UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Sviluppo di un sistema di Intelligenza Artificiale per l'analisi di Time Series e Data Imputation nel contesto di Precision Farming

Development of an Artificial Intelligence system for Time Series Analysis and Data Imputation in the Context of Precision Farming

Relatore:
Dr. Galdelli Alessandro

Laureando:
Pescetti Leonardo

Correlatore:
Prof. Mancini Adriano

Anno Accademico 2023-2024

Possiamo vedere solo poco davanti a noi, ma possiamo vedere tante cose che bisogna fare.

- Alan Turing, Computing Machinery and Intelligence (1950).

Indice

Capitolo 1

Introduzione

Negli ultimi decenni, l'agricoltura è stata oggetto di considerevoli trasformazioni grazie all'avanzamento tecnologico e alla sempre maggiore disponibilità di dati. In particolare, l'introduzione dell'Agricoltura di Precisione ha rivoluzionato il modo in cui le aziende agricole operano, ottimizzando sia le attività sul campo, come la lavorazione dei terreni, sia la gestione digitale delle informazioni, grazie a software avanzati che consentono un controllo più efficiente delle operazioni e dei raccolti. Questa metodologia, che combina l'uso di tecnologie innovative come il GPS, i sensori remoti, i droni e i Sistemi Informativi Geografici (GIS) e software di raccolta ed elaborazione dati, consente ai produttori di adottare un approccio più mirato ed efficiente per la gestione delle coltivazioni.

In questa Tesi verrà approfondito l'impatto dell'agricoltura di precisione sull'efficienza operativa, la sostenibilità ambientale e la redditività delle aziende agricole analizzando come questa tecnologia possa ottimizzare le risorse, ridurre gli sprechi e migliorare la competitività del settore agricolo.
Tale approfondimento fatto mediante l'analisi di hardware e software che ci permettono di raccogliere dati così da costituire le cosiddette *Time Series* (Serie Temporali) con il conseguente processo di *Data Imputation* (Imputazione dei dati). Una corretta imputazione dei dati assicura strutture dati compatte e con una perdita minima di informazioni, consentendo ai software e dispositivi utilizzati nell'Agricoltura di Precisione di operare in modo ottimale, massimizzando i benefici e migliorando l'efficienza complessiva dei processi.

Si parla di Agricoltura di Precisione (nota anche come **AdP**) e di Agricoltura 4.0 e 5.0, poiché la nostra società è in continua evoluzione, e richiede nuovi approcci e tecnologie per affrontare sfide sempre più rilevanti, come:

1. *La continua crescita della popolazione*: con conseguente bisogno crescente di alimenti, come dimostrato dalle ultime stime delle Nazioni Unite (Figura 1.1) suggeriscono che la popolazione mondiale potrebbe crescere fino a circa 8,5 miliardi nel 2030, 9,7 miliardi nel 2050 e 10,4 miliardi nel 2100 [1].

2. *La riduzione delle aree coltivabili*: in occasione della Giornata Mondiale del Suolo, Coldiretti ha sottolineato che l'Italia ha perso quasi un terzo delle sue superfici agricole nel corso degli ultimi cinquant'anni, principalmente a causa della cementificazione e dell'abbandono delle campagne [2].

3. *L'inquinamento ed emissioni* causate da un utilizzo non ottimale dei mezzi agricoli, come l'impiego di macchinari obsoleti e inefficienti, la scarsa manutenzione dei motori e l'eccessivo uso di carburanti fossili, che aumenta il consumo energetico e le emissioni di gas inquinanti.

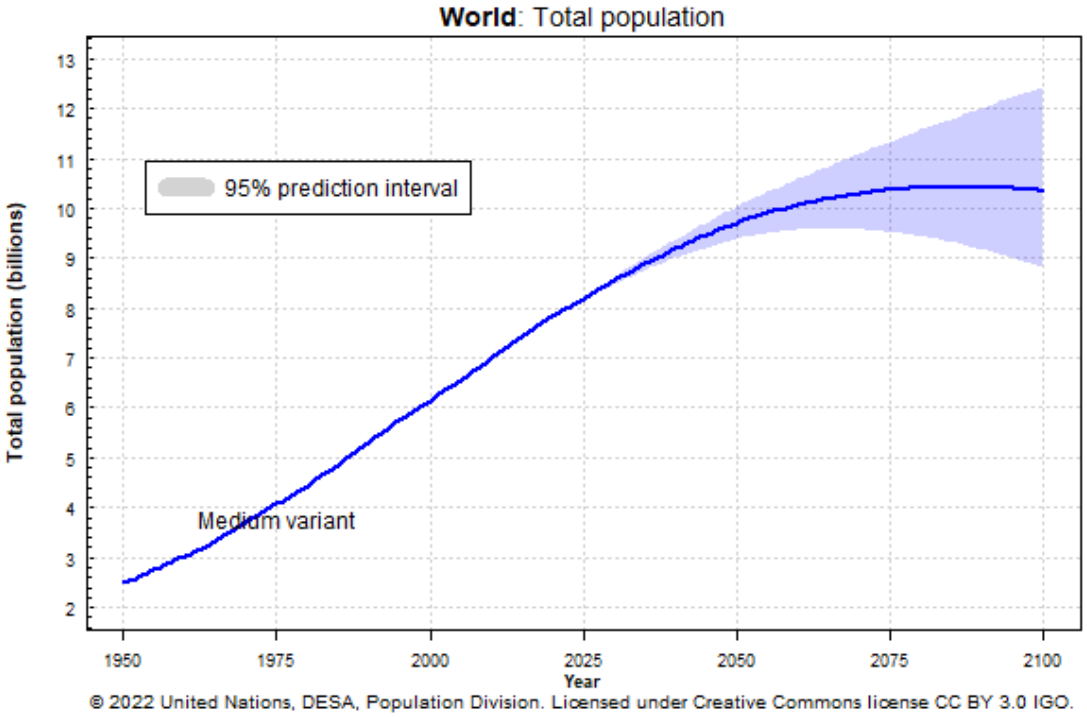

Figura 1.1: Stima della crescita demografica mondiale. Fonte [3].

Per affrontare le problematiche precedentemente menzionate, è essenziale un perfezionamento del settore agricolo, finalizzato a un uso più efficiente delle superfici coltivabili. Questo consentirebbe una produzione alimentare più sostenibile, contribuendo così a mitigare il problema della disponibilità di risorse alimentari.

In tale contesto, l'Agricoltura di Precisione (AdP) assume un ruolo cruciale, ma per garantirne l'efficacia è imprescindibile un adeguato supporto tecnologico. Di conseguenza, sorge il quesito: come può l'AdP essere ottimizzata dal punto di vista digitale?
Una possibile soluzione consiste nell'integrare dispositivi avanzati per il monitoraggio e la raccolta dati con software efficienti per l'analisi delle informazioni. Questa combinazione permette di ottenere una visione approfondita, facilitando scelte strategiche e ottimizzando le operazioni agricole.

Questo introduce il tema centrale della presente tesi: nei sistemi di gestione per la raccolta di dati organizzati come serie temporali (Time Series), possono verificarsi situazioni di incompletezza delle informazioni, spesso dovute a malfunzionamenti. Pertanto, verrà esplorato come affrontare la perdita dei dati, proponendo tecniche di Data Imputation per assicurare un'adeguata e continua operatività dei sistemi di gestione delle informazioni mancanti.

1.1 Obiettivi

L'obiettivo principale di questa tesi è sviluppare un' **applicazione di Intelligenza Artificiale per l'analisi di Time Series e Data Imputation nel contesto di Precision Farming**, che permetta di ricostruire le traiettorie e attività dei trattori agricoli. Questi dati vengono raccolti tramite dispositivi GPS e comprendono: dettagli precisi relativi alla posizione geografica, velocità operativa, consumo di carburante, efficienza complessiva e altre metriche chiave correlate alle prestazioni del trattore.

Si intende fornire agli agricoltori uno strumento efficace per analizzare correttamente i dati raccolti, anche in caso di informazioni incomplete. Questo consente di ottimizzare le decisioni aziendali, aumentando l'efficienza e la precisione delle operazioni agricole.

1.1.1 Applicazione di Time Series Data Imputation

Il linguaggio di programmazione principale sarà Python, scelto per sviluppare il sistema di raccolta dati e l'ambiente di addestramento dell'Intelligenza Artificiale necessario per implementare la corretta Data Imputation. Grazie alla sua flessibilità e alla vasta gamma di librerie disponibili, Python verrà utilizzato per elaborare i dati, sviluppare algoritmi e interagire con altri componenti del sistema, garantendo efficienza e interoperabilità.
Gli algoritmi verranno sviluppati in Visual Studio Code (VS Code), un ambiente di sviluppo integrato (o IDE, Integrated Development Environment) scelto per la scrittura e la gestione del codice Python.
Per il monitoraggio dei trattori, saranno utilizzati diversi dispositivi della famiglia Teltonika: questi si suddividono in unità di monitoraggio GPS, che individuano la posizione geografica del trattore in tempo reale e dispositivi che forniscono informazioni ambientali e dati di vario genere, come la rilevazione di ostacoli e i parametri del suolo.
Verrà utilizzata anche una piattaforma open-source di monitoraggio GPS: Traccar, che viene sfruttata per gestire i dispositivi di controllo e visualizzare le traiettorie. Tale piattaforma consentirà di monitorare i trattori in tempo reale, generare report dettagliati e analizzare le prestazioni dell'intero sistema. Da tali report verranno raccolti i dati che genereranno successivamente le Time Series.

L'obbiettivo finale è quello di ridurre lo spreco di risorse e di ottimizzare l'utilizzo dei fattori produttivi nell'ambito dell'agricoltura di precisione. Affrontando il problema della Data Imputation nelle Time Series, si mira a migliorare l'efficienza complessiva delle operazioni agricole, ridurre l'impatto ambientale e massimizzare i rendimenti.

1.2 Struttura della Tesi

La presente Tesi è strutturata per fornire una descrizione completa del lavoro svolto. L'introduzione fornisce il contesto della ricerca, definisce gli obiettivi principali e sottolinea l'importanza dell'applicazione della Time Series Data Imputation nell'ambito dell'agricoltura di precisione.

Successivamente, vengono trattati i temi dell'Agricoltura di Precisione e delle tecniche di Time Series e Data Imputation, che rappresentano i concetti centrali di questa Tesi. Vengono inoltre descritte le tecnologie utilizzate: il linguaggio di programmazione Python, l'ambiente di sviluppo Visual Studio Code, e i dispositivi Teltonika per il monitoraggio, oltre alle funzionalità del software Traccar.

Il progetto realizzato viene poi descritto in dettaglio, partendo da una panoramica generale fino all'analisi degli algoritmi di Intelligenza Artificiale utilizzati e dei risultati ottenuti.

Infine, vengono esposte considerazioni sull'uso futuro dell'applicazione e sui suoi possibili sviluppi.
La Tesi si conclude con la bibliografia, l'elenco delle figure e la pagina dei ringraziamenti.

Capitolo 2

L'Agricoltura di Precisione

Per Agricoltura di Precisione (AdP) si intende un insieme di tecnologie e strumenti applicati ai processi produttivi in agricoltura posti in essere al fine di migliorare la produzione, minimizzare i danni ambientali ed elevare gli standard qualitativi dei prodotti agricoli. La "precisione" introdotta da tali tecnologie consente di effettuare una distribuzione mirata dei principali fattori di produzione (acqua, fertilizzanti, fitofarmaci, etc.) solo dove serve e nella quantità corrispondente al reale fabbisogno della coltivazione in atto.

Il crescente interesse per l'AdP deriva dalle problematiche connesse al crescente aumento della popolazione mondiale (stimato dalla FAO in oltre 9 miliardi di persone entro il 2050, e dalle Nazioni Unite come citato nell'introduzione) e dalla conseguente necessità di incrementare la resa e la produttività dell'agricoltura a livello mondiale. Le tecnologie sulle quali si fonda l'AdP, consentono, quindi di ottenere sistemi produttivi sempre più efficienti e sostenibili, aspetto che colloca l'agricoltura in un contesto moderno e di attualità [4].

Possiamo definire l'AdP come una gestione aziendale basata sull'osservazione, la misura e la risposta di variabili quantitative e qualitative come l'umidità del suolo, la qualità del raccolto, e la composizione chimica del terreno, sia a livello intercampo che intracampo, che intervengono nell'ordinamento produttivo. Ciò al fine di definire, dopo analisi dei dati sito-specifici, un sistema di supporto decisionale per l'intera gestione aziendale, con l'obiettivo di ottimizzare i rendimenti nell'ottica di una sostenibilità climatica, ambientale, economica, produttiva e sociale [5].

È possibile identificare diverse fasi nell'implementazione dell'Agricoltura di Precisione:

1. La prima fase consiste nella misura ed interpretazione della variabilità spazio-temporale associata a tutti gli aspetti della produzione agraria, tra-

mite l'acquisizione di dati ambientali negli agroecosistemi[1] e l'elaborazione degli stessi utilizzando metodologie innovative. Il prodotto finale è la delineazione del campo in aree con caratteristiche sufficientemente omogenee.

2. La seconda fase utilizza l'informazione raccolta nella fase precedente per adattare gli input agronomici (ad esempio: acqua, fertilizzanti, prodotti fitosanitari, etc.) alle specifiche condizioni locali, differenziando così gli interventi agronomici all'interno di uno stesso appezzamento.

3. La terza fase consiste nella validazione della metodologia, in modo da calibrare le direttive gestionali prima del suo trasferimento agli agricoltori.

Figura 2.1: Figura esemplificativa che mostra in generale come lavora l'AdP, unendo strumenti digitali con l'agricoltura. [2]

[1] In scienze agrarie, ecosistema secondario caratterizzato dall'intervento umano finalizzato alla produzione agricola e zootecnica

[2] Immagine creata tramite il software Microsoft Designer.

2.1 Origini dell'Agricoltura di Precisione

L'Agricoltura di Precisione ha avuto le sue origini negli anni '80, quando le prime tecnologie GPS sono diventate disponibili per uso commerciale. Queste sono state inizialmente sviluppate per fornire una navigazione precisa per applicazioni militari e aerospaziali, ma hanno rapidamente attirato l'attenzione del settore agricolo per la loro capacità di migliorare l'efficienza delle attività agricole. L'idea di utilizzare il GPS e altre tecnologie per gestire le colture in modo più efficiente è stata il punto di partenza per lo sviluppo dell'AdP.

I pionieri dell'AdP sperimentarono dunque l'uso di queste nuove tecnologie per migliorare la precisione delle operazioni agricole. Ad esempio, venivano utilizzati sistemi GPS per tracciare i percorsi dei trattori nei campi e per mappare le variazioni nei rendimenti delle colture. Queste prime sperimentazioni hanno dimostrato il potenziale delle tecnologie di precisione nell'ottimizzare la gestione delle risorse agricole ispirando ulteriori sviluppi nel campo.

Con il passare del tempo, l'AdP ha iniziato a diffondersi su larga scala, con sempre più agricoltori che adottavano le nuove tecnologie disponibili. Ciò è stato favorito dall'avanzamento delle tecnologie stesse, che sono diventate sempre più sofisticate e accessibili in termini di costo.

I concetti di agricoltura "1.0", "2.0", "3.0", "4.0" rappresentano una progressione nel modo in cui l'agricoltura è stata praticata e gestita nel corso del tempo, in risposta ai cambiamenti tecnologici, economici, sociali e ambientali (Figura 2.2).

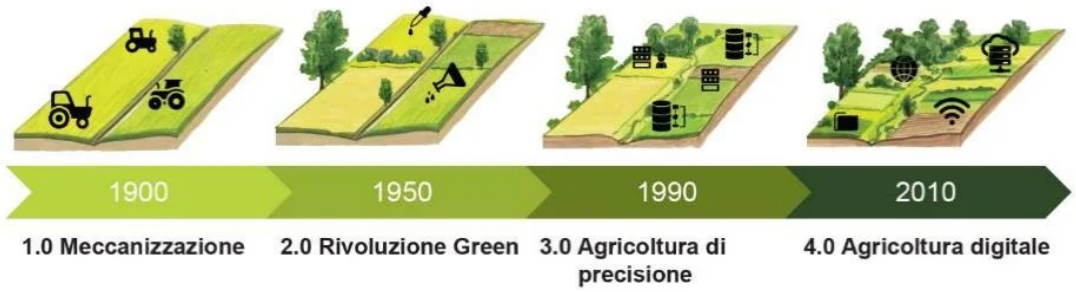

Figura 2.2: La figura mostra l'evoluzione dell'AdP nel tempo.

- **Agricoltura 1.0** Questa fase rappresenta l'agricoltura tradizionale e pre-industriale, caratterizzata da metodi agricoli manuali o dipendenti dagli animali. E' stata dominante per la maggior parte della storia umana fino all'avvento della rivoluzione industriale.

- **Agricoltura 2.0** Con l'avvento della rivoluzione industriale nel XVIII e XIX secolo, nuove tecnologie come la macchina a vapore, il trattore e i fertilizzanti chimici hanno aumentato la produttività agricola e ridotto la dipendenza dal lavoro manuale.

- **Agricoltura 3.0** L'agricoltura 3.0 è stata guidata dall'era dell'informazione e della tecnologia digitale che hanno permesso agli agricoltori di raccogliere e analizzare dati in tempo reale per prendere decisioni più informate sulla gestione delle colture e delle fattorie.

- **Agricoltura 4.0** Questa fase è caratterizzata dall'integrazione di tecnologie avanzate come l'Intelligenza Artificiale (IA), il machine learning, la robotica e la biotecnologia nell'agricoltura. L'obiettivo è migliorare ulteriormente l'efficienza, la sostenibilità affrontando così sfide come il cambiamento climatico e la sicurezza alimentare [6].

2.2 Agricoltura 4.0

L'agricoltura 4.0 rappresenta una fase di evoluzione dell'agricoltura tradizionale che sfrutta tecnologie avanzate e digitali per migliorare l'efficienza, la sostenibilità e la produttività del settore agricolo. Questa si basa sull'integrazione di tecnologie come *l'Intelligenza Artificiale (IA), l'Internet delle Cose (IoT*[3]*), la robotica, i droni, l'analisi dei dati (Big Data Analytics), Sistemi Informativi Geografici (GIS*[4]*)* (Figura 2.3).

Si può pertanto affermare che questa tipologia di agricoltura è caratterizzata da un elevato livello di digitalizzazione delle operazioni, il che consente agli agricoltori di raccogliere, elaborare e analizzare grandi volumi di dati, al fine di prendere decisioni accurate e tempestive, ottimizzando la pianificazione delle colture e migliorando la gestione delle risorse.

Figura 2.3: Agricoltura 4.0.[5]

[3]Con "**IoT**" intendiamo una serie di dispositivi in grado di autoalimentarsi che sono geolocalizzati (quindi ci permettono di conoscere la loro posizione), sono dotati di numerosi sensori che rilevano costantemente una serie di dati (geologici, ambientali, di contesto, di funzionamento) e li trasferiscono attraverso la rete internet a server che li archiviano ed elaborano.

[4]Il **GIS** è un sistema che crea, gestisce, analizza e mappa tutti i tipi di dati; quindi li collega a una mappa, integrando i dati sulla posizione con tutti i tipi di informazioni descrittive. Ciò fornisce una base per la mappatura e l'analisi utilizzata nella scienza e in quasi tutti i settori.

[5]Immagine creata tramite il software Microsoft Designer.

I vantaggi dell'Agricoltura 4.0 per le aziende possono essere identificati principalmente nella razionalizzazione dell'uso delle risorse, con ricadute prevalentemente economiche per le imprese della filiera. Tuttavia, un processo produttivo orientato alla massimizzazione della sostenibilità genera un impatto positivo anche sulla salute pubblica, poiché è possibile offrire ai consumatori finali prodotti più controllati e freschi rispetto a quelli ottenuti con le tecniche tradizionali. Per quantificare questi vantaggi, si parla di un risparmio attorno al 30% per gli input produttivi e di un aumento del 20% della produttività, con un utilizzo limitato di sostanze chimiche.

Per quanto riguarda l'impiego dei dati, va evidenziato che l'analisi in tempo reale delle informazioni provenienti dai campi risulta estremamente utile per una gestione più rapida ed efficiente di tutte le attività legate all'agricoltura. (Figura 2.4). Ad esempio mediante l'analisi dei dati è possibile improntare al massimo dell'efficienza l'utilizzo delle macchine agricole, o utilizzare soltanto la quantità di acqua necessaria, senza sprechi. Mediante lo stesso set di informazioni è possibile prevenire le patologie delle piante o contrastarne i parassiti, limitando i danni grazie al monitoraggio costante e simultaneo delle coltivazioni.

Figura 2.4: Dati della lavorazione del terreno costantemente raccolti, elaborati e archiviati grazie all'IoT [7].

2.2.1 Uno sguardo al futuro: Agricoltura 5.0

Questa è una fase futura ed ipotetica che si concentra sulla convergenza tra l'agricoltura ed altre tecnologie emergenti come l'Intelligenza Artificiale avanzata, la nanotecnologia, la bioingegneria e la tecnologia quantistica. L'agricoltura 5.0 potrebbe portare a rivoluzionarie innovazioni e pratiche agricole completamente nuove, consentendo una maggiore sostenibilità, resilienza e produttività nell'agricoltura del futuro.
Vanno inoltre considerate l'inclusione di robotica, realtà aumentata e tecnologie 6G le quali rappresentano un avanzamento significativo, consentendo il monitoraggio e l'automatizzazione in tempo reale delle pratiche agricole. L'uso dell'Intelligenza Artificiale e dei Big Data rivestiranno un ruolo fondamentale nell'Agricoltura 5.0, fornendo una panoramica ottimale per decisioni fondamentali e analisi predittive (Figura 2.5).
Nel corso della transizione verso l'Agricoltura 5.0, devono essere considerate sfide socio-economiche, tra cui il possibile ostacolo all'adozione delle tecnologie da parte delle comunità agricole meno avanzate dal punto di vista tecnologico, nonché la necessità di modifiche comportamentali da parte degli agricoltori per favorire un'adozione rapida e una personalizzazione efficace delle soluzioni tecnologiche. Nonostante ciò, l'Agricoltura 5.0 offre ampie opportunità per migliorare la sostenibilità, la produttività e la redditività del settore agricolo, influenzando profondamente il futuro dell'agricoltura su scala globale [8].

Figura 2.5: Agricoltuta 5.0, l'AdP del futuro [9].

2.3 Tecnologie GIS e GPS nell'AdP

L'AdP si suddivide categoricamente in tre ambiti principali: terrestre, aereo e satellitare. La tecnologia terrestre si concentra su attività come la pianificazione della produzione, la mappatura, l'esplorazione e il controllo preciso delle macchine. Le dimensioni aerea e satellitare, invece, affrontano sfide più ampie, permettendo un'analisi del rendimento in tempo reale che supera i confini geografici. Un approccio sinergico che integra queste tecnologie consente di sfruttare appieno tutte le informazioni disponibili, ottimizzando le pratiche agricole.

Si parla dunque di *GPS (Global Positioning Systems) e GIS (Geographic Information Systems)*: l'integrazione dei sistemi *GPS* e *GIS* rappresenta il fulcro dell'AdP (Figura 2.6). Questo coesione semplifica la raccolta precisa di dati geolocalizzati, permettendo agli agricoltori di generare mappe dettagliate per l'applicazione di trattamenti differenziati, l'analisi del suolo e il monitoraggio dei parassiti. Grazie al GIS, inoltre, viene facilitata l'analisi spaziale, consentendo agli agricoltori di prendere decisioni basate su una valutazione complessiva e multidimensionale dei dati a disposizione [10].

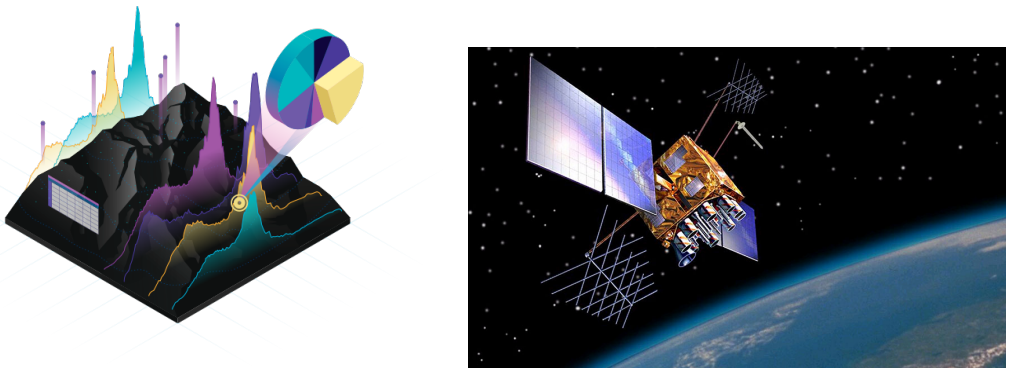

Figura 2.6: Geographic Information Systems (GIS) e Global Positioning Systems (GPS) [10].

Esempi concreti di questi argomenti che saranno successivamente illustrati nella Tesi, includono i dispositivi di tracciamento GPS *Teltonika* assieme all'utilizzo della piattaforma *Traccar*.

2.4 Big Data Analytics

Con il termine Big Data si fa riferimento a dati che contengono una maggiore varietà, che vengono generati in quantità sempre maggiori e con una maggiore velocità (Figura 2.7).

Sono quindi set di dati più grandi e complessi, a tal punto che i software di elaborazione dati tradizionali non sono in grado di gestirli correttamente. Ma questi enormi volumi di dati possono essere utilizzati per affrontare problemi aziendali di natura molto più complessa rispetto alla norma, rendendo dunque i Big Data fondamentali.

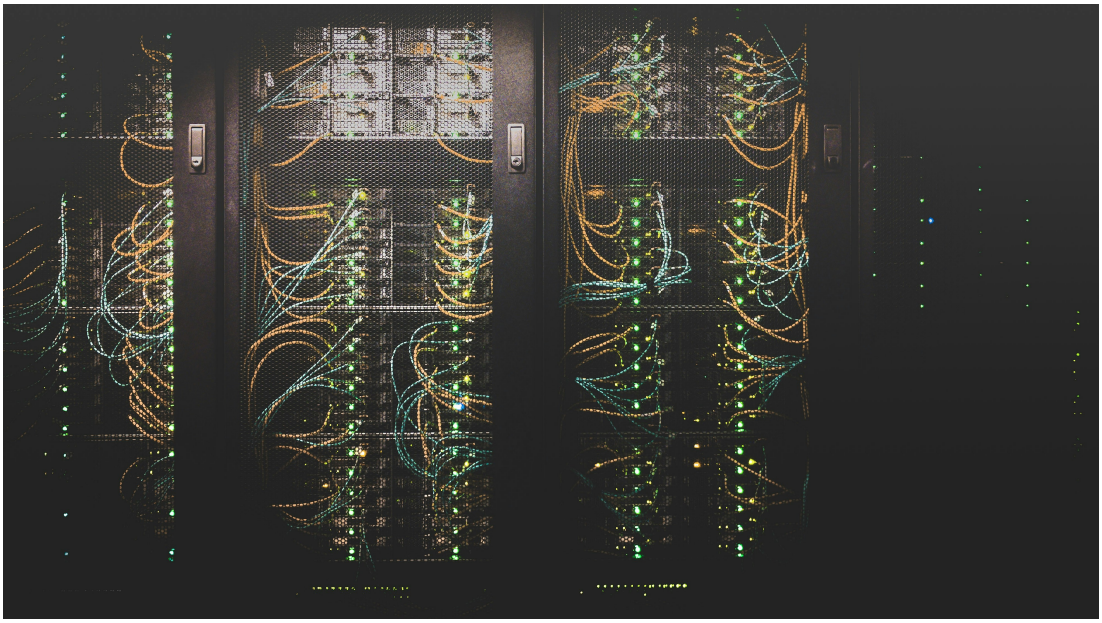

Figura 2.7: Big Data [11].

Il concetto che caratterizza i Big Data è noto come le *5 V*, qui di seguito riportate:

- **Volume** Riferito all'elevato volume di dati, i quali sono non strutturati e a bassa densità (cioè che sono distribuiti in modo sparso e che possono essere altamente variabili in termini di qualità e contenuto).
- **Velocità** Riferito all'elevata velocità con cui questi dati fluiscono.
- **Varietà** Intesa come le numerose tipologie dei dati disponibili. Con l'avvento dei Big Data, i dati arrivano come nuovi tipi di dati non strutturati.

Questi dati come testo, audio e video, richiedono un'ulteriore elaborazione preliminare per ricavare significato e supportare i così i *Metadati*[6].

- **Valore** Si riferisce al potenziale valore che i dati possono fornire quando vengono analizzati e utilizzati in modo efficace.
- **Veridicità** Indica la qualità e l'affidabilità dei dati. Poiché i Big Data possono essere generati da una vasta gamma di fonti, è importante valutare la qualità e la precisione dei dati per garantire che siano affidabili e utilizzabili per l'analisi [12].

Un'aspetto interessante da trattare per quanto riguarda la tematica dei Big Data è la *Big Data Analytics*. Con tale nomenclatura si intendono i metodi, le applicazioni e gli strumenti usati per raccogliere, elaborare e ottenere informazioni dettagliate da vari set di dati di volume elevato e ad alta velocità. Tali set di dati provengono da diverse origini, nonchè da dispositivi facenti parte dell' IoT.

Grazie all'analisi dei Big Data, le imprese possono sfruttare tali informazioni per migliorare rapidamente il loro modo di lavorare, pensare e offrire i vari servizi ai clienti. Con il supporto di strumenti e applicazioni, i Big Data possono dunque fornirci informazioni dettagliate, ottimizzare le operazioni e prevedere risultati futuri (figura 2.8).

Figura 2.8: Le principali applicazioni dei Big Data [13].

[6]Sono dati che forniscono informazioni descrittive su altri dati come contesto e struttura per facilitare l'interpretazione dei dati.

È possibile studiare la correlazione esistente tra Big Data Analytics e l'AdP, definita come *Agricolture Analytics*.
Difatti l'AdP è uno dei contesti principali in cui viene utilizzata l'analisi dei Big Data nel settore agricolo. L'AdP come specificato precedentemente si basa sull'uso di tecnologie avanzate e di dati dettagliati per ottimizzare le pratiche agricole nell'interesse dei produttori e consumatori.

Nel contesto dell'analisi dei Big Data, l'AdP si avvale di una vasta gamma di strumenti e tecnologie, che possono includere sensori remoti, Sistemi di Posizionamento Globale (GPS), droni, stazioni meteo e dispositivi IoT (Internet delle Cose). Tali strumenti raccolgono una grande quantità di dati su vari aspetti agricoli, come condizioni del suolo, umidità, temperatura, vegetazione, resa delle colture e condizioni meteo e altro ancora (Figura 2.9).

Figura 2.9: Agricolture Analytics [14].

L'analisi dei Big Data consente di elaborare questi dati in modo da estrarre informazioni significative e fornire report utili per i produttori agricoli, inoltre si possono sfruttare anche modelli predittivi e algoritmi avanzati (supportati da IA) per prevedere le tendenze future ed effettuare scelte preventive [15].

Il concetto di Big Data offre un'infrastruttura potente e sofisticata per la raccolta, l'archiviazione e l'analisi di enormi volumi di dati provenienti da una vasta gamma di sorgenti. Questi dati comprendono anche le *Time Series*, che rappresentano un tipo essenziale di dati organizzati in sequenze cronologiche. La loro inclusione nei Big Data riflette la crescente importanza di comprendere e sfruttare il potenziale informativo dei dati temporali per prendere decisioni con estrema precisione e anticipare le tendenze future.

Capitolo 3

Time Series Data Imputation

Con la nascita dell'era digitale il crescente aumento di dati ha raggiunto proporzioni senza precedenti in molteplici campi disciplinari. Tra le forme più comuni di dati, le *Time Series* (Serie Temporali) rivestono un ruolo cruciale, rappresentando un insieme di osservazioni raccolte in momenti diversi nel tempo. Tuttavia, le Time Series spesso presentano valori mancanti dovuti a diversi fattori, e tale fenomeno di dati assenti può compromettere significativamente l'analisi e l'interpretazione delle serie temporali, limitando l'efficacia delle tecniche di analisi e previsione.

In risposta a questa sfida, l'impiego di tecniche di *Data Imputation* (Imputazione dei Dati) si è dimostrato cruciale per completare e correggere le serie temporali, consentendo un'analisi più accurata e una migliore previsione dei fenomeni in esame.

3.1 Time Series

Una *Time Series* è un insieme di dati osservati, misurati o raccolti in momenti diversi nel tempo. Questi dati sono organizzati in sequenza cronologica, dove ciascuna osservazione è associata a una specifica unità temporale, come ore, giorni, mesi, anni. Le Time Series sono comunemente utilizzate in vari campi, come l'economia, le scienze sociali, l'ingegneria, le scienze ambientali e vengono sfruttate per analizzare il comportamento di variabili nel tempo e identificare tendenze, stagionalità, cicli o eventi anomali [16].

Queste possono essere rappresentate in diversi modi, come grafici a linea, diagrammi a barre, diagrammi a dispersione (Figura 3.1). Inoltre i dati delle Time Series possono essere analizzati tramite diverse tecniche statistiche e matematiche, come la regressione, la decomposizione, i modelli *ARIMA* (Autoregressive

Integrated Moving Average), i modelli di previsione al fine di fare previsioni o estrarre informazioni significative dai dati nel tempo.

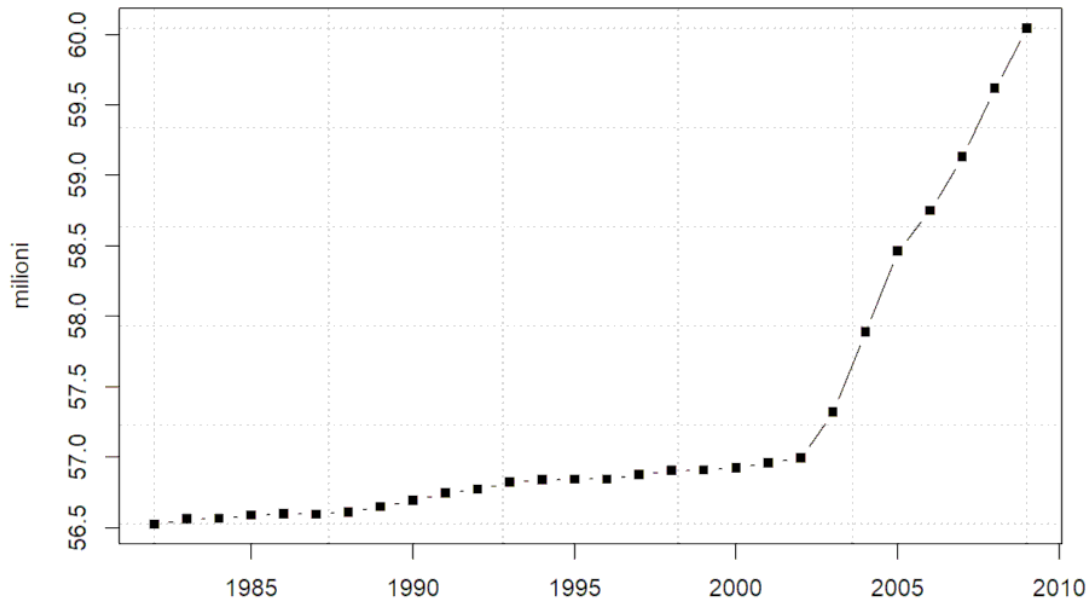

Figura 3.1: Esempio di Time Series, che mostra la variazione del numero della popolazione in Italia [17].

3.2 Data Imputation

Con *Data Imputation* si fa riferimento al processo di stima o sostituzione di valori mancanti in un insieme di dati di una *Time Series* con valori verosimili o attendibili ed è ampiamente utilizzata in diverse discipline compresa l'ingegneria, la medicina, l'economia, le scienze sociali e molte altre (Figura 3.2).

Sono molteplici le cause per cui i dati possono essere persi da una Time Series, e possono essere: errori di registrazione o raccolta dei dati, guasti nei sensori o dispositivi di rilevamento, interruzioni nella trasmissione dei dati (se per esempio provengono da un sistema di comunicazione e avviene un errore di connessione), indisponibilità dei dati (potrebbero esserci delle limitazioni pratiche che impediscono la raccolta di determinate misurazioni in certi momenti), gestione dei dati (che può riguardare la modifica, archiviazione e cancellazione dei dati).

Quando si verificano tali situazioni e si perdono dati da una Time Series, la Data Imputation diviene quindi essenziale per mantenere la completezza e

l'affidabilità dei dati raccolti, consentendo l'analisi e la modellizzazione delle Time Series in modo accurato e affidabile o per la costruzione di modelli predittivi.

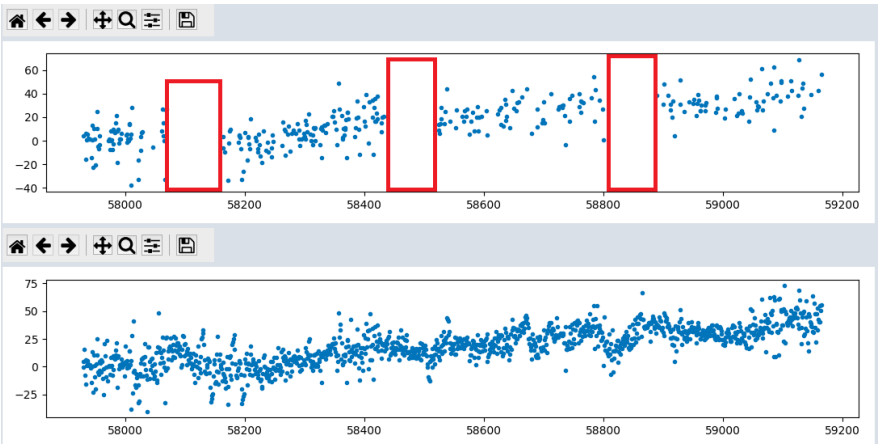

Figura 3.2: Esempio di Data Imputation: i riquadri rossi evidenziano i dati mancanti, mentre il grafico sottostante mostra il risultato di un'imputazione accurata [18].

3.3 Classificazione dei modelli

Verranno ora esaminati i modelli più comunemente utilizzati nella Data Imputation delle Time Series, i quali possono essere classificati in cinque categorie distinte:

1. **Metodi di imputazione semplice**

   - **Imputazione con valore costante**: Sostituisce i valori mancanti con un unico valore, come la media, la mediana o la moda dei dati noti.
   - **Forward fill e Backward fill**: Sostituisce i valori mancanti con il valore più vicino noto rispettivamente avanti o indietro nel tempo (cioè viene utilizzato il valore dell'osservazione più vicina disponibile verso il passato o futuro).
   - **Last Observation Carried Forward (LOCF)**: Sostituisce i valori mancanti con l'ultimo valore noto posto avanti nel tempo (è un caso specifico di *Forward Fill* che si riferisce specificamente all'utilizzo dell'ultimo valore noto).
   - **Last Observation Carried Backward (LOCB)**: Sostituisce i valori mancanti con l'ultimo valore noto posto all'indietro nel tempo (caso specifico di *Backward Fill*).

2. **Metodi di interpolazione**

   - **Linear Interpolation**: Stima i valori mancanti interpolando linearmente tra i valori noti prima e dopo il dato mancante.
   - **Polynomial Interpolation**: Utilizza un polinomio per interpolare i valori mancanti, adattando la forma del polinomio ai dati circostanti.
   - **Cubic Spline**: Approssima i valori mancanti utilizzando una curva spline cubica[1] che si adatta ai dati noti.

3. **Metodi statistici o model-based**

   - **AutoRegressive Integrated Moving Average (ARIMA)**: Utilizza un modello autoregressivo integrato di media mobile per predire i valori mancanti.
   - **Neural networks**: Si serve di reti neurali artificiali per apprendere e predire i valori mancanti in base ai dati noti.
   - **Random Forests**: Fa uso di strutture dati ad albero decisionali basati su *Bagging*[2] per stimare i valori mancanti.

4. **Metodi basati su Cluster**

   - **Imputazione con Cluster**: Suddivide la serie temporale in *Cluster*[3] di osservazioni simili e stima i valori mancanti all'interno di ciascun Cluster.
   - **K-nearest neighbors (KNN)**: Stima i valori mancanti basandosi sui valori noti dei punti più simili nella serie temporale.

5. **Metodi avanzati**

   - **Multiple imputation**: Crea più stime per ciascun valore mancante e combina le stime per ottenerne una finale.
   - **Markov Chain Monte Carlo (MCMC)**: Utilizza campionamento dato dalle *Catene di Markov*[4] per generare campioni dalla distribuzione che tiene conto sia dei dati noti che dei dati mancanti.

[1] Curva interpolante utilizzata per approssimare una serie di punti dati.

[2] Metodo di apprendimento d'insieme comunemente utilizzato per ridurre la varianza in un set di dati rumorosi.

[3] Insieme di items connessi in parallelo.

[4] Sono sequenze di eventi in cui la probabilità che un evento successivo avvenga dipende solo dallo stato corrente e non dalla storia degli eventi precedenti.

- **Deep Learning**: Utilizza modelli di *Deep Learning* o trasformatori per l'imputazione dei dati come Generative Adversarial Network (GAN), Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), Non-Autoregressive Outfilling and Multistep Imputation (NAOMI) i quali verranno successivamente esaminati.

3.3.1 Modelli di Deep Learning

Negli ultimi anni, i modelli di Deep Learning hanno guadagnato popolarità come potenti strumenti per la Data Imputation nelle Time Series. Questi modelli, che sfruttano reti neurali artificiali, sono in grado di catturare le complesse relazioni temporali presenti nei dati e di generare stime accurate per i valori mancanti. I principali modelli saranno presentati di seguito:

1. **Recurrent Neural Network (RNN)**

   Sono una tipologia di rete neurale progettata per lavorare con dati sequenziali, come le Time Series. Costituiscono modelli che sono in grado di catturare le dipendenze sequenziali nei dati, rendendoli adatti per le serie temporali con informazioni temporali complesse (quindi memorizzano informazioni sulle osservazioni precedenti e le utilizzano per influenzare la previsione delle osservazioni successive). La sua struttura presenta tre componenti principali: *Input Layer* (accetta l'input iniziale nella sequenza), *Hidden Layer* (ogni nodo in questo strato riceve dati dall'input layer e dai nodi dello strato nascosto precedente, e li elabora), *Output Layer:* (questo strato produce l'output della rete neurale). Tale approccio può essere particolarmente utile quando i dati mancanti sono correlati a osservazioni precedenti o successive nelle Time Series (Figura 3.3).

2. **Generative Adversarial Network (GAN)**

   Le GAN sono un tipo di modello generativo che può essere utilizzato per l'imputazione dei dati mancanti. Questi modelli sono in grado di generare nuovi campioni dai dati di input e possono essere addestrati per imitare la distribuzione dei dati osservati (Figura 3.4).

   Questo modello si basa sulle Recurrent Neural Network (RNN): in generale le GAN sono composte da due reti neurali: il *generatore* e il *discriminatore*. Il generatore è responsabile della generazione di dati stimati per sostituire i valori mancanti nelle *Multivariate Time Series* (Serie Temporali Multivariate[5]), mentre il discriminatore valuta la qualità dei dati generati confrontandoli con i dati reali [20].

[5]Insieme di dati osservati in momenti temporali successivi, dove ogni osservazione è costituita da più variabili.

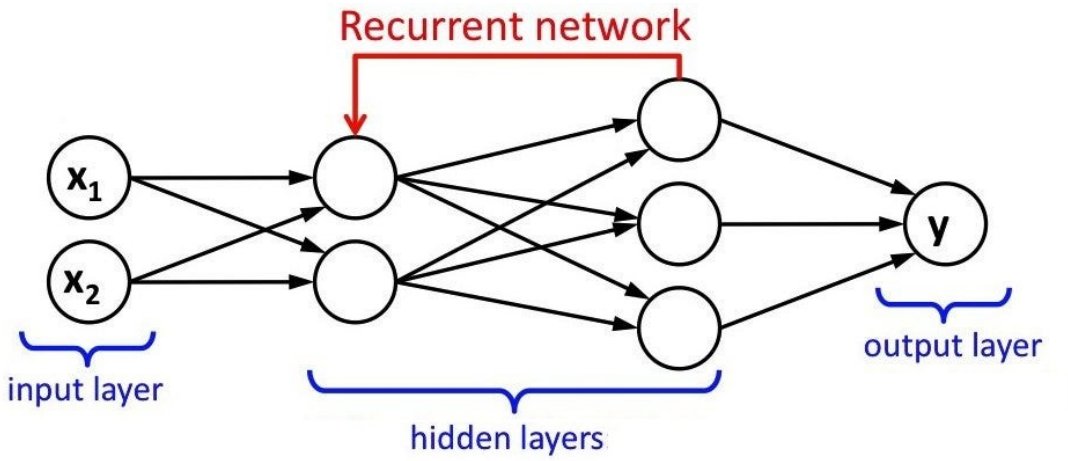

Figura 3.3: Architettura RNN [19].

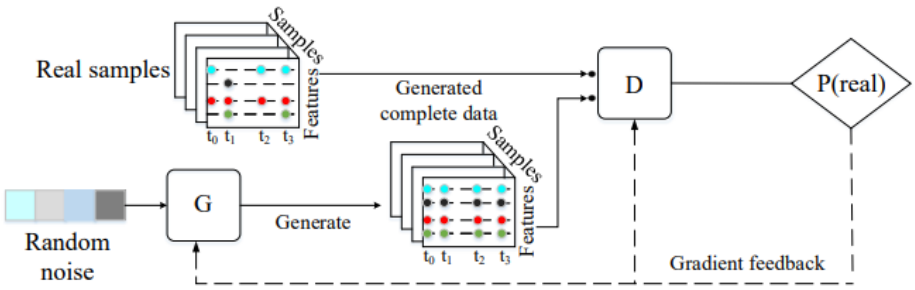

Figura 3.4: Architettura GAN [20].

3. **Gated Recurrent Unit(GRU)**

   Si tratta di un modello di imputazione dei dati che si basa sulle Recurrent Neural Network (RNN). Il GRU sfrutta una *Imputation Gate (Porta di Imputazione)* utilizzata per controllare quanto un valore mancante dovrebbe essere imputato, quindi consente al GRU di decidere quanto del valore mancante dovrebbe essere stimato e incorporato nella previsione corrente. E' caratterizzato da un meccanismo di gating per regolare il flusso di informazioni attraverso la rete, consentendo di catturare relazioni a lungo termine nei dati sequenziali.

4. **Non-Autoregressive Outfilling Multistep Imputation(NAOMI)**

   Il funzionamento di NAOMI si basa sull'utilizzo di tecniche di deep learning, come reti neurali, per apprendere i modelli dai dati noti. Questi modelli vengono quindi utilizzati per generare stime accurate dei valori mancanti. L'approccio non-autoregressivo e il processo di *outfilling* (estende i dati noti oltre i valori mancanti per facilitare l'imputazione) consentono a NAOMI di catturare relazioni complesse nei dati.

3.3.2 Modelli innovativi di Deep Learning

I modelli più utilizzati ad oggi per la gestione dell'imputazione dei dati nelle Time Series, come l'RNN (Recurrent Neural Networks) hanno lo svantaggio di incapparsi nel problema dell'*error compounding*, ossia un fenomeno in cui gli errori si accumulano o si amplificano progressivamente nel corso di iterazioni ripetute di un processo nel tempo. Ciò si verifica quando, ad esempio, si intende imputare un dato specifico x_t al tempo t, basandosi sul dato precedente x_{t-1} al tempo t-1; tuttavia, se il dato precedente contiene errori, questi si rifletteranno su x_t, causandone un significativo aumento nel tempo [21].

Un metodo per attenuare questa problematica è il NAOMI (Non-Autoregressive Multiresolution Sequence Imputation) utilizzato per imputare dati mancanti nelle Time Series senza dipendenze autoregressive(ovvero senza dover fare riferimento a valori passati della serie stessa), ma tale modello è applicabile solo per serie temporali campionarie regolari.

Pertanto, sono stati sviluppati nuovi approcci al fine di risolvere tale problematica e condurre un'efficace Data Imputation: tali metodologie includono il *Non-Recurrent Time Series Imputation (NRTSI)* e il *Bidirectional Recurrent Imputation for Time Series (BRITS)*, il quale sarà impiegato nel contesto dello sviluppo dell'applicazione di Data Imputation all'interno del lavoro di tesi.

3.3.3 Non-Recurrent Time Series Imputation (NRTSI)

Il modello *Non-Recurrent Time Series Imputation (NTRSI)* non fa uso di una struttura ricorrente per modellare le dipendenze temporali nei dati. Piuttosto, interpreta le Time Series come un insieme di tuple, ciascuna composta da un elemento di tempo e i dati corrispondenti. Questo approccio consente di imputare direttamente istanti temporali campionati in modo irregolare, poiché l'insieme può contenere tuple relative a punti temporali arbitrari, inclusi quelli casuali o irregolari [21].

Poiché le informazioni temporali sono incorporate nelle tuple (tempo, dati), la sequenzialità della Time Series rimane intatta. Di conseguenza, è agevole convertire l'insieme di tuple in una sequenza, preservando l'ordine temporale dei dati.

Nel contesto dell'analisi del modello Non-Recurrent Time Series Imputation (NRTSI), è stato eseguito un esperimento per confrontare l'efficacia di NRTSI con altri modelli di Data Imputation Time Series comunemente utilizzati. L'esperimento è basato su un'indagine condotta nell'articolo *"NRTSI: Non-Recurrent Time Series Imputation"* di Siyuan Shan, Junier B. Oliva e Yang Li [21].

L'esperimento verte sulle traiettorie di palline su un tavolo da biliardo. Il dataset utilizzato comprende 4.000 sequenze di allenamento e 1.000 sequenze di

test, ognuna delle quali rappresenta una traiettoria regolarmente campionata di palline da biliardo all'interno di uno spazio rettangolare.

In ogni sequenza, le palline sono inizializzate con posizioni e velocità casuali e le traiettorie sono sviluppate per un totale di 200 passaggi temporali. Ogni pallina ha una dimensione fissa e una densità uniforme, e l'attrito è stato ignorato per semplificare l'esperimento.

Al fine di valutare l'efficacia di NRTSI, sono stati introdotti casualmente da 180 a 195 passaggi temporali mancanti in ciascuna traiettoria. Questo processo è stato ripetuto per un totale di 100 iterazioni. I risultati evidenziano che l'utilizzo di NRTSI ha portato a una riduzione della perdita dei dati del 64% rispetto al metodo di imputazione NAOMI, come possiamo notare dalla Figura 3.5 che riporta i risultati di suddetto esperimento.

Table 1: Quantitative comparison on Billiards dataset. Statistics closer to the expert indicate better performance.

Models	Linear	KNN	GRUI	MaskGAN	SingleRes	NAOMI	NRTSI	Expert
Sinuosity	1.121	1.469	1.859	1.095	1.019	1.006	**1.003**	1.000
step change ($\times 10^{-3}$)	**0.961**	24.59	28.19	15.35	9.290	7.239	5.621	1.588
reflection to wall	0.247	0.189	0.225	0.100	0.038	0.023	**0.021**	0.018
L2 loss ($\times 10^{-2}$)	19.00	5.381	20.57	1.830	0.233	0.067	**0.024**	0.000

Figura 3.5: Confronto dell'NTRSI con gli altri modelli di Data Imputation [21].

Quindi il modello NTRSI si presenta come una solida alternativa nei confronti di NAOMI e altri modelli di Data Imputation in diversi contesti. Come dimostrato dall'esperimento condotto e dai risultati riportati, l'NTRSI si distingue per la sua capacità di gestire efficacemente situazioni in cui è necessario imputare dati mancanti in Time Series caratterizzate da traiettorie complesse e campionate in modo irregolare.

Qualora però i dati presentano pattern temporali complessi e la struttura temporale diviene fondamentale per l'analisi, il modello *BRITS* potrebbe offrire prestazioni superiori grazie alla sua capacità di catturare le dipendenze temporali tramite reti neurali ricorrenti.

3.3.4 Bidirectional Recurrent Imputation for Time Series (BRITS)

Il modello BRITS (Bidirectional Recurrent Imputation for Time Series) è un'innovativa tecnica di Time Series Data Imputation che introduce un approccio bidirezionale per l'imputazione dei valori mancanti, utilizzando *Reti Neurali Ricorrenti (RNN)* e un approccio *Bayesiano* per ottenere previsioni accurate e incerte sui dati mancanti.

Anzichè utilizzare modelli classici che sfruttano la statistica per Time Series come ARMA (modello autoregressivo a media mobile) o ARIMA (integrato) si utilizza un modello che apprende direttamente i valori mancanti in una dinamica ricorrente bidirezionale (ossia cattura le relazioni temporali nei dati in entrambe le direzioni intese come passato e futuro), al fine di migliorare la capacità di previsione. Così facendo si ottengono risultati più accurati per la Data Imputation [22].

Si consideri una *Multivariate Time Series* (serie temporale multivariata) $X = \{x_1, x_2, \ldots, x_T\}$ come una sequenza di T osservazioni. L'osservazione t-esima $x_t \in \mathbb{R}^D$ consiste di D caratteristiche $\{x_{1t}, x_{2t}, \ldots, x_{Dt}\}$, ed è stata osservata al *timestamp*[6] s_t (il divario temporale tra i diversi timestamp potrebbe non essere uniforme). Nella realtà, a causa di eventi imprevisti, come per esempio danni all'attrezzatura o errori di comunicazione, x_t potrebbe presentare valori mancanti.

Per rappresentare i valori mancanti in x_t, si introduce un *Masking Vector* (vettore di mascheramento) m_t dove:

$$m_d^t = \begin{cases} 0 & \text{se } x_d^t \text{ non è osservato} \\ 1 & \text{altrimenti} \end{cases}$$

Il Masking Vector riveste un ruolo fondamentale nel processo di Data Imputation all'interno del modello BRITS. Durante la fase di addestramento del modello, il vettore di mascheramento viene impiegato per calcolare la perdita esclusivamente sui dati osservati, pertanto ignora i valori mancanti. Inoltre, durante il processo di inferenza[7], il vettore di mascheramento viene utilizzato come guida per generare previsioni relative ai valori mancanti.

Considerando la versione unidirezionale del BRITS, ovvero il RITS, gli errori dei valori mancanti stimati sono ritardati fino alla presenza dell'osservazione successiva. Tale ritardo degli errori fa sì che il modello converga lentamente e, di conseguenza, porti a un'inefficienza nell'addestramento. Nel frattempo, porta

[6] Marca temporale corrispondente.

[7] Rappresenta una fase cruciale in cui il modello applica le conoscenze apprese durante l'addestramento per fare previsioni su nuovi dati o per risolvere un problema specifico.

anche al problema del *Bias Exploding*[8], quindi gli errori commessi nelle prime fasi della previsione sequenziale vengono alimentati e passati come input al modello e possono essere rapidamente amplificati (figura 3.6).

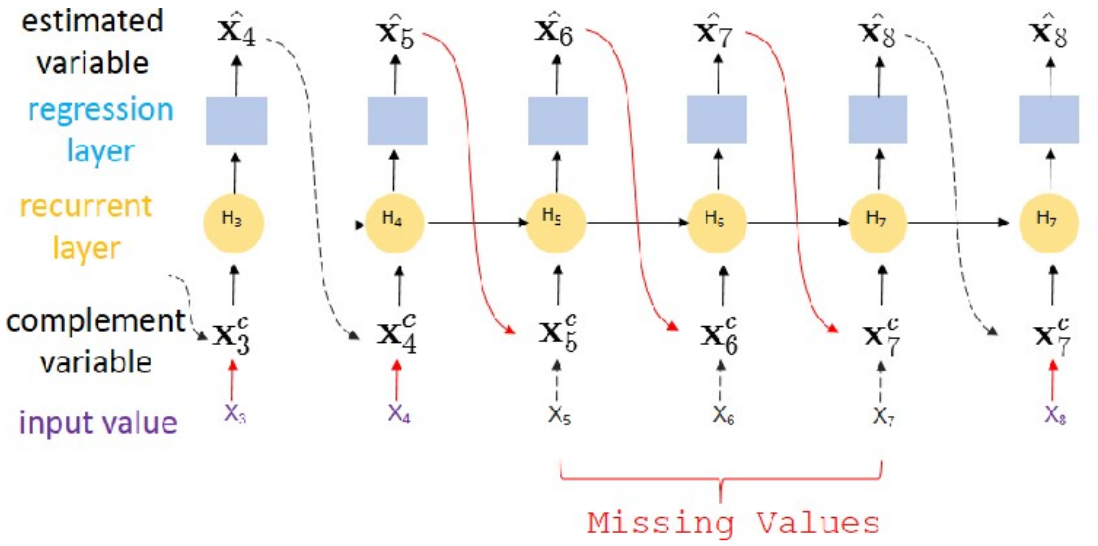

Figura 3.6: Esempio di Data Imputation con RITS [23].

Pertanto viene utilizzato il BRITS, che mediante il suo algoritmo attenua tale problema utilizzando le dinamiche ricorrenti bidirezionali sulla Time Series fornita, cioè oltre alla direzione in avanti, ogni valore nella Time Series può essere derivato anche dalla direzione inversa tramite un'altra funzione arbitraria fissa.

Nella pratica, l'algoritmo BRITS esegue il RITS nelle direzioni in avanti e all'indietro, rispettivamente. Nella direzione in avanti, otteniamo la sequenza di stime $\{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_T\}$ e la sequenza di perdita $\{\lambda_1, \lambda_2, \ldots, \lambda_T\}$. Allo stesso modo, nella direzione all'indietro, otteniamo un'altra sequenza di stime $\{\hat{x_1}^*, \hat{x_2}^*, \ldots, \hat{x}_T^*\}$ e un'altra sequenza di perdita $\{\lambda_1^*, \lambda_2^*, \ldots, \lambda_T^*\}$. Introduciamo un requisito il quale implichi che la previsione ad ogni passaggio sia coerente in entrambe le direzioni mediante l'introduzione dell'elemento *Consistency Loss* (perdita di coerenza) :

$$\ell_t^{cons} = \text{Discrepancy}(\hat{x}_t, \hat{x}_t^*)$$

[8]Si tratta di un fenomeno in cui gli errori accumulati nei passaggi precedenti di un modello di Machine Learning vengono amplificati man mano che il modello procede attraverso ulteriori passaggi temporali, compromettendo la qualità delle previsioni globali del modello.

Viene utilizzato nel BRITS anche l'Errore Medio Assoluto come *misura di discrepanza*[9]. La perdita finale di stima è ottenuta accumulando la perdita in avanti λ_t, la perdita all'indietro λ_t^* e la perdita di coerenza ℓ_t^{cons}. La stima finale al passaggio t è la media tra $\hat{x}_t$ e $\hat{x}_t^*$.

Per dimostrare l'efficienza del BRITS si è condotto un esperimento (il quale è riportato sull'articolo scientifico: *"BRITS: Bidirectional Recurrent Imputation for Time Series"* di Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li e Yitan Li), in cui viene confrontata l'efficacia dei metodi classici (definiti anche state-of-the-art) con l'approccio RITS e BRITS (dove quest'ultimo risulta il più performante), sottolineando la proficuità di tali modelli, poiché hanno dimostrato di ottenere risultati superiori. L'esperimento si è focalizzato sull'utilizzo di sensori indossati da cinque soggetti impegnati in attività quotidiane, come camminare, sedersi e sdraiarsi, per un totale di 11 attività. Questi sensori hanno raccolto coordinate tridimensionali, generando una serie temporale costituita da 30.917 valori. Durante l'esperimento, tramite Python, è stato casualmente selezionato e rimosso il 10% di questi dati al fine di valutare le prestazioni dei vari modelli di imputazione dei dati (figura 3.7).

Le prestazioni di Data Imputation sono valutate in termini di Errore Medio Assoluto (MAE) e Errore Medio Relativo (MRE), definendo label_i il valore reale dell'i-esimo elemento, pred_i l'output dell'i-esimo elemento e assumendo che ci siano in totale N elementi. Allora, MAE e MRE sono definiti come:

$$MAE = \frac{\sum_i |\text{pred}_i - \text{label}_i|}{N}, \quad MRE = \frac{\sum_i |\text{pred}_i - \text{label}_i|}{\sum_i |\text{label}_i|}$$

Method		Human Activity
Non-RNN	Mean	0.767 (96.43%)
	KNN	0.479 (58.54%)
	MF	0.879 (110.44%)
	MICE	0.477 (57.94%)
	STMVL	/
RNN	GRU-D	0.558 (70.05%)
	M-RNN	0.248 (31.19%)
Ours	RITS	0.248 (31.21%)
	BRITS	**0.219 (27.59%)**

Figura 3.7: Risultati dell'esperimento condotto [22].

In conseguenza di ciò, vista la sua efficacia, verrà utilizzato il BRITS nel lavoro di Tesi.

[9]Tale misura è utilizzata per quantificare l'errore tra i valori predetti e i valori osservati o veri.

Capitolo 4

Tecnologie Utilizzate

Di seguito un illustrazione degli strumenti adottati per la realizzazione del progetto alla base di questo di lavoro di Tesi:

- *Python*: linguaggio di programmazione.
- *Visual Studio Code*: ambiente di sviluppo software.
- *Teltonika FMC130*: per il rilevamento del movimento del corpo trattore-strumento.
- *Teltonika EYE BEACON e EYE Sensor*: sensore di rilevamento di movimento.
- *Traccar*: piattaforma di tracciamento GPS.

4.1 Python

Python è il linguaggio di programmazione che verrà utilizzato.
Consente di scrivere programmi per computer; la maggior parte dei linguaggi di programmazione è di tipo testuale, ma alcuni possono essere grafici. Si tratta di un linguaggio informatico (Figura 4.1).

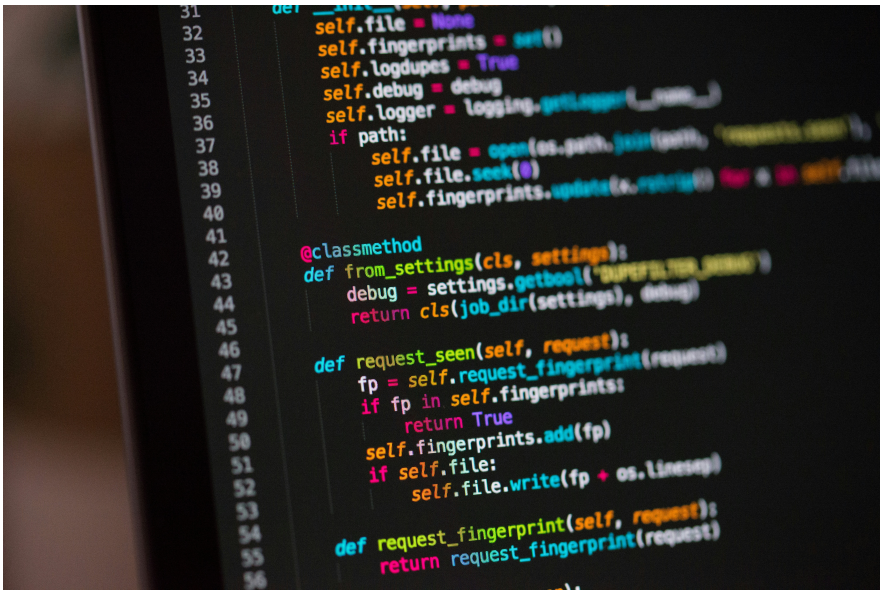

Figura 4.1: Linguaggio Python [11].

4.2 Visual Studio Code

Visual Studio Code (VS Code) è un editor di codice sorgente utilizzato per scrivere algoritmi in diversi linguaggi di programmazione. Dispone di una serie di funzionalità che si adattano al linguaggio di programmazione in uso. VS Code fornisce strumenti per il debugging e l'integrazione con sistemi di controllo di versione, semplificando lo sviluppo e il testing del software. Inoltre, consente di modificare il linguaggio di programmazione del documento in fase di modifica e di avviare il software.

Il servizio fornisce una macchina virtuale integrata e un ambiente già predisposto per l'esecuzione di codice (Figura 4.2).

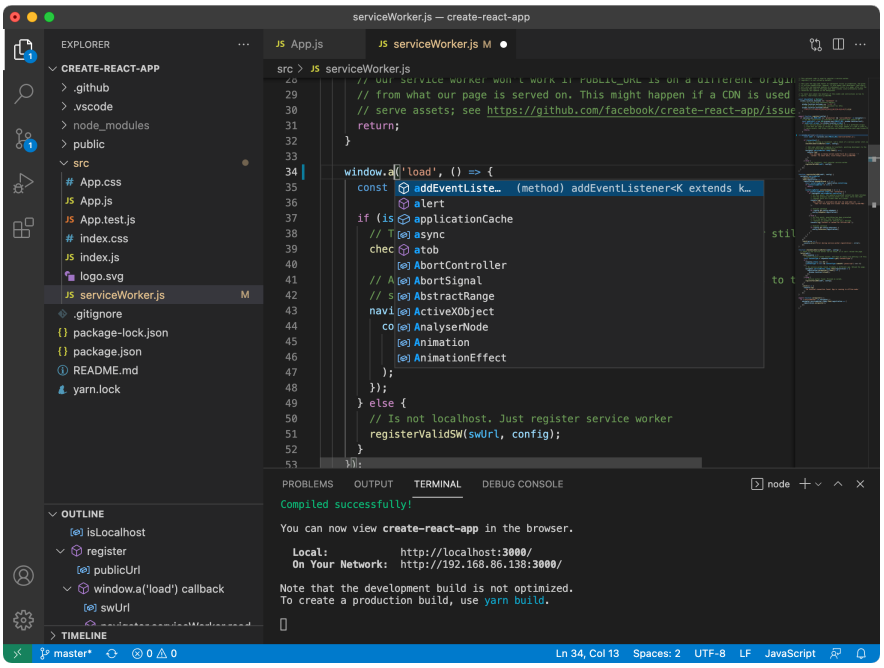

Figura 4.2: Interfaccia Visual Studio Code [11].

Il grande vantaggio consiste nella possibilità di eseguire le istruzioni su unità di elaborazione dedicate come GPU (Graphics Processing Unit) o acceleratori TPU(Tensor Processing Unit): unità progettata per l'esecuzione di operazioni tensoriali e ottimizzata nel consumo, fornisce tempi di computazione incredibilmente inferiori a calcolatori generici (come CPU).
Ció consente di svincolare l'esecuzione del codice dalla potenza di calcolo del proprio computer, migliorando notevolmente il tempo di addestramento della rete neurale.

VS Code nello sviluppo della Tesi è stato utilizzato per scrivere ed eseguire il codice che ci permetterà di effettuare la Data Imputation nelle Time Series, dopo aver costruito e configurato correttamente il Dataset composto dalle varie traiettorie dei trattori, raccolte tramite sensori GPS.

4.3 Teltonika FMC130

FMC130 rappresenta un dispositivo di tracciamento in tempo reale, compatto e professionale, dotato di connettività al *Global Navigation Satellite System* (ossia il Sistema Globale di Navigazione Satellitare, GNSS). Il GNSS è un sistema di geo-radiolocalizzazione e navigazione utilizzato su terra, mare o aria, che fa affidamento su una rete di satelliti artificiali in orbita e pseudoliti, oltre a connessioni LTE/3G/GSM e una batteria di backup.
Il dispositivo include moduli GNSS/Bluetooth/LTE, un modulo GNSS interno, antenne, ingressi digitali e analogici, uscite digitali configurabili, ingressi negativi e impulsi. È ideale per applicazioni che richiedono il monitoraggio remoto di oggetti, come la gestione di flotte, noleggio auto, servizi taxi, trasporti pubblici, logistica aziendale e veicoli personali (Figura 4.3).

Questo dispositivo permette di monitorare e tracciare gli altri sensori (Teltonika EYE BEACON e Telkonica Eye Sensor) Telkonica in tempo reale, quindi ci permette di rilevare la posizione geografica, velocità, direzione e altre informazioni relative al trattore e attrezzo.

Consente, poi, la creazione di aree geografiche definite (*Geofence*) per tracciare l'ingresso o l'uscita dei veicoli o delle risorse mobili da determinate zone permettendo di monitorare i confini geografici e ricevere notifiche quando un veicolo o una risorsa supera tali confini.

Il dispositivo offre diverse possibilità di connettività dati, tra cui GPRS[1], 3G, 4G LTE o connessione cablata, e viene impiegato per trasmettere i dati di localizzazione (posizione geografica, velocità, direzione) a un server o a un'applicazione di monitoraggio.

Figura 4.3: Dispositivo Teltonika FMC130.

[1] *General Packet Radio Service*, è una tecnologia di trasmissione dati utilizzata nei sistemi di telecomunicazioni mobili.

4.4 Teltonika Eye Beacon e Eye Sensor

Ideali per scenari di tracciamento, monitoraggio delle consegne e localizzazione di diversi oggetti mobili nel settore della logistica (quali rimorchi e contenitori), nell'ambito agricolo (ad esempio, attrezzi da trattore) e nel campo delle costruzioni (strumentazione e inventario). Inoltre, adatti all'uso interno come soluzioni di tracciamento per articoli all'interno di magazzini, strutture ospedaliere, hub di trasporto e altre aree industriali (Figura 4.4).

I dispositivi sono pienamente integrati con il firmware Teltonika, una piattaforma che offre una vasta gamma di funzionalità. La configurazione, la scansione e gli aggiornamenti possono essere effettuati in qualsiasi momento e da qualsiasi luogo tramite un'applicazione mobile dedicata Teltonika.

Questi dispositivi rappresentano gli strumenti essenziali per monitorare la posizione, la velocità e altre informazioni relative ai trattori e agli attrezzi associati.

Figura 4.4: Teltonika Eye Sensor.

4.5 Traccar

Traccar è un moderno sistema di tracciamento GPS gratuito e open source progettato per monitorare la posizione e l'attività di veicoli, dispositivi mobili e altri asset. Si tratta di un software che garantisce elevate prestazioni e stabilità.

Affinché Traccar funzioni correttamente e fornisca informazioni aggiornate sulla posizione e l'attività dei dispositivi di tracciamento, è necessario collegarlo a un server. A questo scopo, è essenziale configurare correttamente il server Traccar, che funge da punto centrale per la ricezione dei dati inviati dai dispositivi di tracciamento, elabora e archivia tali dati e fornisce un'interfaccia per la visualizzazione e l'amministrazione delle informazioni di tracciamento. In altre parole, il server Traccar costituisce il punto di connessione tra i dispositivi di tracciamento e il sistema di monitoraggio e gestione. Traccar offre anche un'interfaccia web per desktop e dispositivi mobili, nonché applicazioni mobili native per piattaforme Android e iOS. Inoltre, fornisce una suite di applicazioni in grado di trasformare i dispositivi mobili in localizzatori GPS.

Il sistema consente di visualizzare la posizione in tempo reale dei dispositivi GPS senza alcun ritardo e offre diverse opzioni di mappatura, incluse mappe stradali e immagini satellitari, permettendo di scegliere quella più adatta. Inoltre, è in grado di gestire in modo efficiente una vasta gamma di sensori e dati forniti dalle unità GPS (Figura 4.5).

Per il lavoro di Tesi, questo software viene utilizzato come sistema di tracciamento per i dati inviati dal dispositivo Teltonika FMC130, che monitora i sensori Eye Sensor.

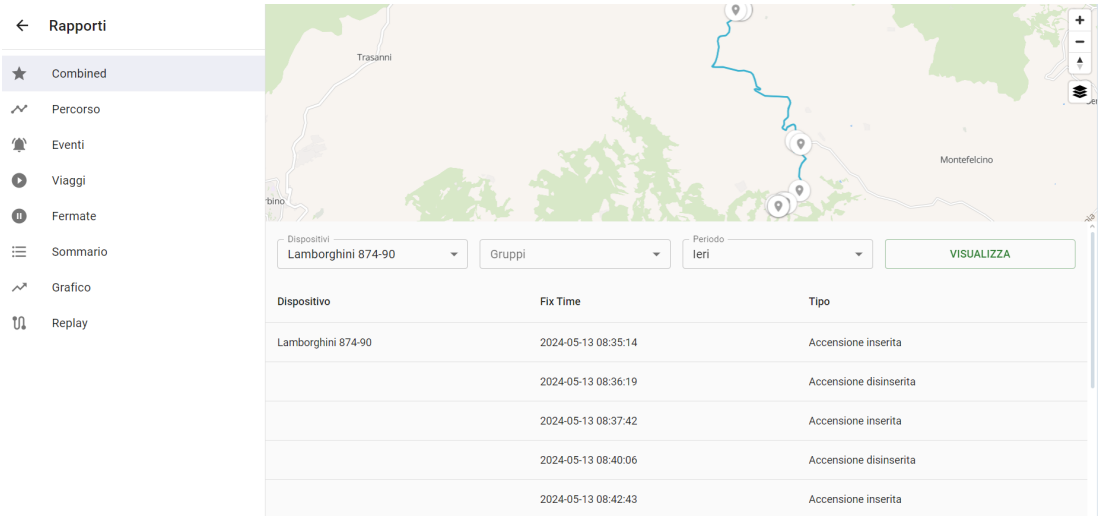

Figura 4.5: Sistema di tracciamento Traccar.

Capitolo 5

Sviluppo del Progetto e Realizzazione del Software

L'obiettivo di questo progetto è quello di sviluppare un' **applicazione di Intelligenza Artificiale per l'analisi di Time Series e Data Imputation nel contesto di Precision Farming**, utilizzando un dataset formato dai dati raccolti dal sistema GPS, che fornirà posizione, velocità e direzione del trattore.

Questi dati vengono archiviati sulla piattaforma Traccar, quindi è necessario estrapolarli sotto forma di file *.xlsx* o *.json*. Ogni documento contiene informazioni relative a un determinato trattore in un intervallo di tempo specifico, entrambi selezionabili tramite la piattaforma Traccar. In dettaglio, ogni documento riporta il modello del mezzo agricolo, il periodo di osservazione scelto, e per ciascuna misurazione temporale sono disponibili i seguenti dati: latitudine (in gradi), longitudine (in gradi), altitudine (in metri), velocità (in nodi). Inoltre, sono inclusi i dati relativi al conducente del mezzo, le ore di lavorazione totali, la carica della batteria e lo stato dell'accensione (espressi come valori booleani).

Per il lavoro di questa Tesi saranno considerati per costruire la Time Series solamente i valori relativi al tempo, latitudine e longitudine mentre per quanto riguarda l'estensione scelta verrà utilizzata quella in formato *xlsx*. L'obiettivo sarà quindi quello di formare un dataset utile all'addestramento del modello di Intelligenza Artificiale per la ricostruzione dei dati mancanti (Data Imputation). Dopo aver estratto i dati di interesse dal sistema di tracciamento GPS, verrà creata una Time Series da cui verrà rimossa artificialmente una certa percentuale di dati, in modo da addestrare il modello alla ricostruzione della Time Series utilizzando il modello BRITS.
Infine verranno creati dei grafici che mostreranno le varie traiettorie con i valori mancanti e le relative predizioni di queste mediante la Data Imputation eseguita dal modello.

5.1 Creazione del Dataset e Time Series

Nel codice sono state importate diverse librerie e moduli necessari per la generazione e la manipolazione dei dati. Le librerie utilizzate sono state *Pypots*, che fornisce strumenti avanzati per la generazione di dati, l'ottimizzazione e la stima di dati mancanti in serie temporali, *Pandas* che contiene vari metodi per manipolare i DataFrames, *Sklearn* che contiene alcuni metodi per la costruzione di Dataset per modelli di Machine Learning e *BRITS* che contiene il modello per la Data Imputation che dovremo utilizzare. Il processo inizia impostando un *seed* casuale per garantire la riproducibilità dei risultati.

Successivamente, viene generato un dataset applicando un tasso di dati mancanti del 10% mediante una funzione apposita che elabora il dataset iniziali.

All'inizio della funzione avviene l'estrazione dei dati da un file Excel specificato da un certo percorso di interesse. Utilizzando la libreria Pandas, i dati vengono letti e filtrati per includere solo le colonne "Time", "Latitude" e "Longitude". Successivamente, queste colonne vengono convertite in liste e organizzate in un dizionario (Figura 5.1).

```
44  link= r"C:\Users\Leo\OneDrive\Tirocinio\Report\report1.xlsx"
45  extract=pd.read_excel(link, header=7)
46  request=pd.DataFrame(extract,columns=["Time","Latitude","Longitude"])
47  Time=request["Time"].tolist()
48  Latitude=request["Latitude"].tolist()
49  Longitude=request["Longitude"].tolist()
50
51  data = {
52      "Latitude": [],"Longitude": [],"Time": []}
53  data["Latitude"]=Latitude
54  data["Longitude"]=Longitude
55  data["Time"]=Time
56
57  df = pd.DataFrame(data)
58  df["Time"] = pd.to_datetime(df["Time"])
59  df["RecordID"] = np.arange(1, len(df) + 1)
60
61  def expand_to_48_steps(df, record_id):
62      record_df = df[df["RecordID"] == record_id]
63      if len(record_df) < 48:
64          expanded_df = record_df.iloc[np.arange(48) % len(record_df)].copy()
65          expanded_df["Time"] = pd.date_range(start=record_df["Time"].iloc[0], periods=48, freq='h')
66          return expanded_df
67      else:
68          return record_df.iloc[:48]
69  expanded_df = pd.concat([expand_to_48_steps(df, record_id) for record_id in df["RecordID"].unique()])
70
```

Figura 5.1: Estrazione dei dati dal file .xlsx, predisposizione del dataset e creazione dataframe con conseguente espansione per raggiungere il numero minimo di passi temporali richiesti.

I dati estratti vengono organizzati in un DataFrame di Pandas. La colonna "Time" viene convertita in formato datetime, e viene aggiunta una colonna "Re-

cordID" per identificare ogni record. Per ogni RecordID, il dataset viene espanso a 48 passi temporali, se necessario, per garantire uniformità nella lunghezza delle serie temporali. Così facendo nel caso viene passato in input un dataset con meno di 48 passi temporali, questo viene completato generando dati artificiali.

Successivamente avviene un ridimensionamento dei dati (Figura 5.2) in modo tale che le proporzioni di addestramento, validazione e test rispettino determinati rapporti. Viene dunque determinata la dimensione massima di un dataset da suddividere in set di addestramento, validazione e test, rispettando proporzioni specificate (80%, 20%, 20% rispettivamente e considerando che il validation è del 20% rispetto al train) e verificando che tutte le dimensioni dei set siano divisibili per un valore dato (cioè divisor, 48 nel nostro caso).

```
80
81  def calculate_total_size(original_size, train_ratio, validation_ratio, test_ratio, divisor):
82      max_divisible_size = (original_size // divisor) * divisor
83
84      while max_divisible_size > 0:
85          train_size = int(train_ratio * max_divisible_size)
86          test_size = int(test_ratio * max_divisible_size)
87          validation_size = int(train_ratio * validation_ratio * max_divisible_size)
88
89          if train_size % divisor == 0 and test_size % divisor == 0 and validation_size % divisor == 0:
90              return max_divisible_size
91          max_divisible_size -= divisor
92      raise ValueError("Non è possibile trovare una dimensione adatta.")
93
94  train_ratio = 0.8
95  validation_ratio = 0.2
96  test_ratio = 0.2
97  divisor = 48
98
99  total_size = calculate_total_size(len(df), train_ratio, validation_ratio, test_ratio, divisor)
100 df = df[:total_size]
```

Figura 5.2: Ridimensionamento dei dati secondo determinate proporzioni.

Il dataset viene diviso in set di addestramento, validazione e test utilizzando la funzione *TrainTestSplit* della libreria Sklearn (Figura 5.3). Gli ID dei record vengono suddivisi con una proporzione che varia a seconda della dimensione del DataFrame, successivamente, il set di addestramento viene ulteriormente diviso in set di addestramento e validazione.

A seguire le colonne "RecordID" e "Time" vengono rimosse dai set di dati per mantenere solo le caratteristiche rilevanti per l'addestramento del modello BRITS, ovvero latitudine e longitudine.

```
all_recordID = expanded_df["RecordID"].unique()
split_value = len(df)/100*10

train_set_ids, test_set_ids = train_test_split(df, test_size=(split_value/len(df)))
train_set_ids, val_set_ids = train_test_split(train_set_ids, test_size=((split_value*2)/len(df)))

train_set = train_set_ids.sort_values("Time")
val_set = val_set_ids.sort_values("Time")
test_set = test_set_ids.sort_values("Time")

train_set = train_set.drop(["RecordID", "Time"], axis=1)
val_set = val_set.drop(["RecordID", "Time"], axis=1)
test_set = test_set.drop(["RecordID", "Time"], axis=1)

train_X, val_X, test_X = (
    train_set.to_numpy(),
    val_set.to_numpy(),
    test_set.to_numpy(),
)
```

Figura 5.3: Divisione del dataset in set di addestramento, convalida e test.

Successivamente si passa alla fase di normalizzazione dei dati (Figura 5.4), quindi vengono normalizzati utilizzando *StandardScaler*, un processo che standardizza le caratteristiche del set di addestramento per avere media zero e deviazione standard unitaria. Le stesse trasformazioni di standardizzazione vengono applicate anche ai set di validazione e test.

```
scaler = StandardScaler()
train_X = scaler.fit_transform(train_X)
val_X = scaler.transform(val_X)
test_X = scaler.transform(test_X)
train_X = train_X.reshape(len(train_set_ids)//48, n_steps, -1)
val_X = val_X.reshape(len(val_set_ids)//48, n_steps, -1)
test_X = test_X.reshape(len(test_set_ids)//48, n_steps, -1)
```

Figura 5.4: Normalizzazione dei dati.

Dopo aver normalizzato i dati, viene effettuata la conversione dei dati. In sintesi i dati vengono rimodellati per adattarsi all'addestramento del modello di Time Series, assumendo una forma definitiva formata da: numero di campioni (*Samples*[1]), lunghezza della Time Series (*Timesteps*[2]), numero di caratteristiche per passo temporale (*Features per Timestep*[3]).

Viene creato il dizionario contenente tutte le informazioni utili ad addestrare il modello BRITS successivamente avviene il mascheramento artificioso dei dati specificato dalla variabile *ArtificiallyMissingRate* $\in$ [0,1).

La funzione restituisce un dizionario contenente tutte le informazioni necessarie per l'addestramento del modello (Figura 5.5). Questo include il numero di passi temporali, il numero di caratteristiche, i set di addestramento, validazione e test, e lo scaler utilizzato per la normalizzazione. Inoltre, se sono stati introdotti valori mancanti, il dizionario contiene sia i dati con i valori mancanti che quelli originali per la validazione e il test.

```
134    data = {
135        "n_steps": n_steps,
136        "n_features": train_X.shape[-1],
137        "train_X": train_X,
138        "val_X": val_X,
139        "test_X": test_X,
140        "scaler": scaler,
141    }
142
143    if artificially_missing_rate > 0:
144        def mcar(data, rate):
145            mask = np.random.binomial(1, rate, size=data.shape).astype(bool)
146            data[mask] = np.nan
147            return data
148
149        def mcar_numpy(
150            X: np.ndarray,
151            p: float,
152        ) -> np.ndarray:
153            assert 0 < p < 1, f"p must be in range (0, 1), but got {p}"
154
155
156            X = np.copy(X)
157            mcar_missing_mask = np.asarray(np.random.rand(np.prod(X.shape)) < p)
158            mcar_missing_mask = mcar_missing_mask.reshape(X.shape)
159            X[mcar_missing_mask] = np.nan
160            return X
```

Figura 5.5: Predisposizione e assemblaggio del dictionary di Python contenente i valori atti a costituire il modello di Data Imputation.

[1]Il numero di campioni si riferisce al numero totale di osservazioni o sequenze di dati indipendenti presenti nel dataset.

[2]La lunghezza della Time Series è il numero di punti temporali o osservazioni all'interno di ciascun campione.

[3]Il numero di caratteristiche per ciascun passo temporale rappresenta il numero di variabili o attributi misurati ad ogni istante temporale.

Se previsto, viene introdotto un tasso di valori mancanti attraverso la parametrizzazione del valore di maschera dei dati, simulando situazioni reali di dati mancanti (Figura 5.6).

```
162         val_X_ori = val_X
163         val_X = mcar_numpy(val_X, artificially_missing_rate)
164         test_X_ori = test_X
165         test_X = mcar_numpy(test_X, artificially_missing_rate)
166 
167         data["val_X"] = val_X
168         data["val_X_ori"] = val_X_ori
169         data["test_X"] = test_X
170         data["test_X_ori"] = np.nan_to_num(test_X_ori)
171         data["test_X_indicating_mask"] = np.isnan(test_X_ori) ^ np.isnan(test_X)
172     return data
```

Figura 5.6: Mascheramento dei dati.

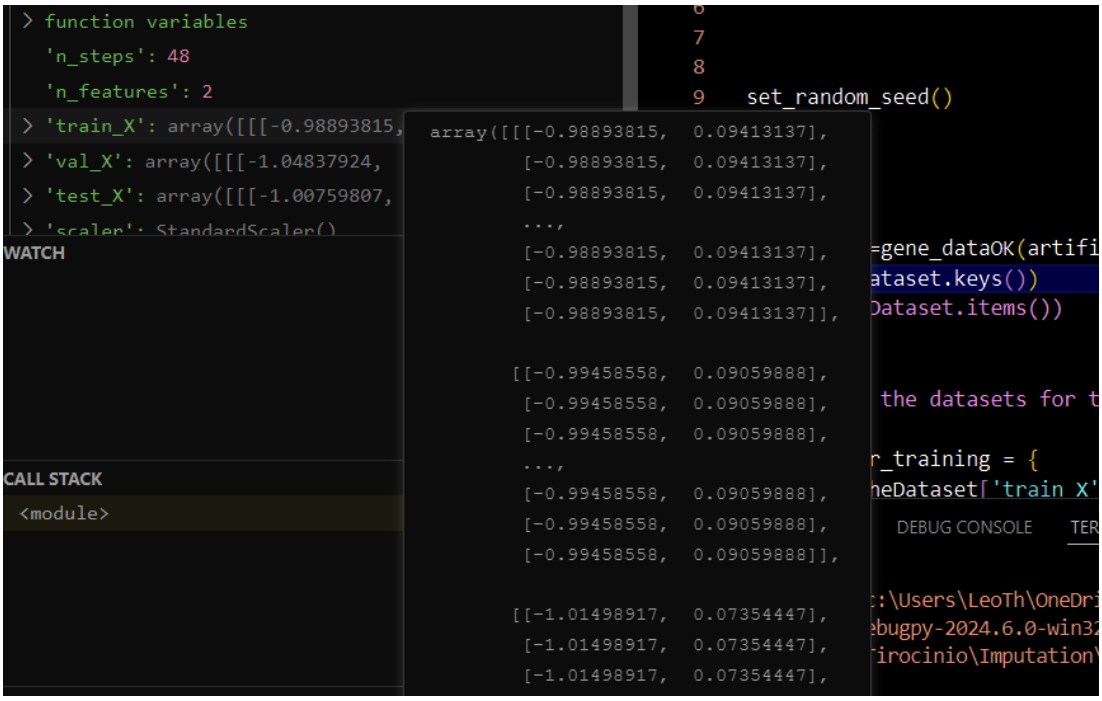

Figura 5.7: Dataset formato per l'addestramento del modello BRITS.

5.2 Data Imputation

Formato il dizionario questo viene assemblato e convalidato per potere essere utilizzato nel modello BRITS per quanto riguarda l'addestramento, la validazione e il testing. Quindi dopo aver fatto ciò si inizializza anche il modello BRITS: vengono specificati diversi iperparametri, come la dimensione nascosta della RNN, la dimensione del batch, il numero di epoche e il tasso di apprendimento dell'ottimizzatore. Inoltre, vengono definiti parametri per il salvataggio dei risultati e la gestione del modello durante l'addestramento, come la pazienza per l'Early Stopping (nel caso si ottengono risultati soddisfacenti) e la strategia di salvataggio del modello. Questo modello verrà utilizzato per addestrare e valutare la Data Imputation nella Time Series fornita dal dataset (Figura 5.8).

```
183  dataset_for_training = {
184      "X": data['train_X'],
185  }
186
187  dataset_for_validating = {
188      "X": data['val_X'],
189      "X_ori": data['val_X_ori'],
190  }
191
192  dataset_for_testing = {
193      "X": data['test_X'],
194  }
195
196  brits = BRITS(
197      n_steps=data['n_steps'],
198      n_features=data['n_features'],
199      rnn_hidden_size=128,
200      batch_size=32,
201      epochs=200,
202      patience=3,
203      optimizer=Adam(lr=1e-3),
204      num_workers=0,
205      device=None,
206      saving_path="results/imputation/brits",
207      model_saving_strategy="best",
208      verbose=True
209  )
```

Figura 5.8: Assemblaggio del dataset per la Data Imputation e inizializzazione del modello BRITS.

Successivamente viene verificato il valore booleano della variabile **train** (Figura 5.9), se *true* allora viene attivato il modello BRITS per effettuare l'addestramento sul set di addestramento e validato sul set di convalida per selezionare il miglior modello per il successivo testing. A seguire, il modello viene utilizzato per imputare i valori mancanti sia di quelli originalmente mancanti che di quelli artificialmente mancanti nel set di test. Verranno inoltre stampati a schermo grafici contenenti le rotte formate dalle Time Series con i vari dati mascherati e le relative predizioni effettuate dalla Data Imputation del BRITS.

Nel caso il valore della variabile **train** sia *false* allora viene utilizzato un modello di Data Imputation già salvato nel path specificato.

Infine, vengono calcolati i seguenti indici, nonchè le metriche di valutazione utilizzate:

1. **Mean Absolute Error (MAE)** rappresenta l'errore medio assoluto sui valori artificialmente mancanti, considerati come *Ground Truth* (nonchè i dati di riferimento reali e completi prima dell'introduzione dei dati mancanti artificiali), per valutare le prestazioni del modello.

2. **Mean Square Error (MSE)** rappresenta la media dei quadrati degli errori o delle deviazioni; fornisce un modo per misurare la qualità di un estimatore riflettendo sia la varianza che il bias[4].

3. **Root Mean Squared Error (RMSE)** è la Radice dell'Errore Quadratico Medio che fornisce un'indicazione della grandezza media degli errori di previsione, quantificando così la varianza dell'errore.

4. **Mean Relative Error (MRE)** indicatore che mostra la media delle differenze assolute tra i valori previsti e quelli effettivi, normalizzate per i valori effettivi. Può rivelarsi utile per confrontare gli errori su diverse scale.

[4]Il bias, in contesto di modelli statistici e di Machine Learning, rappresenta la differenza sistematica tra le previsioni del modello e i valori reali.

```
if train:
    brits.fit(train_set=dataset_for_training, val_set=dataset_for_validating)
    brits_results = brits.predict(dataset_for_testing)
    brits_imputation = brits_results["imputation"]
    testing_mae = calc_mae(
        brits_imputation,
        data['test_X_ori'],
        data['test_X_indicating_mask'],
    )

    testing_mse = calc_mse(
        brits_imputation,
        data['test_X_ori'],
        data['test_X_indicating_mask'],
    )

    testing_rmse = calc_rmse(
        brits_imputation,
        data['test_X_ori'],
        data['test_X_indicating_mask'],
    )

    testing_mre = calc_mre(
        brits_imputation,
        data['test_X_ori'],
        data['test_X_indicating_mask'],
    )
    print(f"Testing mean absolute error: {testing_mae:.4f}")
    print(f"Testing mean square error: {testing_mse:.4f}")
    print(f"Testing Root Mean Square Error: {testing_rmse:.4f}")
    print(f"Testing mean relative error: {testing_mre:.4f}")
    imputed_data = brits_imputation
else:
    model_path = "./results/imputation/brits/20240704_T161744/BRITS.pypots"
    brits.load(model_path)
    imputed_data= brits.impute(dataset_for_testing)
plot_data(data['test_X_ori'], data['test_X'],imputed_data, n_rows=2, n_cols=2, sample_idx=10)
plt.show(block=True)
```

Figura 5.9: Imputazione dei dati effettuata dall'addestramento di un nuovo modello o da un modello persistente.

Oltre alla conferma dell'inizializzazione del modello con gli iperparametri specificati e il numero totale di parametri addestrabili, vengono forniti i dettagli relativi all'andamento della perdita dei dati durante ciascuna epoca di addestramento e convalida (Figura 5.10), mentre nella Tabella 5.1 sono contenuti i risultati ottenuti dal modello. La perdita durante l'addestramento e la convalida viene riportata per ciascuna epoca, consentendo una valutazione della convergenza del modello. Va inoltre osservato che il numero di epoche è esiguo perchè il training è stato eseguito su un dataset di 500 traiettorie di 5 trattori agricoli. Infine, viene evidenziata l'epoca che ha prodotto il miglior modello in base alla perdita di convalida delle Time Series (tale fase è nota come fine-tuning del modello). Queste informazioni divengono preziose nel monitorare e valutare l'addestramento del modello, nonché per identificare il modello migliore da utilizzare per le fasi successive di testing e valutazione riguardo la Data Imputation.

```
2024-07-22 19:11:08 [INFO]: BRITS initialized with the given hyperparameters, the number of trainable parameters: 138,544
2024-07-22 19:11:11 [INFO]: Epoch 001 - training loss: 1.5699, validation loss: 0.6664
2024-07-22 19:11:12 [INFO]: Epoch 002 - training loss: 1.6489, validation loss: 0.6267
2024-07-22 19:11:13 [INFO]: Epoch 003 - training loss: 1.3672, validation loss: 0.5801
2024-07-22 19:11:14 [INFO]: Epoch 004 - training loss: 1.3367, validation loss: 0.5163
2024-07-22 19:11:15 [INFO]: Epoch 005 - training loss: 1.1940, validation loss: 0.4505
2024-07-22 19:11:15 [INFO]: Epoch 006 - training loss: 1.2354, validation loss: 0.4377
2024-07-22 19:11:16 [INFO]: Epoch 007 - training loss: 1.3047, validation loss: 0.4400
2024-07-22 19:11:17 [INFO]: Epoch 008 - training loss: 1.3419, validation loss: 0.4170
2024-07-22 19:11:18 [INFO]: Epoch 009 - training loss: 1.2118, validation loss: 0.3623
2024-07-22 19:11:19 [INFO]: Epoch 010 - training loss: 1.0672, validation loss: 0.3033
2024-07-22 19:11:19 [INFO]: Finished training. The best model is from epoch#10.
2024-07-22 19:11:19 [INFO]: Saved the model to C:\Users\LeoTh\OneDrive\Desktop\Università\Tirocinio\RisultatiBrits\20240722_T191108\BRITS.pypo
ts
Testing mean absolute error: 0.4228
Testing mean square error: 0.4374
Testing Root Mean Square Error: 0.6614
Testing mean relative error: 0.5591
Backend TkAgg is interactive backend. Turning interactive mode on.
```

Figura 5.10: Addestramento del modello con i relativi dati per ogni epoca.

Indice	**Valore**
Mean Absolute Error	0.4228
Mean Square Error	0.4374
Root Mean Square Error	0.6614
Mean Relative Error	0.5591

Tabella 5.1: Risultati del modello.

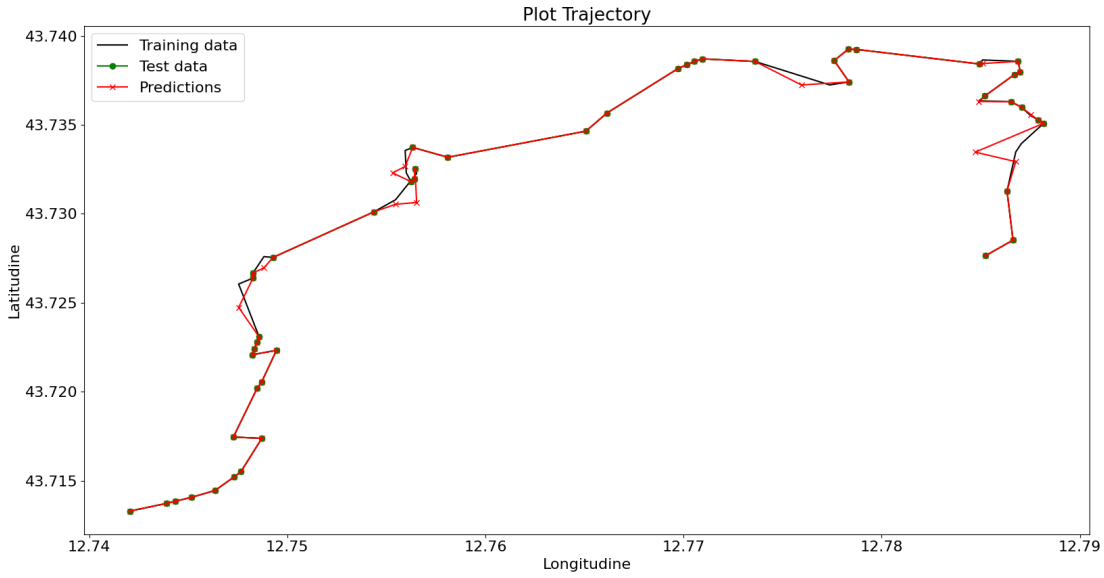

Figura 5.11: Grafico della traiettoria di un trattore preso in esame con i dati mancanti e le relative predizioni (su larga scala).

Dalle Figure 5.11 e 5.12 è possibile notare come il modello di Data Imputation ricostruisce la traiettoria nonostante il 10% dei dati della traiettoria mancanti. Maggiori sono gli addestramenti del modello e maggiore la precisione della predizione è elevata, fornendo dunque delle stime accettabili per la ricostruzione dei dati mancanti.

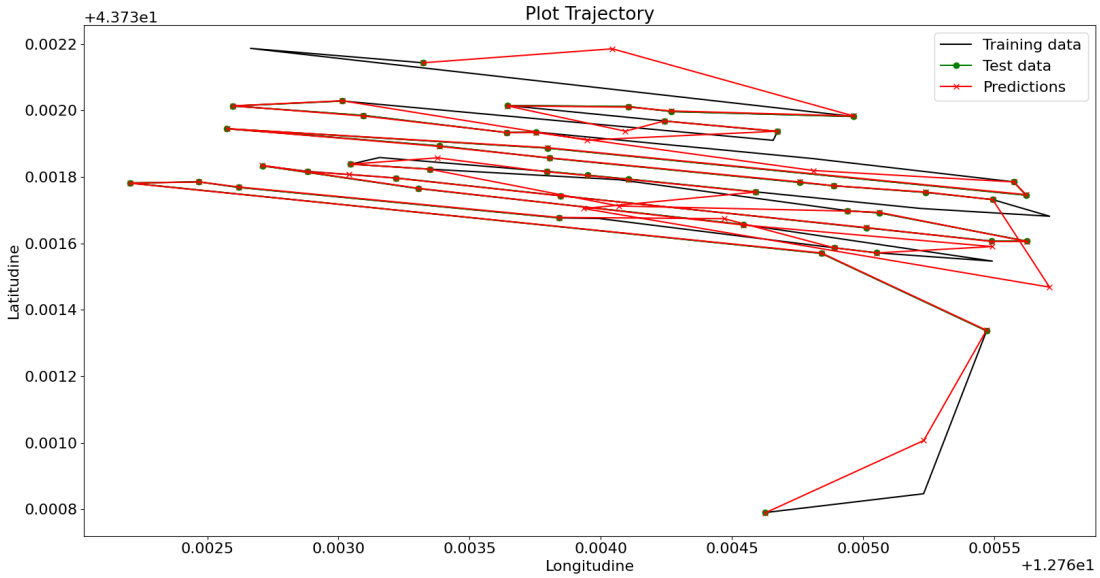

Figura 5.12: Grafico aggiuntivo della traiettoria di un altro trattore analizzato, che mostra i dati mancanti e le corrispondenti predizioni (su piccola scala).

Concludendo è possibile affermare che il modello sviluppato rappresenta un avanzamento significativo per quanto riguarda la Data Imputation per le Time Series dimostrandosi uno strumento efficace nell'affrontare le sfide legate ai dati mancanti, e tale progresso è rilevante per i dati raccolti dai sensori IoT nell'ambito dell'AdP. Mediante l'applicazione del modello BRITS, è stato possibile migliorare la qualità e la completezza dei dati, permettendo così di effettuare analisi accurate e decisioni più consapevoli.

Il modello sviluppato può essere ulteriormente migliorato aumentando il numero di parametri raccolti dai sensori IoT che compongono la Time Series, come ad esempio l'altitudine o la velocità. Inoltre, il lavoro presentato può essere applicato in vari altri settori o situazioni che richiedono la Data Imputation, come ad esempio nel monitoraggio delle condizioni ambientali, nella gestione delle risorse energetiche, nell'analisi delle performance finanziarie.

Capitolo 6

Conclusioni

Concludiamo questo lavoro di tesi ricordando come l'agricoltura 4.0 rappresenti un'evoluzione imprescindibile per aumentare l'efficienza, la sostenibilità e la qualità dei prodotti agricoli. Rimane comunque essenziale dover essere sempre pronti ad affrontare qualsiasi sfida legata al problema della perdita dei dati, e sicuramente la Data Imputation nelle Time Series sono un ottima soluzione per affrontare tali situazioni.

Attraverso tali innovazioni, l'agricoltura può evolvere in modo sempre più moderno ed efficiente. Tuttavia, è fondamentale che tali progressi siano accompagnati da una formazione adeguata e da una cultura digitale diffusa tra gli operatori del settore per sfruttarne appieno il potenziale che l'AdP può offrire.

6.1 Sviluppi futuri

Considerando la situazione globale attuale e le ricerche condotte in questo campo è possibile affermare che in futuro i modelli di Intelligenza Artificiale associati al mondo dell'AdP con il supporto di una efficiente Data Imputation di Time Series potranno riguardare:

1. *Ottimizzazione delle risorse agricole*: Miglior utilizzo dell'acqua, fertilizzanti e pesticidi attraverso l'analisi predittiva dei dati.

2. *Monitoraggio delle colture in tempo reale*: Implementazione di sistemi avanzati per rilevare anomalie e malattie nelle coltivazioni, consentendo interventi tempestivi.

3. *Miglioramento della resa agricola*: Previsione delle condizioni ottimali per la semina e il raccolto, aumentando l'efficienza e la produttività.

4. *Gestione delle condizioni climatiche*: Sviluppo di modelli in grado di prevedere l'impatto delle variazioni climatiche sulle colture, permettendo agli agricoltori di adattarsi rapidamente.

5. *Automazione e robotica agricola*: Integrazione dei modelli di machine learning con sistemi automatizzati e robotici per la gestione e la cura delle coltivazioni.

6. *Sostenibilità ambientale*: Utilizzo di modelli per promuovere pratiche agricole sostenibili, riducendo l'impatto ambientale e preservando le risorse naturali.

7. *Personalizzazione delle pratiche agricole*: Creazione di soluzioni su misura per le esigenze specifiche di ogni azienda agricola, basate su analisi dettagliate dei dati raccolti.

Tali prospettive indicano un futuro in cui l'agricoltura sarà sempre più smart, sostenibile ed efficiente, grazie ai continui progressi nei modelli di machine learning e anche grazie alle tecniche di Data Imputation delle Time Series, inoltre si estenderà l'algoritmo di AI utilizzando modelli di deep learning più aggiornati e potenti, utilizzando dataset più grandi rispetto a quello utilizzato in questo lavoro di tesi.

Bibliografia

[1] United Nations Department of Economic and Social Affairs Population Division. World population prospects 2019. https://www.un-ilibrary.org/content/books/9789210042352.

[2] Coldiretti. Giornata della terra: l'italia ha perso 1/3 delle campagne. https://www.coldiretti.it/economia/giornata-della-terra-litalia-ha-perso-1-3-delle-campagne.

[3] United Nations Department of Economic and Social Affairs Population Division. World population prospects 2022. https://population.un.org/wpp/ .

[4] Camera dei Deputati. Agricoltura di precisione. https://temi.camera.it/leg18/temi/agricoltura-di-precisione.html.

[5] Ministero dell'agricoltura della sovranità alimentare e delle foreste. Linee guida per lo sviluppo dell'agricoltura di precisione in italia. https://www.politicheagricole.it/flex/cm/pages/ServeBLOB.php/L/IT/IDPagina/12069 .

[6] David Franzen David Mulla. Precision agriculture technology for crop farming - a history of precision agriculture(1° capitolo). https://library.oapen.org/handle/20.500.12657/41698 .

[7] Valtra. Smart farming. https://www.valtra.it/blog/tecnologia/the-top-ten-benefits-of-smart-farming.html.

[8] Institute of Electrical and Electronics Engineers. Agriculture 5.0: Cutting-edge technologies, trends, and challenges. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10471250 .

[9] Ceresagritech. Agriculture 5.0: Unleashing ai. https://www.ceresagritech.org/projects/agriculture-5-0-unleashing-ai/.

[10] Environmental System Research Institute(ESRI). What is gis. https://www.esri.com/it-it/what-is-gis/overview .

[11] UnSplash. Foto. https://unsplash.com/it.

[12] Oracle. What is big data. https://www.oracle.com/it/big-data/what-is-big-data/ .

[13] Bigdata4innovation. Big data, cosa sono e come sono utili. https://www.bigdata4innovation.it/big-data.

[14] Cubic Telecom. The impact of big data analytics on precision farming. https://www.cubictelecom.com/blog/big-data-modern-precision-farming-agtech/ .

[15] Andreas Kamilaris Andreas Kartakoullis Francesc X. Prenafeta-Boldú. A review on the practice of big data analysis in agriculture. https://www.sciencedirect.com/science/article/pii/S0168169917301230 .

[16] Chenguang Fang Chen Wang. Time series data imputation: A survey on deep learning approaches.

[17] ISTAT. Popolazione residente in italia 1982-2009.

[18] Nkschols. Time-series-imputation. https://github.com/nkschlos/time-series-imputation.

[19] Towards Data Science. Implementation of rnn, lstm, and gru. https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090.

[20] Yonghong Luo Xiangrui Cai Ying Zhang Jun Xu Yuan xiaojie. Multivariate time series imputation with generative adversarial networks.

[21] Siyuan Shan Yang Li Junier B. Oliva. Nrtsi: Non-recurrent time series imputation.

[22] Wei Cao Dong Wang Jian Li Hao Zhou Lei Li Yitan Li. Brits: Bidirectional recurrent imputation for time series.

[23] Chen Wang Chenguang Fang. Time series data imputation: A survey on deep learning approaches.

[24] Camera dei Deputati Servizio Studi. Agricoltura di precisione. https://www.camera.it/temiap/documentazione/temi/pdf/1338314.pdf .

[25] AgriFood. Agricoltura 4.0: cos'è, incentivi e tecnologie abilitanti. https://www.agrifood.tech/digital-farming/agricoltura-4-0-cose-incentivi-e-tecnologie-abilitanti/ .

[26] Emanuele Leoni. Agricoltura di precisione, cos'è e come può aiutare a risolvere le sfide alimentari del futuro.

Elenco delle figure

Ringraziamenti

Voglio ringraziare e dedicare questo lavoro di Tesi a tutti coloro che in questi anni di Università mi sono stati accanto e hanno creduto in me, dandomi la forza e la volontà di andare avanti e non arrendermi.

Prima di tutto la mia famiglia che mi ha sempre sostenuto senza dubitare delle mie capacità, per questo non smetterò mai di volervi bene.

Poi voglio ringraziare tutti i miei amici, sia coloro che conosco da molti anni che quelli conosciuti durante il percorso universitario, è sicuramente grazie a voi che questi anni sono stati più piacevoli tra momenti belli passati assieme e le varie avventure passate all'interno dell'Università, non mi dimenticherò mai di tutto ciò.

Infine voglio ringraziare il Dott. Galdelli e il Prof. Mancini che mi hanno permesso di svolgere questa importante esperienza di Tirocinio e mi hanno accompagnato per questo tratto conclusivo facendomi svolgere un lavoro di peso.

A tutti voi vanno i miei più sentiti ringraziamenti, perchè siamo noi che dobbiamo impegnarci in prima persona ma se circondati dalle giuste persone si può raggiungere qualunque obbiettivo con il sorriso, qualsiasi sia la sua difficoltà.