07.

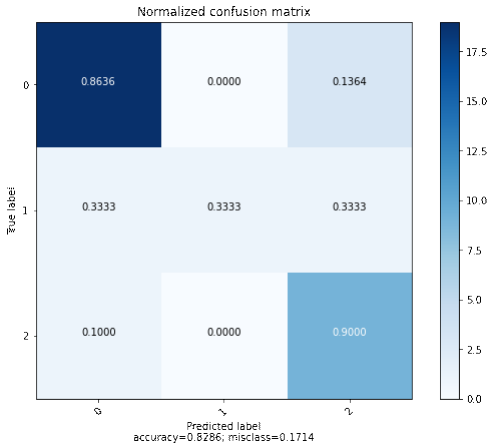

Figura 5.15: Matrice di confusione della F08-11-08-03-07(3 classi)

	precision	recall	f1-score	support
0	0.90	0.86	0.88	22
1	1.00	0.33	0.50	3
2	0.69	0.90	0.78	10
accuracy			0.83	35
macro avg	0.87	0.70	0.72	35
weighted avg	0.85	0.83	0.82	35

Figura 5.16: Performance della F08-11-08-03-07(3 classi)

5.2.2.2 Risultati classificazione 4 classi

Estendo il processo di classificazione da 3 a 4 classi si ottengono dei risultati diversi a parità di indice. Nelle figure 5.17 e 5.18 sono illustrate la matrice di confusione e la tabella delle performance della migliore iterazione di F08-07-06-03-04. In questo caso è stato raggiunto un livello di accuracy pari a 80%.

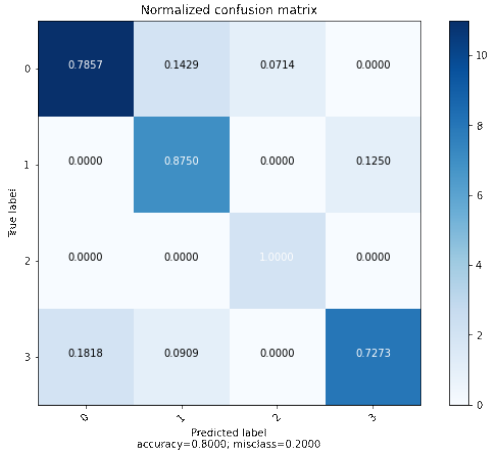

Figura 5.17: Matrice di confusione della F08-07-06-03-04

	precision	recall	f1-score	support
0	0.85	0.79	0.81	14
1	0.70	0.88	0.78	8
2	0.67	1.00	0.80	2
3	0.89	0.73	0.80	11
accuracy			0.80	35
macro avg	0.78	0.85	0.80	35
weighted avg	0.82	0.80	0.80	35

Figura 5.18: Performance della F08-07-06-03-04

Per completezza andiamo ad inserire anche i risultati della seconda combinazione migliore, la F08-11-08-03-07. In questo caso è stato raggiunto un livello di accuracy pari al 77%. Nella figure 5.20 e 5.19 è illustrata la matrice di confusione e la tabella delle performance dell'iterazione migliore per F08-11-08-03-07.

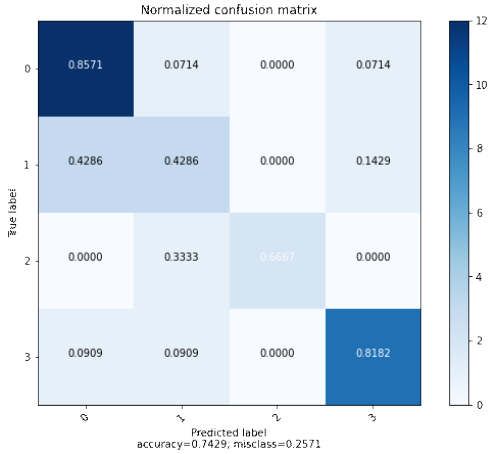

Figura 5.19: Matrice di confusione della F08-11-08-03-07

	precision	recall	f1-score	support
0	0.75	0.86	0.80	14
1	0.50	0.43	0.46	7
2	1.00	0.67	0.80	3
3	0.82	0.82	0.82	11
accuracy			0.74	35
macro avg	0.77	0.69	0.72	35
weighted avg	0.74	0.74	0.74	35

Figura 5.20: Performance della F08-07-06-03-04

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

In questo elaborato, sono stati illustrati due diverse metodologie per il riconoscimento delle comunità vegetali attraverso serie temporali NDVI rilevate a distanza. Le metodologie proposte sono state valutate su un singolo luogo (Monte Conero) e classificano le varie comunità vegetali in modo oggettivo, superando così la soggettività intrinseca del metodo fitosociologico. Gli obiettivi prefissati all'inizio del progetto sono stati raggiunti con successo. Sono stati generati modelli di predizione, basati sulle immagini satellitari di Sentinel-2, sia utilizzando il classificatore Random Forest sia la rete neurale InceptionTime. I risultati ottenuti possono essere ritenuti validi in quanto per entrambi i classificatori utilizzati sono stati raggiunti livelli di accuracy elevati. Per quanto riguarda il classificatore Random Forest la performance migliore per la funzione F03 è stato ottenuto per la time-series di indice 07-05 con un valore di accuracy pari al 80%. Mentre per gli indici della funzione F08 la prestazione migliore è stata ottenuta per la time-series 10-05-03-04 con un valore di accuracy pari a 78%. Per il classificatore InceptionTime, le performance migliore per la funzione F03 è stata ottenuta per la time-series di indice 07-04 con un valore di accuracy pari al 89%. Mentre per gli indici della funzione F08 la prestazione migliore è stata ottenuta per la time-series di indice 07-06-03-04 con un valore di accuracy pari al 80%. L'utilizzo delle time-series derivate nel processo di classificazione con Random forest non ha raggiunto i risultati attesi. Nonostante presentino risultati con livelli di accuracy elevati non sono tali da poter sostituire le time-series normali. In conclusione, considerando i risultati citati in precedenza, possiamo ritenere che tra i due classificatori proposti il migliore risulta essere la rete neurale InceptionTime.

6.2 Sviluppi futuri

In eventuali progetti futuri si potrebbe ripetere il processo di classificazione utilizzando algoritmi diversi come:

- **ResNet:**è una rete neurale convoluzionale molto profonda. Utilizzando la tecnica *skip connection* può addestrare fino a 152 livelli ;
- **AdaBoost:** un algoritmo che permette di convertire più modelli con un accuratezza bassa (classificatore debole) in modelli con accuratezza alta (classificatore forte);
- **Ranger:** una nuova implementazione del classificatore Random forest già presente nel pacchetto caret.

Bibliografia

[1] Simone Pesaresi, Adriano Mancini, Giacomo Quattrini, and Simona Casavecchia. Mapping mediterranean forest plant associations and habitats with functional principal component analysis using landsat 8 ndvi time series. *Remote Sensing*, 12(7), 2020.

[2] Esa. Indici di vegetazione. http://www.istitutoveneto.org/venezia/documenti/tesi_laurea_dott/tesi_costantini/cap3.pdf.

[3] Tin Kam Ho. Random decision forests. In *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–282, M, 1995. IEEE Computer Society.

[4] "Copernicus.eu". Normalized difference vegetation index. https://land.copernicus.eu/global/products/ndvi.

[5] Mirco Boschetti Lucio Bolzan, Mariano Bresciani, Claudia Giardino, Alba L'Astorina, Riccardo Lanari, Michele Manunta, Emanuela Mauri, and Eugenio Zilioli. "diffusione e sperimentazione della cartografia, del telerilevamento e dei sistemi informativi geografici, come tecnologie didattiche applicate allo studio del territorio e dell'ambiente". http://space4agri.irea.cnr.it/it/file/CATGISVol_3_1_ilTLR.pdf.

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

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

[8] Jorge Leonel. Supervised learning. https://medium.com/@jorgesleonel/supervised-learning-c16823b00c13, 2018.

[9] Eijaz Allibhai. Hold-out vs. cross-validation in machine learning-medium. https://medium.com/@eijaz/holdout-vs-cross-validation-in-machine-learning-7637112d3f8f, 2018.

[10] Afroz Chakure. Regressione forestale casuale. -https://medium.com/swlh/random-forest-and-its-implementation-71824ced454f, 2019.

[11] Jason Brownlee. Bagging and random forest for imbalanced classification. https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/, 2020.

[12] Wikipedia. Random forest. https://en.wikipedia.org/wiki/Random_forest#Algorithm.

[13] Flairnet. Machine learning overview. https://www.flair-net.it/wp-content/uploads/2019/12/Flairnet-Machine-Learning-White-Paper_Allegato-1.pdf.

[14] Marco Del Pra. Una panoramica dell'architettura e dei dettagli di implementazione dei più importanti algoritmi di deep learning per la classificazione delle serie temporali. https://ichi.pro/it/classificazione-delle-serie-storiche-con-deep-learning-231141987876207.

[15] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 2020.

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

[17] Jupyter project documentation. https://jupyter.org/.

[18] Wikipedia. https://it.wikipedia.org/wiki/PuTTY.

[19] Wikipedia. https://it.wikipedia.org/wiki/MATLAB.

[20] Google Research. Cos'è colaboratory? https://colab.research.google.com/notebooks/intro.ipynb#scrollTo=5fCEDCU_qrC0.

[21] Paul Gilbert and Ravi Varadhan. Package 'numderiv'. https://cran.r-project.org/web/packages/numDeriv/numDeriv.pdf, 2019.

[22] Prabhu. Medium-understanding hyperparameters and its optimisation techniques, 2018.

[23] Sarang Narkhede. Understanding confusion matrix, 2018.

Elenco delle figure

1.1 Immagine satellitare del Monte Conero. 6
1.2 Flowchart dei metodi proposti 7

2.1 Immagine sentinel-2A [6]. 10
2.2 Bande Sentinel [6]. 11
2.3 Schema suddivisione dataset mediante k-fold [9]. 13
2.4 Struttura del metodo Random forest [10]. 14
2.5 Struttura rete neurale biologica [13]. 15
2.6 Struttura rete neurale artificiale [13]. 16
2.7 Architettura InceptionTime [14]. 17
2.8 Architettura Modulo Inception [14]. 18
2.9 Logo di R [16]. 19
2.10 Logo di JupyterLab [17]. 20
2.11 Logo di Putty [18]. 20
2.12 Logo di Matlab [19]. 21
2.13 Logo di Colaboratory [20]. 21

3.1 Esempio di time-series 27
3.2 Esempio di time-series derivata 27

4.1 Esempio tabella performance RF. 32
4.2 Esempio matrice di confusione. 33
4.3 Esempio struttura tabella delle performance 36

5.1 Performance F03-07-05 RF 40
5.2 Matrice di confusione F03-07-05 RF 40
5.3 Performance F08-10-05-03-04 RF 41
5.4 Matrice di confusione F08-10-05-03-04 RF 41
5.5 Matrice di confusione della F03-07-04(3 classi) 44
5.6 Performance della F03-07-04(3 classi) 44
5.7 Matrice di confusione della F03-10-07(3 classi) 44
5.8 Performance della F03-10-07(3 classi) 44
5.9 Matrice di confusione della F03-07-04 45
5.10 Performance della F03-07-04 45

5.11 Matrice di confusione della F03-10-07 45
5.12 Performance della F03-10-07 45
5.13 Matrice di confusione della F08-07-06-03-04(3 classi) 47
5.14 Performance della F08-07-06-03-04(3 classi) 47
5.15 Matrice di confusione della F08-11-08-03-07(3 classi) 47
5.16 Performance della F08-11-08-03-07(3 classi) 47
5.17 Matrice di confusione della F08-07-06-03-04 48
5.18 Performance della F08-07-06-03-04 48
5.19 Matrice di confusione della F08-11-08-03-07 48
5.20 Performance della F08-07-06-03-04 48

Elenco delle tabelle

3.1 Esempio struttura time-series 26
3.2 Categorie di classi vegetali 28

4.1 Esempio struttura time-series FPCA 30
4.2 Esempio struttura time-series InceptionTime 33
4.3 Esempio struttura risultati InceptionTime 35

5.1 Struttura file Excel per valutare la performance Random forest . . 39
5.2 Tabella performance F03 con Random forest 40
5.3 Tabella performance F08 con Random forest 41
5.4 Tabella descrittiva per la performance di un indice 42
5.5 Tabella performance indici in termini di valore assoluto F03 43
5.6 Tabella performance indici in termini di valore medio F03 43
5.7 Tabella performance indici in termini di valore assoluto F08 46
5.8 Tabella performance indici in termini di valore medio F08 46