UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Architetture serverless per la predizione di serie temporali a partire da materiali self-sensing nell'ambito del progetto ReCity

Serverless architectures for time series prediction from self-sensing materials within the ReCity project

Relatore:
Ch.Mo. Prof. Mancini Adriano
Correlatore:
Dr. Galdelli Alessandro

Laureando:
Marselletti Andrea

Anno Accademico 2022-2023

Indice

Capitolo 1

Introduzione

Il presente studio si inserisce nel contesto ambizioso del progetto ReCity, focalizzandosi sull'implementazione di un sistema avanzato e dinamico attraverso l'utilizzo del Serverless Computing. L'obiettivo principale è sviluppare un servizio Cloud che non solo archivi ed esamini dati con efficacia, ma che sia anche capace di predire in modo accurato l'andamento di serie temporali, addestrando una Rete Neurale con dati autogenerati attraverso materiali self-sensing.

La progettazione del sistema mira a superare le limitazioni tradizionali, proponendo un approccio innovativo e flessibile grazie all'utilizzo di tecnologie all'avanguardia. Il Serverless Computing, fondamentale per il nostro modello, consente una scalabilità automatica delle risorse, adattandosi dinamicamente alla domanda, e sposta la responsabilità di affidabilità e tolleranza ai guasti al provider di servizi Cloud. Questa infrastruttura permette agli sviluppatori di concentrarsi esclusivamente sulla logica dell'applicazione, liberandoli dalla gestione dettagliata dell'infrastruttura sottostante e riducendo notevolmente i costi operativi.

Il sistema sviluppato offre funzionalità accessibili tramite API (Application Programming Interface), permettendo agli utenti di interagire direttamente con le risorse e le capacità del sistema. Per garantire la massima affidabilità e completezza, ci affidiamo ai servizi di Amazon Web Services (AWS), leader mondiale nel settore dei servizi cloud.

L'innovazione nell'ambito del Serverless Computing ha radicalmente trasformato il modo in cui concepiamo lo sviluppo e l'implementazione delle applicazioni. L'approccio proposto non solo mira a superare le sfide tradizionali legate ai server dedicati, ma a ridefinire il panorama delle previsioni di serie temporali, integrando la potenza computazionale del cloud in modo agile e scalabile. Questo studio si pone come contributo significativo a un futuro in cui la predizione di dati temporali si fonda su sistemi avanzati, flessibili e all'avanguardia.

1.1 Obiettivi

Questo progetto di tesi si propone di delineare un sistema all'avanguardia basato su Serverless Computing, con l'obiettivo di ottimizzare l'acquisizione e l'inserimento efficiente dei dati all'interno di un database. Tale processo, noto come data ingestion, sarà orchestrato attraverso l'utilizzo sinergico di servizi Cloud.

Un passaggio cruciale di questa ricerca è dedicato all'addestramento di una Rete Neurale per la predizione di dati time series. Per affrontare questa sfida, abbiamo adottato il modello NeuralProphet [1], implementato in Python per offrire un approccio sia semplice che efficace nella creazione di modelli predittivi basati su Reti Neurali. La fase di sviluppo prevede la creazione di un'immagine Docker, che sarà successivamente resa disponibile come servizio tramite un endpoint grazie all'utilizzo di un servizio Cloud specializzato nella gestione e distribuzione di API, ossia l'API Gateway di Amazon.
In sintesi, il flusso di lavoro delineato comprende:

- L'utilizzo del servizio Amazon S3 per la creazione di un bucket nel quale sarà caricato il file excel contenente i dati.
- L'utilizzo del servizio Amazon ECR per la creazione della nostra repository per l'immagine docker.
- L'implementazione di Lambda Functions per facilitare l'inserimento ed estrazione dei dati, riferiti al contesto specifico di ReCity, in un determinato intervallo temporale.
- La creazione di un'immagine Docker che racchiude l'intero codice relativo alla progettazione e all'addestramento della Rete Neurale.
- La messa a disposizione di queste funzioni attraverso le API, sfruttando l'efficace servizio Amazon API Gateway.

La scelta del linguaggio di programmazione è ricaduta su Python, un linguaggio ad alto livello e multi-paradigma, selezionato per la sua versatilità e adattabilità ai diversi contesti applicativi.

1.2 Struttura della tesi

Nel contesto sempre più dinamico e competitivo delle tecnologie dell'informazione, l'adozione di soluzioni avanzate e innovative è diventata fondamentale per mantenere un vantaggio competitivo. La ricerca si colloca all'incrocio tra due ambiti di grande rilevanza: il Cloud Computing e l'analisi delle serie temporali. In questo contesto, è stato esplorato approfonditamente le potenzialità offerte dai servizi di AWS e sono state integrate metodologie all'avanguardia, tra cui l'uso di serie temporali e modelli predittivi basati su reti neurali.
Il lavoro di tesi viene strutturato come segue:

- **Il secondo capitolo** della tesi è dedicato a una disamina approfondita del paradigma di *cloud computing* e della sua estensione, il *serverless computing*. Si esploreranno le fondamenta concettuali e le implicazioni pratiche di questi approcci innovativi nell'ambito dell'informatica e dello sviluppo di applicazioni.

  Si inizierà con una panoramica completa del cloud computing, delineando i concetti chiave di **Platform as a Service (PaaS)**, **Infrastructure as a Service (IaaS)**, e **Software as a Service (SaaS)**. Si approfondirà la presenza preponderante di AWS come provider di servizi cloud leader, evidenziando la sua vasta gamma di servizi e la rilevanza nel panorama tecnologico.

  Focalizzeremo la nostra attenzione su servizi specifici di AWS, quali , **Lambda**, **API Gateway** e **S3**, esplorando le loro caratteristiche, funzionalità e ruoli nell'ecosistema cloud. Tratteremo inoltre con particolare attenzione **AWS IAM** e **AWS ECR**, sottolineando il loro ruolo critico nella gestione degli accessi e nella distribuzione di container Docker.

- **Il terzo capitolo** costituirà il nucleo metodologico e tecnologico della tesi, delineando le metodologie e le tecnologie che hanno guidato l'implementazione del progetto. Si inizierà con un'analisi approfondita sulle *serie temporali*, mettendo in luce la loro rilevanza nel contesto.

  Successivamente, si presenterà **NeuralProphet**, una libreria *open-source* basata su Python, sviluppata per la previsione temporale utilizzando modelli di reti neurali. Partendo dal framework di previsione Prophet di Facebook, NeuralProphet si distingue per la sua struttura modulare, che include trend, stagionalità, effetti di eventi, e altro ancora.

  La tesi approfondirà il processo di previsione temporale con NeuralProphet, comprendente pre-processing dei dati, decomposizione dei *trend*, generazione delle previsioni attraverso una rete neurale, ottimizzazione dei parametri, *back-testing* e valutazione delle prestazioni. La flessibilità offerta dal *fine-tuning* consentirà l'adattamento del modello alle esigenze specifiche di previsione temporale.

  Inoltre, sarà esplorato **Docker**, una tecnologia di virtualizzazione leggera e portatile, e il linguaggio di programmazione **Python**, centrali nella realizzazione del progetto. Si analizzeranno le caratteristiche distintive di Docker, i suoi vantaggi e il suo ruolo nell'isolamento e distribuzione di servizi. Python, con la sua sintassi chiara, versatilità e ampie librerie standard, sarà presentato come il linguaggio chiave per l'implementazione di diverse componenti del progetto. Questo capitolo fornirà una base solida per comprendere le applicazioni pratiche dei concetti e delle tecnologie discusse nei capitoli successivi, preparando il terreno per le conclusioni della tesi.

- Nel **quarto e conclusivo capitolo**, esploreremo le conclusioni emerse durante il corso del progetto. Inoltre, introdurremo prospettive future per lo sviluppo del progetto, proiettando lo sguardo verso le potenzialità e le

direzioni che potrebbero essere esplorate per arricchire ulteriormente la nostra ricerca.

Capitolo 2

Cloud e Serverless Computing AWS

Nella progettazione e implementazione del nostro sistema, il Cloud Computing emerge come elemento chiave offrendo vantaggi strategici e strumenti avanzati, sebbene richieda una valutazione attenta delle sfide associate per garantire un'implementazione sicura ed efficiente.

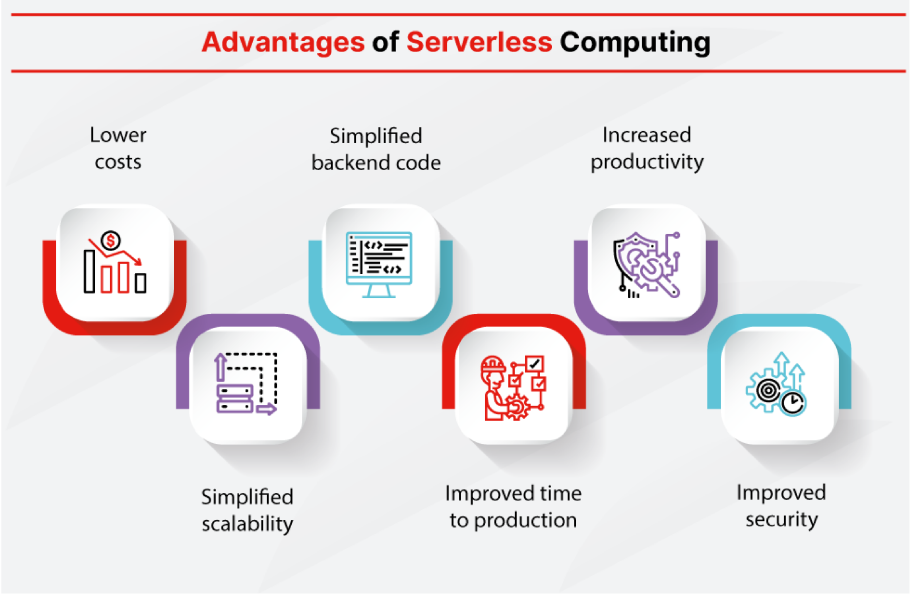

Figura 2.1: Vantaggi del Serverless Computing tratta dall'articolo "What Is Serverless Computing?", Fortinet [2].

2.1 Modelli di Distribuzione dei Servizi

L'avvento del Cloud Computing ha rivoluzionato radicalmente il modo in cui si sviluppano e si implementano le applicazioni, portando con sé un'innovazione

che ha ridefinito il panorama tecnologico. Il Cloud Computing rappresenta una forma di erogazione di servizi informatici basata sulla condivisione di risorse attraverso internet. Ciò significa che anziché affidarsi a infrastrutture locali o server dedicati, le aziende possono accedere a risorse informatiche scalabili e flessibili attraverso il web, fornite da fornitori di servizi Cloud.

L'approfondimento di questo panorama ci porta a esaminare i tre modelli principali di distribuzione dei servizi Cloud:

- **Infrastructure as a Service (IaaS)**: fornisce server, memorie di massa e infrastrutture di rete virtualizzate.
- **Platform as a Service (PaaS)**: offre ambiente e strumenti a livello superiore per lo sviluppo e l'esecuzione delle applicazioni.
- **Software as a Service (SaaS)**: costituito da applicazioni e servizi necessari, accessibili via Internet.

2.1.1 Infrastructure as a Service (IaaS)

IaaS rappresenta il livello più fondamentale di servizi Cloud, offrendo risorse di base come server virtuali, memoria, e storage. Con IaaS, gli utenti hanno il controllo completo sull'ambiente virtuale, potendo configurare e gestire le risorse a loro piacimento. Questo modello è particolarmente adatto per le aziende che necessitano di flessibilità nell'implementazione di applicazioni e servizi su un'infrastruttura personalizzata. Alcuni esempi di servizi IaaS includono Amazon EC2, Microsoft Azure Virtual Machines e Google Compute Engine.

2.1.2 Platform as a Service (PaaS)

PaaS si posiziona a un livello superiore rispetto a IaaS, offrendo un ambiente di sviluppo completo e gestito per la creazione, il test e la distribuzione di applicazioni. Con PaaS, gli sviluppatori possono concentrarsi esclusivamente sullo sviluppo del codice, ignorando la gestione dell'infrastruttura sottostante. Questo modello è ideale per accelerare il ciclo di sviluppo e ridurre la complessità operativa. Servizi PaaS includono piattaforme di sviluppo come Google App Engine, Microsoft Azure App Service e Heroku.

2.1.3 Software as a Service (SaaS)

SaaS rappresenta il livello più alto di astrazione, offrendo applicazioni complete ospitate e gestite dal provider Cloud. Gli utenti accedono a queste applicazioni tramite un browser web, senza dover installare o mantenere software localmente. SaaS è adatto per utenti finali che desiderano accedere a servizi pronti all'uso senza gestire l'infrastruttura sottostante. Esempi di SaaS includono Google Workspace, Microsoft 365 e Salesforce.

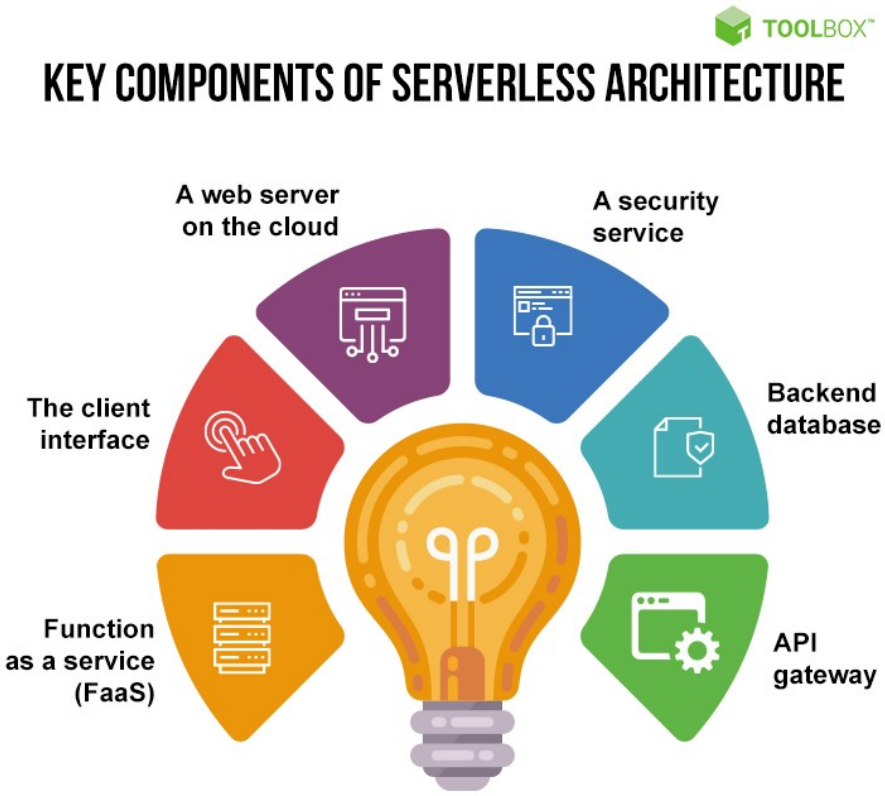

Figura 2.2: Componenti chiave dell'architettura serverless [3].

2.2 Vantaggi del Cloud Computing nel Contesto del Progetto

- **Gestione Delegata**: Con il Cloud, la gestione dell'infrastruttura fisica è affidata al provider, consentendo al nostro team di concentrarsi esclusivamente sullo sviluppo e l'implementazione di funzionalità.
- **Scalabilità Dinamica**: Grazie alla scalabilità automatica, il sistema può adattarsi dinamicamente ai cambiamenti nei carichi di lavoro, garantendo prestazioni ottimali senza investimenti eccessivi in risorse statiche.
- **Costi Flessibili**: Il modello di pagamento Pay-As-You-Go si traduce in una tariffazione basata sull'utilizzo effettivo delle risorse, evitando spese inutili e consentendo una gestione efficiente del budget.
- **Accesso a Servizi Avanzati**: Attraverso il Cloud, accediamo a una vasta gamma di servizi avanzati come Machine Learning, API Gateway e Database NoSQL (DynamoDB), arricchendo il nostro progetto con funzionalità potenti.

2.3 Svantaggi e Sfide

- **Latenza della Rete**: Il trasferimento di dati tra server Cloud può introdurre latenza, influenzando la velocità di trasferimento e l'addestramento di modelli di Machine Learning.
- **Sicurezza e Privacy**: Gestire dati sensibili nel Cloud solleva questioni di sicurezza e privacy. La scelta di un provider con robuste misure di sicurezza è essenziale per affrontare queste preoccupazioni.
- **Dipendenza dal Provider**: L'utilizzo prolungato del Cloud implica una dipendenza dal provider scelto. Valutare attentamente contratti, politiche di prezzo e funzionalità è cruciale per una partnership a lungo termine.

2.4 Applicazioni Specifiche nel Progetto

- **AWS Lambda e API Gateway**: Le Lambda Functions e l'API Gateway consentono l'inserimento ed estrazione di dati all'interno di DynamoDB, contribuendo al flusso di lavoro di ReCity.
- **Machine Learning con NeuralProphet**: L'implementazione di NeuralProphet in Python per la predizione di dati timeseries arricchisce il progetto con capacità di previsione avanzate.
- **Immagine Docker e API Gateway**: La realizzazione di un'immagine Docker contenente il codice per l'addestramento della Rete Neurale, messa a disposizione come servizio tramite API Gateway, completa il flusso di lavoro.

2.5 Amazon Web Services (AWS)

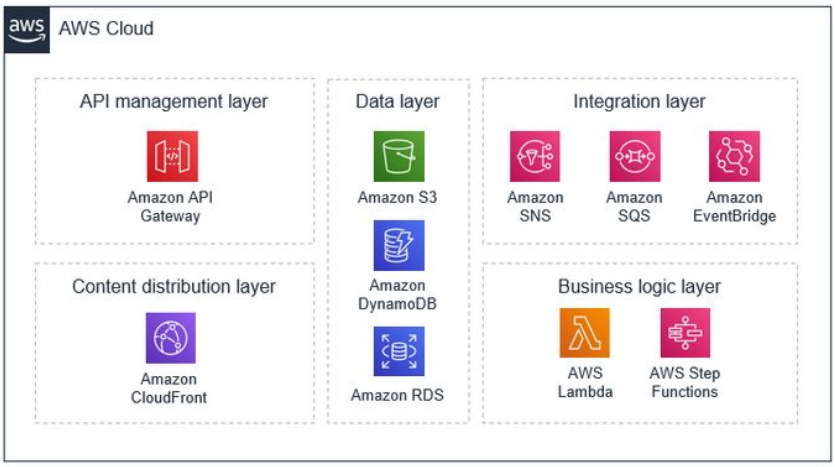

Figura 2.3: AWS [4].

Amazon Web Services (AWS) rappresenta una pietra miliare nell'evoluzione dell'informatica moderna, introducendo un nuovo paradigma di distribuzione delle risorse informatiche attraverso il cloud. Concepito da Amazon nel 2006, AWS ha rivoluzionato il modo in cui le aziende concepiscono e implementano le infrastrutture IT. La sua filosofia si basa su principi fondamentali, tra cui l'elasticità, la scalabilità e la gestione semplificata delle risorse, consentendo alle organizzazioni di concentrarsi sulla creazione di applicazioni innovative senza doversi preoccupare dell'amministrazione delle infrastrutture sottostanti.

2.5.1 Tecnologie Chiave del Progetto

2.5.1.1 Amazon S3 (Simple Storage Service)

Amazon S3, come servizio di storage oggetti, svolge un ruolo fondamentale nella gestione delle risorse di ReCity. La sua scalabilità, sicurezza e durabilità elevata lo rendono ideale per archiviare grandi quantità di dati non strutturati e risorse di immagini. S3 funge da repository centrale, consentendo un accesso rapido e affidabile a dati importanti per l'analisi temporale. La facilità d'uso di S3 semplifica anche il processo di gestione e il recupero dei dati, contribuendo alla coerenza e all'affidabilità del sistema.

2.5.1.2 AWS Lambda

AWS Lambda è il motore operativo che consente l'esecuzione efficiente di operazioni su dati temporali in ReCity. La sua natura serverless permette di scrivere e implementare codice senza preoccuparsi dell'infrastruttura sottostante. Nel contesto di ReCity, Lambda è utilizzato per l'inserimento e l'estrazione di dati in DynamoDB in modo efficiente. La sua capacità di rispondere dinamicamente alle richieste in base al carico di lavoro si traduce in un utilizzo ottimale delle risorse, garantendo che il sistema possa gestire fluttuazioni nella richiesta di dati temporali in modo flessibile ed efficiente.

2.5.1.3 Amazon API Gateway

Amazon API Gateway svolge un ruolo critico nell'esposizione sicura delle funzionalità del sistema di ReCity attraverso API. Questo componente semplifica la creazione e la gestione di API, fornendo un'interfaccia chiara per connettere le applicazioni ai servizi di backend. L'utilizzo di API Gateway facilita l'accesso controllato ai dati temporali e alle previsioni, creando un'interfaccia sicura e gestibile per gli sviluppatori e gli utenti finali.

2.5.1.4 AWS IAM (Identity and Access Management)

AWS IAM gioca un ruolo chiave nella sicurezza e nella gestione degli accessi a risorse AWS in ReCity. La sua capacità di fornire un controllo granulare sugli utenti e sulle risorse contribuisce a garantire un ambiente sicuro. IAM è essenziale per proteggere i dati sensibili e mantenere l'integrità del sistema, assicurando che solo utenti autorizzati possano accedere alle risorse AWS coinvolte nel progetto.

2.5.1.5 Amazon ECR (Elastic Container Registry)

Amazon ECR semplifica la distribuzione e la gestione dei contenitori Docker, creando un ambiente sicuro e scalabile per le immagini Docker necessarie per implementare il modello NeuralProphet in ReCity. La sua funzionalità di repository sicuro contribuisce a garantire che le immagini Docker necessarie per il modello siano archiviate e distribuite in modo efficiente. L'utilizzo di ECR ottimizza la gestione delle immagini, consentendo un deployment agevole e una distribuzione efficace del modello previsionale.

L'interazione sinergica di queste tecnologie nel contesto di ReCity forma una base tecnologica robusta e avanzata, che consente di gestire in modo dinamico e efficiente i dati temporali. Nell'approfondimento dell'implementazione pratica del progetto, si esploreranno le sinergie specifiche tra queste tecnologie, evidenziando come lavorano insieme per ottenere gli obiettivi del sistema.

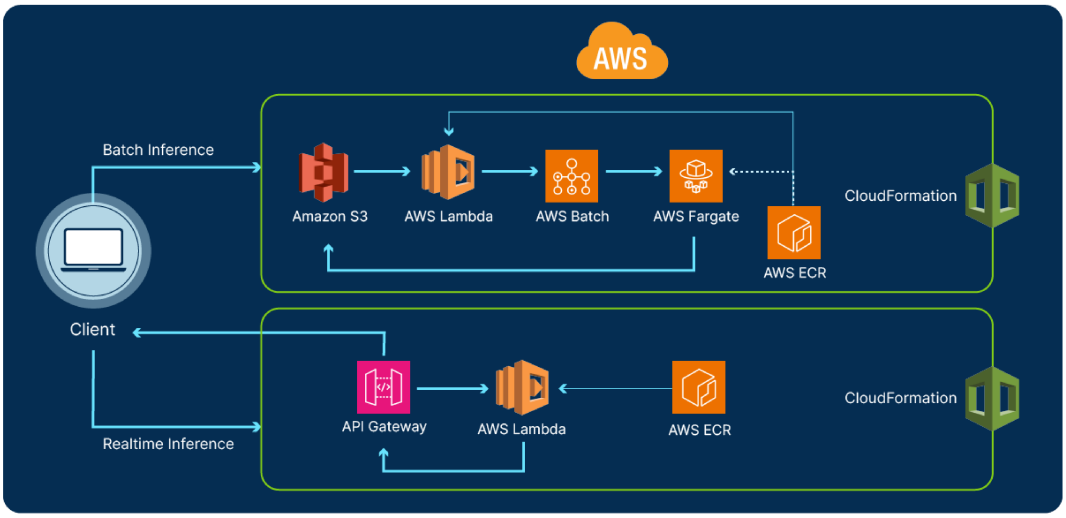

Figura 2.4: Architettura AWS [5].

Capitolo 3

Metodologie e Tecnologie Utilizzate

3.1 Time Series

Le serie temporali costituiscono un aspetto cruciale nell'ambito della data science e dell'analisi statistica. Esse rappresentano dati che variano nel tempo, organizzati in sequenza cronologica. Queste sequenze di osservazioni temporali si verificano in una vasta gamma di contesti, come dati finanziari, sensori IoT (Internet of Things), andamenti climatici e molti altri. L'analisi delle serie temporali riveste un ruolo fondamentale nel comprendere i modelli di comportamento nel tempo e nell'effettuare previsioni significative.

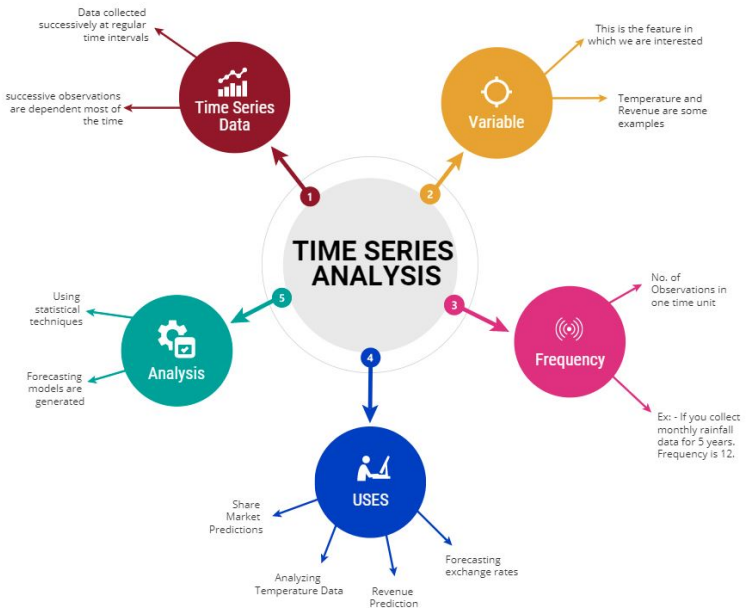

Figura 3.1: Analisi Serie Temporali[6].

3.1.1 Caratteristiche Distintive delle Serie Temporali

- **Dipendenza Temporale** Le serie temporali sono caratterizzate da una dipendenza temporale, dove le osservazioni passate influenzano quelle future. Questo legame dinamico richiede metodi analitici specifici per cogliere le relazioni intrinseche nei dati nel corso del tempo.

- **Stazionarietà** L'assunzione di stazionarietà semplifica l'analisi delle serie temporali. Una serie stazionaria mantiene una struttura statistica costante nel tempo, semplificando l'applicazione di modelli predittivi.

- **Trend e Stagionalità** Molte serie temporali mostrano trend a lungo termine e pattern ciclici stagionali. L'identificazione e la gestione di queste componenti sono essenziali per un'analisi accurata e la creazione di modelli predittivi robusti.

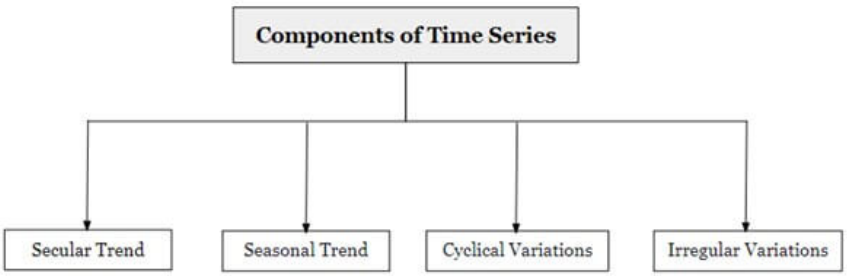

Figura 3.2: Componenti Serie Temporali [7].

3.1.2 Approccio nel Contesto di ReCity

- **Ruolo Cruciale delle Serie Temporali** Nel progetto ReCity, le serie temporali svolgono un ruolo cruciale nel monitorare il flusso dinamico di dati provenienti dai sensori self-sensing. L'analisi temporale consente di individuare pattern, rilevare anomalie e comprendere le dinamiche sottostanti.

- **Utilizzo di NeuralProphet per la Predizione** Per affrontare la complessità delle serie temporali, il progetto si avvale di NeuralProphet, un modello basato su reti neurali progettato per gestire dati temporali e generare previsioni accurate. Questo approccio avanzato consente una previsione dinamica e adattativa alle variazioni nel tempo.

- **Gestione Scalabile tramite Architettura Serverless** L'architettura serverless adottata facilita la gestione efficiente delle serie temporali.

Attraverso l'uso di Lambda Functions e servizi Cloud, come Amazon DynamoDB, il sistema può archiviare, analizzare ed estrarre dati temporali in modo scalabile, garantendo una risposta dinamica alle esigenze del progetto.

L'analisi delle serie temporali nel contesto di ReCity non solo fornisce una comprensione approfondita delle variazioni temporali nei dati self-sensing ma costituisce anche la base per un sistema predittivo avanzato. Integrando l'Intelligenza Artificiale, l'approccio serverless e l'analisi temporale, il progetto mira a ottimizzare la gestione delle risorse , anticipando dinamiche e contribuendo a una visione più intelligente.

3.2 Forecasting per le Serie Temporali con NeuralProphet

3.2.1 Introduzione al Forecasting per le Serie Temporali

Le serie temporali giocano un ruolo sempre più predominante in diversi settori industriali, con molte aziende che raccolgono quantità considerevoli di dati. Questa abbondanza di dati ha superato le capacità analitiche tradizionali, suscitando l'interesse nell'elaborazione e nella previsione delle serie temporali industriali. Tale previsione è diventata cruciale, soprattutto in applicazioni in cui le decisioni aziendali o operative basate su previsioni possono avere conseguenze significative, anche potenzialmente fatali.

3.2.2 Caratteristiche delle Serie Temporali

Le serie temporali rappresentano dati organizzati in sequenze temporali ordinate, con ciascun dato associato a un *timestamp* che indica il momento della sua acquisizione. Possono essere unidimensionali, come una sequenza di valori numerici registrati a intervalli regolari, o multidimensionali, coinvolgendo più variabili misurate simultaneamente. Le serie temporali spesso mostrano strutture intrinseche come *trend*, *stagionalità*, *ciclicità*, *periodicità* e *variazioni casuali*.

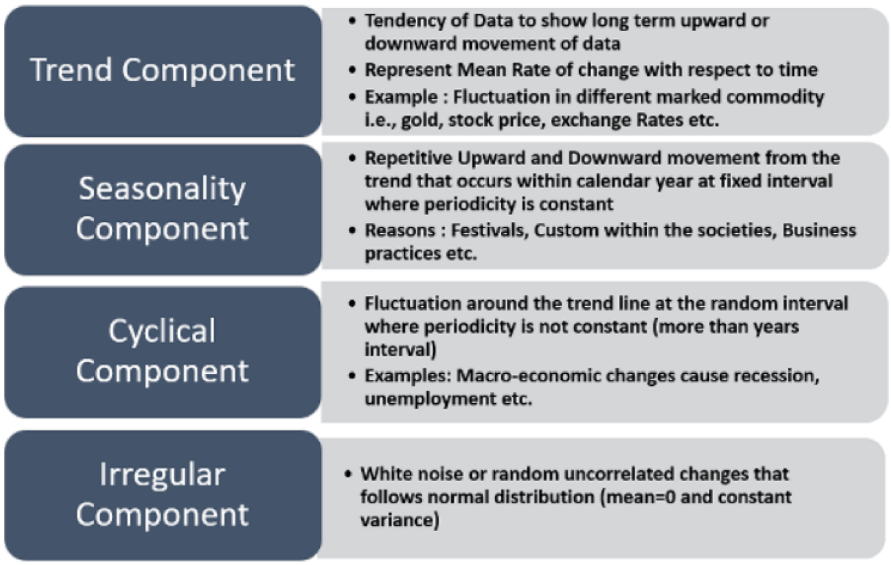

Figura 3.3: Componenti Serie Temporali [8].

3.2.3 Importanza dell'Analisi e della Previsione delle Serie Temporali

L'analisi e la previsione delle serie temporali rivestono un ruolo cruciale in molti contesti. Comprendere i modelli e le dinamiche nei dati fornisce informazioni preziose per prendere decisioni informate, fare previsioni future e identificare anomalie o deviazioni dai comportamenti attesi.

3.2.4 Approccio di Forecasting con NeuralProphet

Estendendo il framework di previsione Prophet di Facebook, NeuralProphet sfrutta una rete neurale *feedforward* completamente connessa, con una struttura modulare che comprende trend, stagionalità, effetti di eventi, effetti di regressione, auto-regressione e regressione su osservazioni ritardate.

Figura 3.4: NeuralProphet Logo [1].

3.2.5 Processo di Previsione Temporale con NeuralProphet

Il processo di previsione temporale con NeuralProphet comprende il pre-processing dei dati, la decomposizione dei trend utilizzando una variante della decomposizione di Fourier, la generazione delle previsioni attraverso una rete neurale,

l'ottimizzazione dei parametri del modello, il back-testing delle previsioni e la valutazione delle prestazioni con metriche come *l'errore medio assoluto (MAE)*, *l'errore quadratico medio (RMSE)* e *coefficiente di determinazione (R^2)* . NeuralProphet offre anche la flessibilità di iterare e migliorare il modello attraverso il fine-tuning, consentendo la personalizzazione per specifiche esigenze di previsione temporale.

NeuralProphet rappresenta un ponte tra i metodi tradizionali e quelli basati sull'apprendimento automatico, offrendo un approccio flessibile e potente per il forecasting preciso delle serie temporali nel contesto di ReCity.

3.3 Introduzione a NeuralProphet

3.3.1 Panoramica di NeuralProphet

Essa rappresenta una significativa estensione del framework di previsione chiamato Prophet, originariamente sviluppato da Facebook per la previsione temporale. NeuralProphet sfrutta le potenzialità delle reti neurali per migliorare la precisione delle previsioni, introducendo un approccio modulare che permette la configurazione flessibile di vari aspetti del modello.

3.3.2 Fondamenti di NeuralProphet

3.3.2.1 Struttura Modulare

Una caratteristica distintiva di NeuralProphet è la sua struttura modulare. Il modello è composto da diversi moduli, ognuno dei quali contribuisce con un componente specifico alla previsione. Questi moduli includono:

- **T(t): Tendenza al tempo t**
- **S(t): Effetti stagionali al tempo t**
- **E(t): Effetti degli eventi e delle festività al tempo t**
- **F(t): Effetti di regressione al tempo t per variabili esogene conosciute nel futuro**
- **A(t): Effetti di auto-regressione al tempo t basati su osservazioni passate**
- **L(t): Effetti di regressione al tempo t per osservazioni ritardate di variabili esogene**

La possibilità di attivare o disattivare singolarmente questi moduli conferisce una flessibilità notevole, adattando il modello alle esigenze specifiche del problema di previsione.

3.3.2.2 Modello Feedforward Completamente Connesso

NeuralProphet si basa su un modello di rete neurale feedforward completamente connesso, noto anche come *rete neurale feedforward multistrato (MLP)*. Questo tipo di modello è riconosciuto per la sua abilità di apprendere modelli complessi dai dati di input. L'approccio feedforward implica il passaggio dell'informazione attraverso strati successivi di nodi, trasformando gli input in previsioni accurate.

3.3.2.3 Incorporazione del Modello AR-Net

Un aspetto significativo di NeuralProphet è l'introduzione del modello **AR-Net (AutoRegressive Neural Network)**, una rete neurale completamente connessa in cui l'output istantaneo corrisponde al valore al tempo t, mentre gli input sono costituiti dai p valori precedenti della serie temporale. Questo modello è particolarmente efficace nel trattamento delle serie temporali in cui il futuro a breve termine dipende dallo stato attuale del sistema.

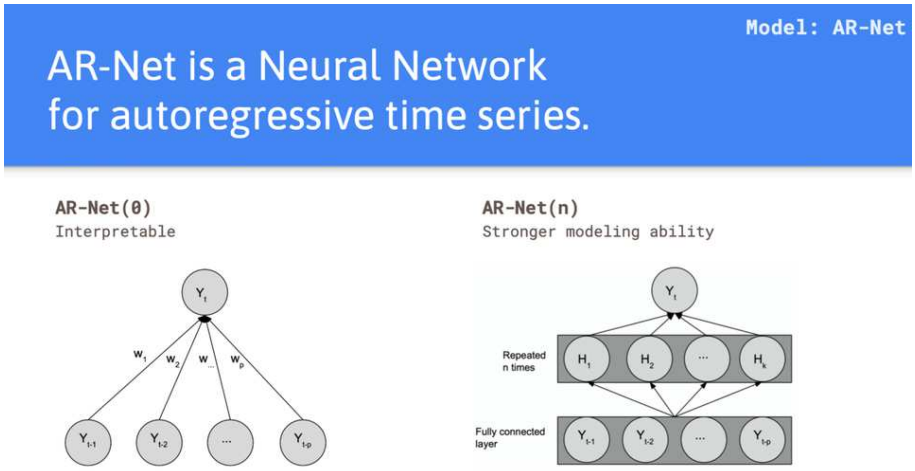

Figura 3.5: AR-Net [9].

3.3.3 Processo di Previsione Temporale con NeuralProphet

3.3.3.1 Pre-processing dei Dati

Prima di generare previsioni, NeuralProphet richiede un pre-processing dei dati. Questo include l'elaborazione dei dati temporali, l'imputazione dei valori mancanti e, se necessario, la normalizzazione dei dati. NeuralProphet può gestire sia dati con intervalli di tempo regolari che irregolari, garantendo flessibilità nella gestione di diversi contesti di serie temporali.

3.3.3.2 Decomposizione dei Trend

Utilizzando una variante della decomposizione di Fourier, NeuralProphet identifica i diversi componenti del trend nei dati. Ciò include la tendenza generale,

le stagionalità annuali, le stagionalità settimanali e gli effetti delle festività. Questa decomposizione dei trend consente al modello di separare i fattori che contribuiscono alle fluttuazioni dei dati.

3.3.3.3 Generazione delle Previsioni

Dopo la decomposizione dei trend, NeuralProphet utilizza la sua rete neurale per generare previsioni precise. La rete neurale, durante l'addestramento, ottimizza i pesi delle connessioni tra i nodi per ridurre al minimo l'errore tra le previsioni e i valori reali.

3.3.3.4 Ottimizzazione dei Parametri

NeuralProphet sfrutta un algoritmo di ottimizzazione per trovare i parametri del modello che massimizzano l'accuratezza delle previsioni. Questi parametri includono l'architettura della rete neurale, i pesi delle connessioni, gli iper-parametri dell'algoritmo di ottimizzazione e altre configurazioni specifiche del modello.

3.3.3.5 Valutazione delle Prestazioni

Dopo l'addestramento del modello, NeuralProphet consente la generazione di previsioni future, valutate utilizzando misure di prestazione come *l'errore medio assoluto (MAE)*, *l'errore quadratico medio (RMSE)* o il *coefficiente di determinazione (R^2)*.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}|$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2$$

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ − *predicted value of y*

$\bar{y}$ − *mean value of y*

Figura 3.6: Formule MAE, MSE, RMSE, R^2 [10].

3.3.3.6 Fine-tuning e Iterazione

NeuralProphet offre la possibilità di iterare e migliorare il modello attraverso il **fine-tuning**. Gli utenti possono regolare i parametri del modello, modificare l'architettura della rete neurale o apportare altre modifiche per ottenere previsioni più accurate.

3.3.4 Versatilità e Personalizzazione di NeuralProphet

NeuralProphet è estremamente flessibile e può essere personalizzato per adattarsi alle specifiche esigenze del problema di previsione temporale. L'utente ha la possibilità di configurare diversi aspetti del modello, come l'architettura della rete neurale, i parametri di ottimizzazione e le modalità di decomposizione dei trend.

In sintesi, NeuralProphet rappresenta un'importante risorsa per la previsione temporale avanzata, combinando l'affidabilità del framework Prophet con le potenzialità delle reti neurali, offrendo così una soluzione potente e personalizzabile per affrontare sfide complesse nel contesto di ReCity.

3.4 Docker

3.4.1 Introduzione a Docker

Docker è una piattaforma di virtualizzazione leggera e potente che consente di impacchettare, distribuire e eseguire applicazioni in contenitori. Questa tecnologia ha rivoluzionato l'approccio allo sviluppo e al deployment delle applicazioni, fornendo un ambiente isolato e portatile che garantisce coerenza tra sviluppo, test e produzione.

Figura 3.7: Docker Logo [11].

3.4.2 Concetti Chiave

3.4.2.1 Contenitori Docker

Un contenitore Docker è un'istanza eseguibile di un'immagine, che a sua volta è un pacchetto leggero che include tutto il necessario per eseguire un'applicazione: il codice sorgente, le dipendenze, le librerie, e le configurazioni. L'utilizzo di contenitori facilita la creazione di ambienti consistenti, riducendo al minimo i problemi dovuti alle differenze tra le configurazioni dei sistemi.

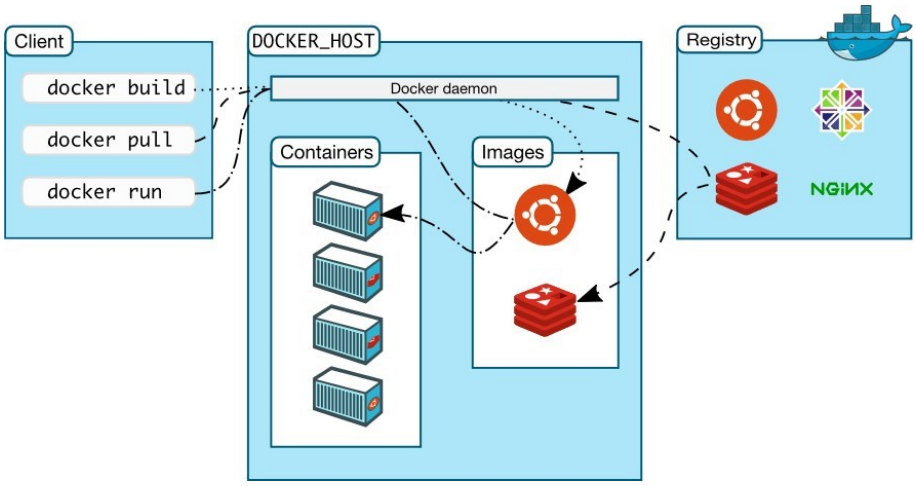

Figura 3.8: Architettura Docker[12].

3.4.2.2 Docker Hub

Docker Hub è un registro online che funge da repository per immagini Docker. Gli sviluppatori possono condividere le proprie immagini, semplificando la distribuzione e la condivisione di applicazioni e servizi. Docker Hub offre un accesso rapido a una vasta gamma di immagini pronte all'uso, riducendo il tempo di sviluppo e semplificando l'integrazione di software di terze parti.

3.4.3 Vantaggi di Docker

3.4.3.1 Portabilità

I contenitori Docker forniscono un'astrazione efficiente a livello di sistema operativo, garantendo la stessa esperienza di esecuzione indipendentemente dall'ambiente in cui vengono eseguiti. Ciò consente la portabilità delle applicazioni tra ambienti di sviluppo, test e produzione senza preoccuparsi delle differenze infrastrutturali.

3.4.3.2 Isolamento

I contenitori offrono un isolamento leggero tra le applicazioni, consentendo loro di eseguire processi separati su una stessa macchina host. Questo contribuisce a evitare conflitti tra dipendenze e a garantire la stabilità dell'ambiente di esecuzione.

3.4.3.3 Efficienza delle Risorse

Docker ottimizza l'utilizzo delle risorse, poiché i contenitori condividono il kernel del sistema operativo host, riducendo l'overhead rispetto alle tradizionali macchine virtuali. Ciò si traduce in una maggiore efficienza nell'uso di CPU, memoria e spazio di archiviazione.

3.4.3.4 Scalabilità

La natura leggera e modulare dei contenitori semplifica la scalabilità delle applicazioni. Docker consente di replicare e distribuire facilmente contenitori su più nodi, gestendo il carico di lavoro in modo efficiente e garantendo la disponibilità delle applicazioni.

3.4.4 Utilizzo di Docker in ReCity

3.4.4.1 Contenitori per Servizi

Nel contesto di ReCity, Docker viene impiegato per isolare e distribuire servizi cruciali, garantendo una gestione semplificata e affidabile. I servizi come DynamoDB, Lambda, e l'API Gateway vengono eseguiti all'interno di contenitori Docker, facilitando la distribuzione su ambienti cloud come AWS.

3.4.4.2 Ambienti di Sviluppo Consistenti

Nel progetto ReCity utilizziamo Docker per creare ambienti di sviluppo consistenti, eliminando le disparità tra le configurazioni di sviluppo e produzione. Ciò riduce gli errori dovuti a differenze ambientali e accelera il ciclo di sviluppo.

3.4.4.3 Integrazione con Strumenti di Continuous Integration

Docker integra senza soluzione di continuità con gli strumenti di continuous integration, consentendo la creazione di pipeline di sviluppo e rilascio automatiche. Questa integrazione agevola la consegna continua e la distribuzione affidabile delle applicazioni.

Docker ha trasformato il modo in cui sviluppatori e operatori gestiscono le applicazioni, offrendo un approccio innovativo e efficiente alla virtualizzazione basata su contenitori. Nell'ambito di ReCity, l'utilizzo di Docker ha migliorato la coerenza ambientale, semplificato la distribuzione delle applicazioni e ottimizzato l'uso delle risorse. L'adozione di Docker rappresenta un passo significativo verso un ambiente di sviluppo e produzione più coeso e scalabile.

3.5 Python

Nel contesto di questo progetto, Python emerge come linguaggio di programmazione chiave, svolgendo un ruolo fondamentale nell'implementazione di diverse componenti. Le sue caratteristiche distintive contribuiscono all'efficienza, alla flessibilità e alla coerenza del sistema sviluppato.

Figura 3.9: Logo del linguaggio di programmazione Python [13].

3.5.1 Caratteristiche Principali di Python

- **Sintassi Chiara e Leggibile** La sintassi di Python è progettata per essere chiara e leggibile, facilitando la comprensione del codice. Questa caratteristica è cruciale per il mantenimento e la collaborazione tra sviluppatori.

- **Versatilità e Modularità** Python supporta un approccio modulare, consentendo agli sviluppatori di organizzare il codice in moduli riutilizzabili. Questa versatilità favorisce la creazione di un sistema ben strutturato e mantenibile.

- **Ampia Libreria Standard** Python dispone di una vasta libreria standard che copre una vasta gamma di funzionalità. Queste librerie offrono strumenti predefiniti che semplificano compiti comuni, accelerando lo sviluppo e riducendo la necessità di scrivere codice da zero.

- **Interpretato e Orientato agli Oggetti** Python è un linguaggio interpretato, il che significa che il codice può essere eseguito direttamente senza una fase di compilazione separata. Inoltre, Python segue un paradigma di programmazione orientato agli oggetti, consentendo la creazione di codice organizzato in classi e oggetti.

- **Community Attiva e Supporto** Python gode di una vasta community di sviluppatori che contribuiscono al suo sviluppo continuo. Questo garantisce un rapido accesso a risorse, documentazione e supporto online, facilitando la risoluzione di problemi e l'apprendimento.

- **Compatibilità con Altre Tecnologie** Python è interoperabile con diverse tecnologie e piattaforme. La sua flessibilità facilita l'integrazione con altri servizi e strumenti utilizzati nel contesto del progetto.

Non a caso, a partire dall'anno 2016, Python ha registrato un notevole incremento nell'utilizzo da parte dell'utenza, rispetto ad altri linguaggi ottimizzati per le data science (come ad esempio il linguaggio di programmazione R), come si vede in figura 3.10.

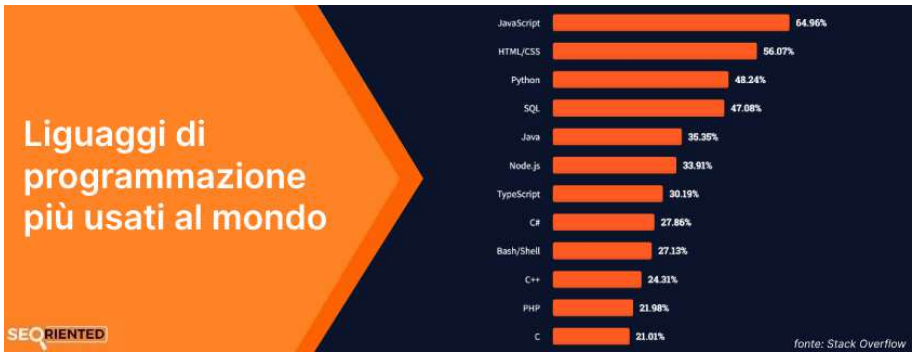

Figura 3.10: Linguaggi di programmazione più usati nel mondo nel 2022 [14].

3.5.2 Ruolo di Python nel Progetto

Nel contesto di questo progetto, Python viene utilizzato come linguaggio principale per lo sviluppo di Lambda Functions, l'implementazione della Rete Neurale con NeuralProphet, e la creazione di un'immagine Docker per la distribuzione di servizi. La scelta di Python si basa sulla sua idoneità per lo sviluppo rapido, la gestione di algoritmi di machine learning, e la creazione di soluzioni scalabili in un ambiente serverless. La sua versatilità si riflette nella capacità di adattarsi alle diverse esigenze del progetto, fornendo un fondamento solido per la realizzazione di un sistema complesso e innovativo.

Capitolo 4

Sviluppo del Progetto

Le serie temporali rappresentano una risorsa cruciale nell'analisi dei dati, svolgendo un ruolo fondamentale nella comprensione dei fenomeni che evolvono nel tempo. Nel contesto del progetto ReCity, si è cercato di sviluppare un modello predittivo basato su NeuralProphet, una libreria avanzata per il forecasting temporale. Questo capitolo è dedicato a delineare il processo di implementazione, compresi i passaggi necessari per integrare dati isometrici e pluviometrici provenienti da un file Excel specificamente creato per il progetto ReCity.

4.1 Origine dei Dati: Progetto ReCity

I dati utilizzati per alimentare il modello provengono da un file Excel che raccoglie informazioni isometriche e pluviometriche rilevanti per il progetto ReCity. Questo dataset rappresenta un'inestimabile fonte di informazioni, poiché consente di esplorare le variazioni temporali delle misurazioni, fornendo così un quadro completo delle dinamiche ambientali coinvolte nel progetto.

4.2 Processo di Implementazione

4.2.1 Elaborazione dei Dati Isometrici e Pluviometrici

Il primo passo coinvolge l'estrazione e l'elaborazione dei dati isometrici e pluviometrici dal file Excel del progetto ReCity. Questi dati vengono quindi preparati per l'ingestione da parte del modello NeuralProphet, garantendo che siano strutturati in modo appropriato per la previsione temporale.

4.2.2 Implementazione del Modello NeuralProphet

La libreria NeuralProphet è stata selezionata per la sua potenza e flessibilità nell'affrontare complesse previsioni temporali. La configurazione del modello include la scelta di parametri rilevanti, la suddivisione del dataset per la formazione e la validazione, nonché la gestione di eventuali caratteristiche stagionali o tendenze presenti nei dati.

4.2.3 Utilizzo di Docker per Garantire la Riproducibilità

Al fine di garantire la portabilità e la riproducibilità del modello, è stato adottato Docker per creare un ambiente isolato. Ciò consente di assicurare che il codice e le dipendenze possano essere eseguiti in modo coerente, indipendentemente dall'ambiente di esecuzione.

4.2.4 Creazione di un File requirements.txt per la Gestione delle Dipendenze

Per semplificare l'installazione delle librerie necessarie, è stato creato il file requirements.txt, che elenca tutte le dipendenze necessarie per eseguire il codice con successo. Questo approccio assicura che l'ambiente di sviluppo sia coerente e che il modello possa essere eseguito senza difficoltà su diverse piattaforme.

Questa sezione fornisce un contesto dettagliato sulle origini dei dati, il processo di preparazione e la struttura del modello, ponendo le basi per una comprensione approfondita dei risultati presentati successivamente nel capitolo.

4.3 Struttura Codice

Il codice all'interno del file denominato *data.py* implementa una soluzione completa per la previsione delle serie temporali utilizzando la libreria NeuralProphet e i dati del progetto ReCity, compresi i dati isometrici e pluviometrici provenienti da un file Excel dedicato.

4.3.1 Estrazione e Preparazione dei Dati

- **Caricamento dei Dati**: Viene caricato il file Excel contenente dati isometrici e pluviometrici attraverso la libreria openpyxl.

- **Creazione del DataFrame**: I dati vengono estratti dal foglio di lavoro attivo del file Excel e convertiti in un DataFrame di pandas. Le colonne 'ds', 'y', e 'Pioggia giornaliera [mm]' vengono inizializzate.

- **Conversione e Gestione dei Dati Mancanti**: Le colonne 'ds' vengono convertite in formato datetime, 'y' e 'Pioggia giornaliera [mm]' in numerico. Eventuali valori mancanti nella colonna 'y' vengono interpolati. La colonna 'ds' viene anche popolata con il valore minimo meno un giorno nel caso in cui ci siano dati mancanti iniziali.

```python
def exctract_yhat(forecasts, size):
    columns = forecasts.columns[3:]
    newframe = forecasts[['ds', 'yhat1']].iloc[-size:].copy()
    for col in columns:
        if 'yhat' in col:
            newframe['yhat1'] = newframe['yhat1'].fillna(forecasts[col])
    return newframe

# Read Excel file using openpyxl
dataframe = openpyxl.load_workbook("Dati inclinometrici e pluviometrici_ReCity.xlsx")
dataframe1 = dataframe.active

# Create a list to store rows
rows_list = []

# Iterate through rows and columns to extract data
for row in dataframe1.iter_rows(min_row=2, values_only=True):
    row_dict = {'ds': row[0], 'y': row[1], 'Pioggia giornaliera [mm]': row[2]}
    rows_list.append(row_dict)

# Create a Pandas DataFrame
df = pd.DataFrame(rows_list)

# Convert 'ds' to datetime format
df['ds'] = pd.to_datetime(df['ds'], errors='coerce')

# Convert 'y' to numeric format
df['y'] = pd.to_numeric(df['y'], errors='coerce')

# Convert 'Pioggia giornaliera [mm]' to numeric format
df['Pioggia giornaliera [mm]'] = pd.to_numeric(df['Pioggia giornaliera [mm]'], errors='coerce')

# Fill NaN values in the 'y' column with 0
df['y'] = df['y'].fillna(method='ffill')

# Fill NaN values in the 'ds' column with a specific timestamp
df['ds'] = df['ds'].fillna(value=df['ds'].min() - pd.Timedelta(days=1))
print(df.columns)
```

Figura 4.1: Codice per l'estrazione e preparazione dei dati

4.3.2 Pulizia e Gestione dei Dati Mancanti

Interpolazione dei Dati Mancanti: I dati mancanti nella colonna 'y' vengono ulteriormente gestiti, con l'utilizzo della propagazione in avanti (ffill) e all'indietro (bfill) per garantire la continuità nei dati.

```python
df['y'] = df['y'].fillna(method='ffill')
df['y'] = df['y'].fillna(method='bfill')
df['Pioggia giornaliera [mm]'] = df['Pioggia giornaliera [mm]'].fillna(method='ffill')
df['Pioggia giornaliera [mm]'] = df['Pioggia giornaliera [mm]'].fillna(method='bfill')
```

Figura 4.2: Codice per la gestione dei dati mancanti

4.3.3 Filtraggio delle Date e Suddivisione del Dataset

Filtraggio delle Date: Viene effettuato un filtro temporale per includere solo le date comprese tra il 2 gennaio 2016 e il 16 settembre 2017.
Suddivisione del Dataset: Il dataset viene suddiviso in dati di allenamento (**dftrain**) e dati di test (**dftest**). In questo caso, il 10% dei dati più recenti viene utilizzato come dati di test.

```
# Filter data between January 2, 2016, and September 16, 2017
start_date = '2016-01-02'
end_date = '2017-09-16'
df_filtered = df[(df['ds'] >= start_date) & (df['ds'] <= end_date)]

df['y'] = df['y'].fillna(method='ffill')
df['y'] = df['y'].fillna(method='bfill')
df['Pioggia giornaliera [mm]'] = df['Pioggia giornaliera [mm]'].fillna(method='ffill')
df['Pioggia giornaliera [mm]'] = df['Pioggia giornaliera [mm]'].fillna(method='bfill')

#train_proportion = 0.8
#sz = round(len(df) * train_proportion)
sz = round(len(df)/100*10)
dftrain = df[:- sz].copy()
dftest = df[-sz:].copy()
```

Figura 4.3: Codice

4.3.4 Allenamento del Modello NeuralProphet

Inizializzazione del Modello: Viene inizializzato un modello NeuralProphet con stagionalità annuale abilitata.
Aggiunta di Regressori: Viene aggiunta una variabile regresiva "Pioggia giornaliera [mm]" al modello.
Addestramento del Modello: Il modello viene quindi addestrato utilizzando i dati di allenamento con una frequenza giornaliera.

```
# Train NeuralProphet model with filtered data and yearly seasonality
model = NeuralProphet(yearly_seasonality=True)
model.add_lagged_regressor('Pioggia giornaliera [mm]')
#model.fit(df_filtered, freq="D")
model.fit(dftrain, freq="D")

regressors = dftest.copy()
regressors['y'] = pd.Series([float('nan')]*len(dftest))
regressors['Pioggia giornaliera [mm]'] = dftest['Pioggia giornaliera [mm]']
```

Figura 4.4: Codice

4.3.5 Generazione delle Predizioni

Creazione del DataFrame Futuro: Viene creato un DataFrame futuro (**future**) che estende la sequenza temporale del dataset di allenamento.
Generazione delle Predizioni: Il modello viene utilizzato per generare le predizioni (**forecast**) basate su questo futuro DataFrame, incorporando la variabile regressiva.

```
future = model.make_future_dataframe(dftrain, periods=30, regressors_df=regressors, n_historic_predictions=True)
forecast = model.predict(future)
```

Figura 4.5: Codice

4.3.6 Visualizzazione delle Predizioni

Creazione della Figura del Grafico: Il metodo **plot** di NeuralProphet viene utilizzato per creare una figura del grafico che rappresenta le previsioni generate dal modello. La figura è composta da diversi sottografici, ciascuno mostrando un aspetto specifico delle predizioni, come trend, stagionalità annuale, settimanale, ecc.
Visualizzazione del Grafico: La funzione **show** di **matplotlib.pyplot** è chiamata per visualizzare il grafico creato. Il grafico presenta chiaramente la comparazione tra i dati reali e le previsioni del modello su un intervallo di tempo specifico.

```python
fig_forecast = model.plot(forecast)
plt.show()
# Print the available columns in the forecast DataFrame
print(forecast.columns)

# Print the forecasted values
print(forecast[['ds', 'yhat1', 'trend', 'season_yearly', 'season_weekly']])

# Plot the actual values
plt.plot(df_filtered['ds'], df_filtered['y'], label='Actual', color='blue' , marker='.')

# Plot the forecasted values
plt.plot(forecast['ds'], forecast['yhat1'], label='Forecast', color='red')

# Plot the trend, yearly seasonality, and weekly seasonality
plt.plot(forecast['ds'], forecast['trend'], label='Trend', linestyle='--', color='green')

# Check if 'season_yearly' is present in the DataFrame
if 'season_yearly' in forecast.columns:
    plt.plot(forecast['ds'], forecast['season_yearly'], label='Yearly Seasonality', linestyle='--', color='orange')

# Plot the weekly seasonality if present
if 'season_weekly' in forecast.columns:
    plt.plot(forecast['ds'], forecast['season_weekly'], label='Weekly Seasonality', linestyle='--', color='purple')

# Add labels and title
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Actual vs Forecasted Values with Trend and Seasonality')

# Show the legend
plt.legend()

# Display the plot
plt.show()
```

Figura 4.6: Codice

4.3.6.1 Dettagli aggiuntivi sul Grafico

Il grafico generato dalla libreria NeuralProphet offre molte informazioni utili:

- **Dati Reali e Predizioni**
  Il grafico visualizza i dati reali, rappresentati dai punti blu, insieme alle predizioni generate dal modello, rappresentate dalla linea rossa. Questo consente un confronto immediato tra le performance del modello e l'andamento effettivo delle serie temporali.

- **Trend**
  La linea verde punteggiata rappresenta il trend generale delle serie temporali. Questo aiuta a identificare le direzioni principali o le variazioni nel tempo dei dati.

- **Stagionalità Annuale**
  La stagionalità annuale nel modello, sarà evidenziata nel grafico. Questo è particolarmente rilevante per identificare pattern ricorrenti che seguono cicli annuali specifici.

- **Stagionalità Settimanale**
  Analogamente, la stagionalità settimanale, sarà chiaramente visibile. Questa componente è utile per catturare variazioni periodiche che si ripetono ogni settimana.

- **Ulteriori Componenti**
  A seconda della configurazione specifica del modello, il grafico può mostrare ulteriori componenti come le componenti di festività o regressive.

La visualizzazione di queste componenti fornisce una rappresentazione dettagliata delle variazioni temporali previste dal modello, consentendo una migliore comprensione delle dinamiche idrologiche e ambientali nel contesto del progetto ReCity.

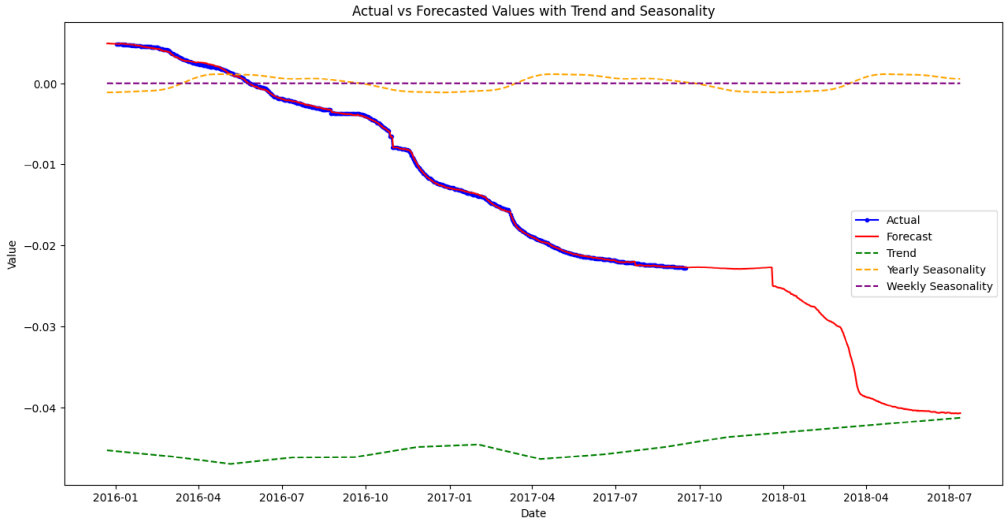

Figura 4.7: Grafico

4.4 Docker

- Nel file Docker, è stato definito l'ambiente di esecuzione isolato per l'applicazione Python. Inizialmente, è stata selezionata l'immagine ufficiale del runtime Python, optando per la versione 3.8 come base per l' ambiente di esecuzione.

- Successivamente, è stata impostata la directory di lavoro all'interno del contenitore nella cartella /app. Questo fornirà un punto di partenza per

l'esecuzione delle successive operazioni all'interno dell' ambiente di sviluppo.

- Per garantire che tutti i file e le dipendenze necessarie siano presenti nell' ambiente isolato, è stato utilizzato il comando ADD per copiare i contenuti della directory corrente (dove si trova il file Docker) all'interno del contenitore, precisamente nella directory /app.
- Per soddisfare le dipendenze specificate nel file requirements.txt, è stato utilizzato il comando RUN per eseguire l'installazione delle librerie, evitando di memorizzare nella cache per ridurre le dimensioni dell'immagine Docker.
- Si è deciso di esporre la porta 80 all'esterno del contenitore attraverso il comando EXPOSE. Questo permette la comunicazione con altri container o applicazioni al di fuori dell' ambiente isolato.
- Per personalizzare ulteriormente l'ambiente isolato, è stata definita una variabile d'ambiente denominata NAME con il valore "World". Questa variabile può essere utilizzata all'interno dell'applicazione Python.
- Infine, è stato specificato che l'applicazione Python denominata "data.py" sarà eseguita nel momento in cui il contenitore viene avviato, utilizzando il comando CMD.

Complessivamente, il file Docker creato configura un ambiente isolato, pronto per eseguire l'applicazione Python, con tutte le dipendenze e le impostazioni ambientali necessarie.

Dockerfile > ...

```
1   # Use an official Python runtime as a parent image
2   FROM python:3.8
3
4   # Set the working directory in the container
5   WORKDIR /app
6
7   # Copy the current directory contents into the container at /app
8   ADD . /app
9
10  # Install any needed packages specified in requirements.txt
11  RUN pip install --no-cache-dir -r requirements.txt
12
13  # Make port 80 available to the world outside this container
14  EXPOSE 80
15
16  # Define environment variable
17  ENV NAME World
18
19  # Run app.py when the container launches
20  CMD ["python", "data.py"]
21
22
```

Figura 4.8: Codice Dockerfile

4.5 File "requirements.txt"

Nel corso dell'implementazione del progetto ReCity, è stato essenziale gestire accuratamente le dipendenze del software per garantire una corretta esecuzione e riproducibilità delle analisi. A tale scopo, è stato creato il file **requirements.txt**, che elenca tutte le librerie Python necessarie per eseguire il codice.

4.5.1 Librerie

Il file requirements.txt è un file di testo che elenca tutte le librerie Python necessarie per eseguire correttamente il progetto.

- **Neuralprophet**:Si tratta di una libreria Python per la previsione di serie temporali basata su reti neurali. Essa offre un'implementazione user-friendly di modelli di previsione delle serie temporali, facilitando l'addestramento e la valutazione di reti neurali specifiche per questo scopo.

- **Pandas**: Questa è una delle librerie più utilizzate in Python per la manipolazione e l'analisi dei dati. Pandas fornisce strutture dati potenti e veloci, come i DataFrame, che semplificano notevolmente le operazioni su dati tabulari.

- **Openpyxl**: Una libreria Python per la lettura e la scrittura di file Excel.

- **Matplotlib**: Una libreria di visualizzazione dati che consente di creare grafici e tracciati.

4.5.2 Approccio al Progetto ReCity

Il file **requirements.txt** include le seguenti librerie:

- **Neuralprophet**: Utilizzata per implementare modelli di previsione delle serie temporali basati su reti neurali.

- **Pandas**: Essenziale per la manipolazione e l'analisi dei dati, fornendo strutture dati potenti come i DataFrame.

- **Openpyxl**: Utilizzata per la lettura e la scrittura di file Excel, indispensabile per l'acquisizione dei dati isometrici e pluviometrici dal file di progetto ReCity.

- **Matplotlib**: Libreria di visualizzazione dati, utilizzata per creare grafici che mostrano le previsioni delle serie temporali.

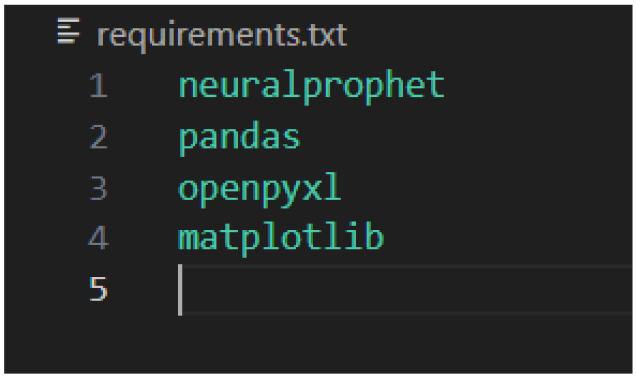

Figura 4.9: File requirements.txt

Queste librerie sono fondamentali per il funzionamento del codice implementato per il progetto ReCity. La loro inclusione nel file *requirements.txt* garantisce una facile riproducibilità dell'ambiente di sviluppo e facilita la condivisione del progetto con altri ricercatori e sviluppatori.

La corretta gestione delle dipendenze è un aspetto cruciale per garantire la solidità del progetto e la replicabilità delle analisi. Il file *requirements.txt* rappresenta uno strumento chiave per raggiungere questi obiettivi, facilitando la corretta installazione delle librerie necessarie.

4.6 "Build" dell'immagine Docker

Una volta creati il file con il codice per le previsioni, il file Docker e il file requiremnts.txt con le librerie, sono stati seguiti i seguenti passaggi per la build dell'immagine Docker contenente l'ambiente di esecuzione del progetto:

- **Preparazione**: Prima di iniziare la build dell'immagine Docker, è stato aperto un terminale e ci si è spostati nella directory in cui è situato il file Dockerfile. Questo è il file di configurazione essenziale per la creazione dell'ambiente di esecuzione isolato.
- **Build dell'Immagine Docker**: Successivamente, è stata eseguita la build dell'immagine Docker utilizzando il seguente comando:

  ```
  docker build -t re-city-docker .
  ```

  Il parametro -t assegna un nome all'immagine, mentre il punto finale indica che la build deve avvenire nella directory corrente e "re-city-docker" è il nome assegnato all'immagine docker.
- **Avvio del Container Docker**: Dopo la corretta costruzione dell'immagine Docker, è stato possibile avviare un container Docker utilizzando il

comando seguente:

```
docker run -p 4000:80 re-city-docker
```

Il parametro -p consente di mappare una porta del sistema host a una porta del container, consentendo così la comunicazione esterna con l'applicazione in esecuzione all'interno del container.

- **Accesso al Terminale del Container**: Per eseguire comandi all'interno del container Docker in esecuzione, è stato possibile aprire una nuova sessione terminale all'interno del container utilizzando il seguente comando e l'id del container:

```
docker exec -it container-ID /bin/bash
```

Questo permette l'esecuzione di comandi direttamente nell'ambiente isolato del container.

- **Navigazione alla Directory di Lavoro**: Dopo l'accesso al terminale del container, è stata eseguita la navigazione alla directory in cui è situato lo script Python. Questo passaggio è essenziale per l'esecuzione corretta dello script all'interno dell'ambiente di sviluppo isolato.

```
cd /app
```

- **Esecuzione dello Script Python**: Infine, lo script Python è stato eseguito all'interno del container utilizzando il seguente comando:

```
python data.py
```

Questo assicura che lo script venga eseguito nell'ambiente di sviluppo isolato, senza interferenze esterne.

In questo modo, l'uso di Docker ha semplificato la gestione delle dipendenze e ha garantito la corretta esecuzione del progetto ReCity in un ambiente controllato e isolato.

Capitolo 5

Implementazione e Deployment su AWS

In questo capitolo, si esaminerà l'implementazione pratica del modello di previsione utilizzando il framework NeuralProphet, la creazione di un ambiente containerizzato con Docker, e il deployment dell'applicazione su Amazon Web Services (AWS). Verranno descritti dettagliatamente i passaggi per la gestione dei dati, la costruzione e il push dell'immagine Docker su Amazon Elastic Container Registry (ECR), e la configurazione di una funzione Lambda per eseguire il modello in risposta agli eventi S3.

5.1 Configurazione AWS

Il primo passo è stato configurare l'Interfaccia della linea di comando AWS (AWS CLI, ovvero AWS Command Line Interface), uno strumento unificato che consente di gestire i servizi AWS. Scaricando e configurando questo unico strumento, è possibile controllare molti servizi AWS dalla riga di comando nonché automatizzarli mediante script.

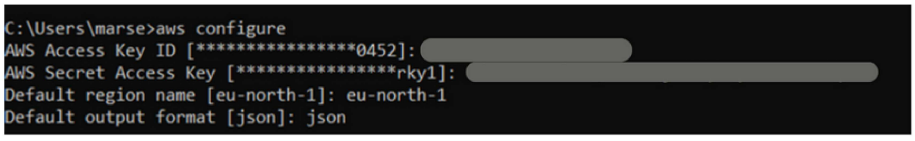

Figura 5.1: Configurazione AWS CLI

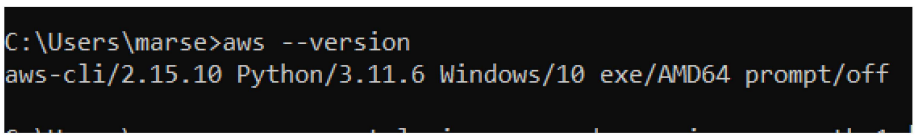

Figura 5.2: Verifica versione AWS

5.1.1 Creazione bucket S3 e caricamento dei dati

Successivamente è stato creato un bucket in S3 denominato "recitybucket" dove è stato caricato il file excel contenente i dati.

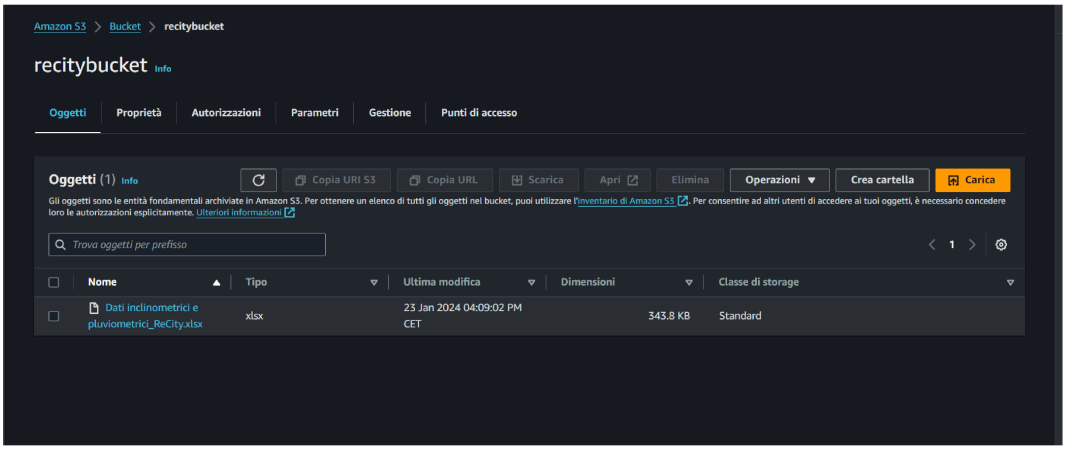

Figura 5.3: Bucket S3

5.1.2 Creazione di un Repository ECR

Il passo successivo è stato quello di creare una repository in ECR per il contenimento dell'immagine docker.

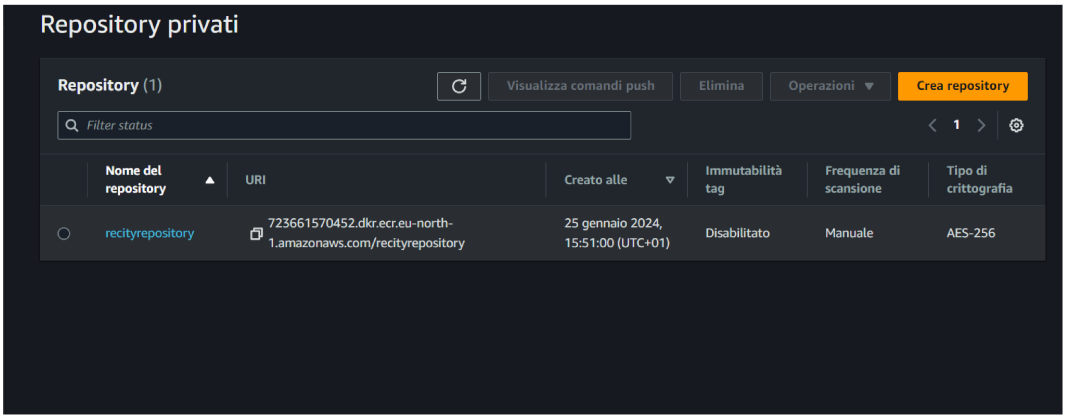

Figura 5.4: Creazione repository ECR

5.1.3 Build e pubblicazione dell'immagine Docker

Per la realizzazione dell'immagine Docker, prima di tutto si è dovuto creare un Dockerfile che consentisse l'integrazione con AWS, infatti AWS e Docker, hanno collaborato per realizzare un'esperienza per sviluppatori semplificata che permette di implementare e gestire direttamente i container su Amazon ECR utilizzando gli strumenti Docker. Nel Dockerfile, in particolare, è stato necessario specificare: la directory della nostra funzione contenente la Rete Neurale,

l'immagine di base da cui partire (nel nostro caso Python3.8), l'installazione delle dipendenze di compilazione per **aws-lambda-cpp** (libreria per sviluppare funzioni AWS Lambda in C++), l'installazione del pacchetto **awslambdaric**, (componente di runtime per le funzioni AWS Lambda scritte in Python), ed infine, l'installazione dei pacchetti utilizzati, contenuti all'interno del file di testo "requirements.txt".

Dockerfile > ...

```
# Define function directory
ARG FUNCTION_DIR="/data"

FROM python:3.8 as build-image

# Install aws-lambda-cpp build dependencies
RUN apt-get update && \
  apt-get install -y \
  g++ \
  make \
  cmake \
  unzip \
  libcurl4-openssl-dev

# Include global arg in this stage of the build
ARG FUNCTION_DIR
# Create function directory
RUN mkdir -p ${FUNCTION_DIR}

#Copy function code
COPY . ${FUNCTION_DIR}

#Install the runtime interface client
RUN pip install \
        --target ${FUNCTION_DIR} \
        awslambdaric

# Multi-stage build: grab a fresh copy of the base image
FROM python:3.8

# Include global arg in this stage of the build
ARG FUNCTION_DIR
# Set working directory to function root directory
WORKDIR ${FUNCTION_DIR}

# Copy in the build image dependencies
COPY --from=build-image ${FUNCTION_DIR} ${FUNCTION_DIR}
COPY requirements.txt ${FUNCTION_DIR}
COPY python .

RUN pip3 install -r requirements.txt --target ${FUNCTION_DIR}

ENTRYPOINT [ "/usr/local/bin/python", "-m", "awslambdaric" ]
CMD [ "data.handler" ]
```

Figura 5.5: Dockerfile

Per completare il processo di autenticazione nel registro dei contenitori di Amazon Elastic Container Registry (ECR) utilizzando Docker è stato usato il seguente comando :

```
aws ecr get-login-password –region eu-north-1 | docker login –username
AWS –password-stdin 723661570452.dkr.ecr.eu-north-1.amazonaws.com
```

Per effettuare il login, invece, è stato usato il comando:

```
aws ecr get-login-password –region eu-north-1 | docker login –username
AWS –password-stdin 723661570452.dkr.ecr.eu-north-1.amazonaws.com
```

Si è poi passati alla costruzione dell'immagine del contenitore con le istruzioni fornite nel Dockerfile nella directory corrente assegnando il tag specificato con il comando :

```
docker build -t 723661570452.dkr.ecr.eu-north-1.amazonaws.com/recityrepository
```

Figura 5.6: Build immagine Docker

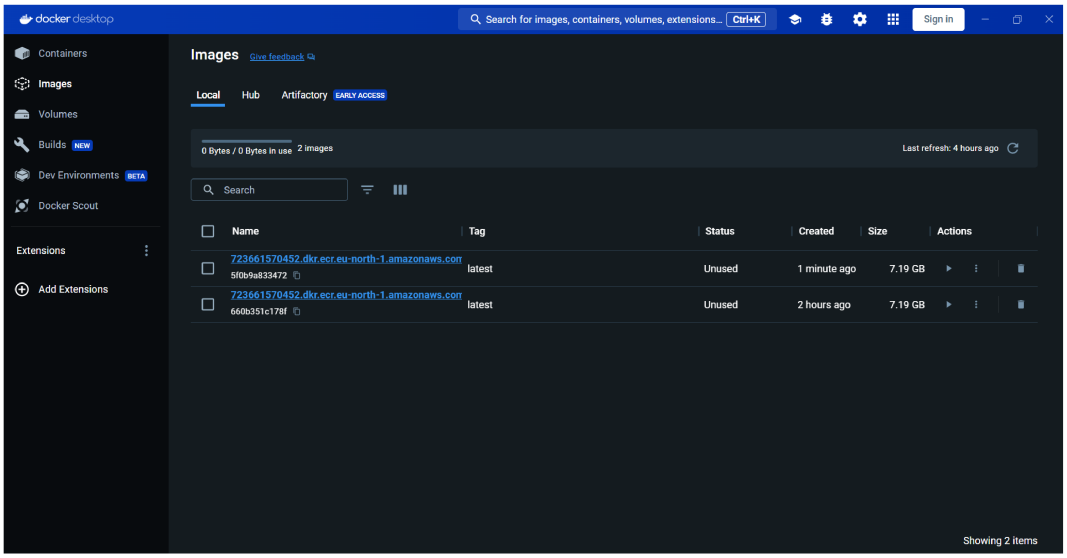

Figura 5.7: Build immagine Docker

L'immagine del contenitore appena costruita è stata poi caricata nel repository specificato nel registro dei contenitori remoto, rendendola cosi disponibile per l'uso e la distribuzione su altri ambienti Docker con il comando :

```
docker push 723661570452.dkr.ecr.eu-north-1.amazonaws.com/recityrepository
```

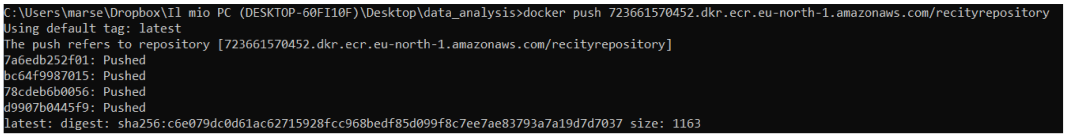

Figura 5.8: Docker push

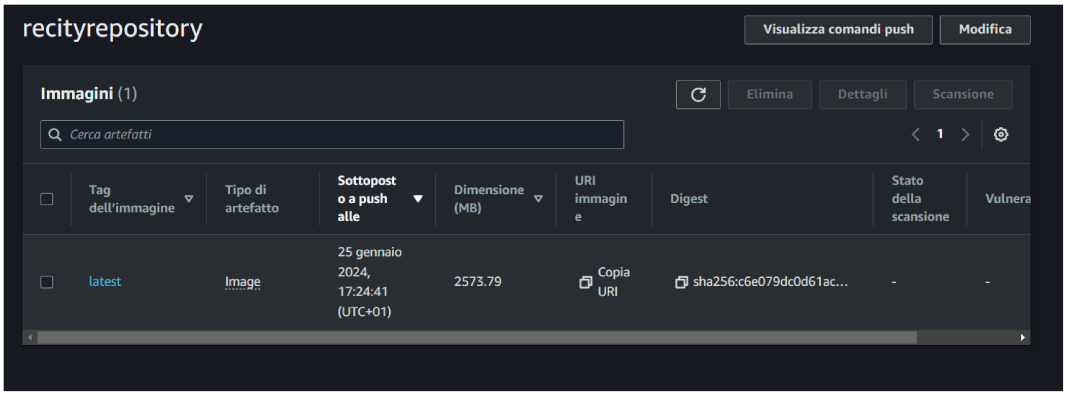

Figura 5.9: Immagine Docker caricata nella repository ECR

5.1.4 Configurazione della Funzione Lambda

La creazione di una Funzione Lambda all'interno dell'ambiente AWS Lambda è un passo cruciale nel processo di sviluppo di applicazioni serverless. Il punto di partenza di questo processo è l'accesso alla console Lambda attraverso l'interfaccia utente di AWS. Questo ambiente fornisce una visione completa delle funzioni Lambda esistenti e permette la creazione di nuove funzioni. Dopo aver avviato il processo di creazione, è necessario configurare la Funzione Lambda con informazioni essenziali e quindi scegliendo il nome della funzione, il runtime e il tipo di origine.

Il codice della Funzione Lambda gestisce due operazioni principali:

Salvataggio dei Dati in Amazon S3

All'arrivo di un evento, il codice serializza i dati in formato JSON e li memorizza all'interno di un bucket S3 specifico. Questa operazione garantisce la persistenza dei dati e ne permette l'accesso futuro.

Politiche di Ruolo per la funzione Lambda

Nel contesto dell'architettura serverless di Amazon Web Services (AWS), la definizione di politiche di ruolo per le Funzioni Lambda riveste un ruolo cruciale nell'assicurare l'accesso appropriato alle risorse e ai servizi AWS.

Le politiche di ruolo definiscono le autorizzazioni e i privilegi di accesso di una funzione Lambda ai servizi e alle risorse AWS. Questo aspetto è fondamentale per garantire che la funzione Lambda possa accedere e interagire in modo sicuro e appropriato con le risorse di cui ha bisogno durante l'esecuzione.

Le politiche di ruolo associate alla funzione Lambda usate sono state :

- **AmazonS3FullAccess**

  Questa politica fornisce accesso completo a tutti i servizi e le risorse di Amazon S3, consentendo operazioni di lettura, scrittura, eliminazione e gestione degli oggetti all'interno dei bucket S3.

- **AmazonS3ReadOnlyAccess**

  Questa politica concede solo accesso in sola lettura a Amazon S3, consentendo la lettura degli oggetti all'interno dei bucket S3 ma impedendo la modifica o l'eliminazione di tali oggetti.

- **AWSAppRunnerServicePolicyForECRAccess**

  Questa politica fornisce accesso ai servizi di AWS App Runner e consente l'accesso ai repository di Elastic Container Registry (ECR) necessari per il deploy delle applicazioni.

- **AWSLambdaBasicExecutionRole**

  Una politica predefinita fornita da AWS per le Funzioni Lambda e fornisce le autorizzazioni di base necessarie per l'esecuzione delle funzioni, inclusi l'accesso ai log di CloudWatch e l'accesso a risorse S3 specifiche.

- **CloudWatchFullAccessV2**

  Questa politica fornisce accesso completo a CloudWatch, consentendo l'accesso ai log, alle metriche e alle altre funzionalità di monitoraggio e gestione offerte da CloudWatch.

- **CloudWatchLogsFullAccess**

  Questa politica concede accesso completo ai log di CloudWatch, consentendo operazioni di lettura, scrittura, eliminazione e gestione dei log all'interno di CloudWatch Logs.

5.1.4.1 Avvio di un Workflow con AWS Step Functions

Successivamente, la Funzione Lambda avvia un workflow basato su stati utilizzando AWS Step Functions. Questo processo di orchestrazione può coinvolgere diverse azioni e decisioni, fornendo un controllo flessibile e reattivo sul flusso di lavoro.

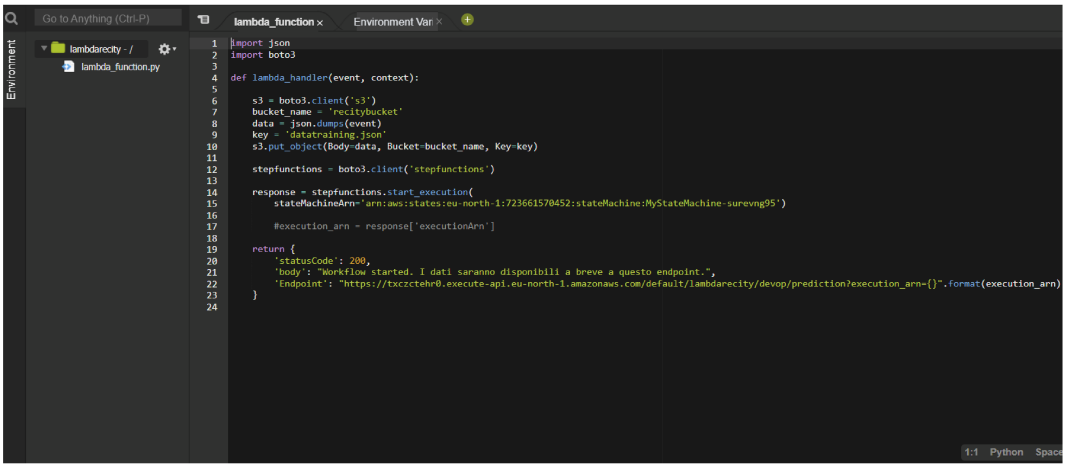

Figura 5.10: Funzione Lambda

5.1.5 Configurazione del Workflow con AWS Step Functions

Nel panorama dello sviluppo delle applicazioni serverless offerto da Amazon Web Services (AWS), l'orchestrazione dei workflow tramite AWS Step Functions rappresenta un'opzione potente per coordinare le attività e le interazioni tra i vari servizi cloud.

L'obiettivo di questo workflow è quello di integrare e coordinare l'esecuzione di una Funzione Lambda all'interno di un flusso di lavoro più ampio. Il workflow sarà responsabile dell'invocazione della Funzione Lambda specificata e della gestione dei dati risultanti, garantendo un'orchestrazione efficace delle attività.

Definizione del Workflow

La definizione del workflow, espressa in JSON, specifica i seguenti dettagli:

Il workflow inizia con l'attività "InvokeNeuralprophetLambda", che invoca una Funzione Lambda specifica.

Invocazione della Funzione Lambda

Durante l'attività "InvokeNeuralprophetLambda", viene invocata la Funzione Lambda identificata dall'ARN specificato. Il risultato dell'invocazione viene salvato nel percorso `"$.prediction_data"`.

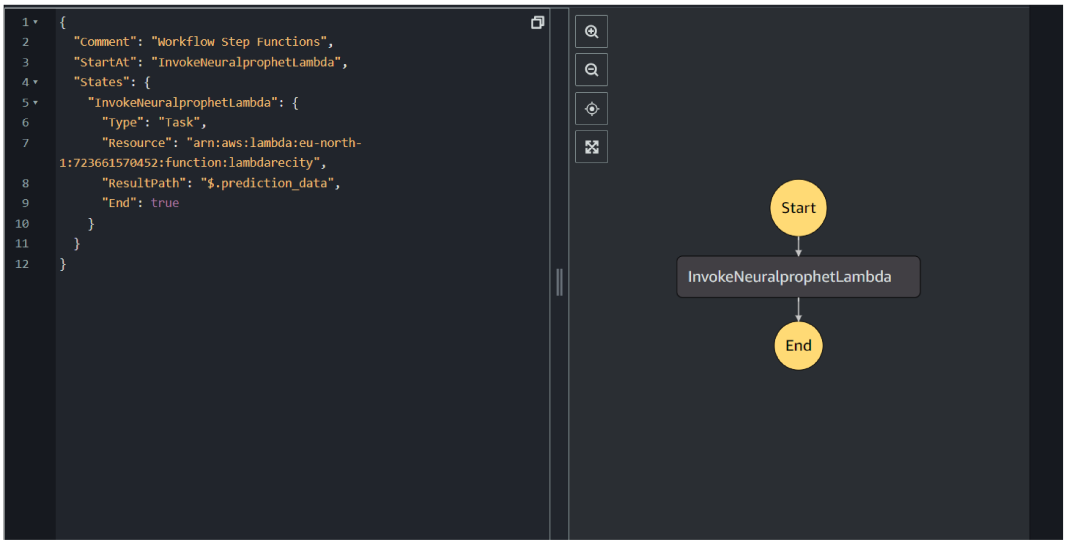

Figura 5.11: Step Functions

5.1.6 Configurazione API Gateway

Nell'architettura serverless di Amazon Web Services (AWS), l'uso di API Gateway come trigger per le funzioni Lambda rappresenta un meccanismo fondamentale per esporre e gestire le funzionalità delle applicazioni tramite API REST.

L'introduzione di un'API Gateway come interfaccia di *front-end* per le funzioni Lambda consente di esporre le funzionalità dell'applicazione tramite *endpoint* HTTP, facilitando l'integrazione con client esterni e altre applicazioni. L'obiettivo è fornire un'interfaccia di comunicazione robusta e scalabile per le funzioni Lambda.

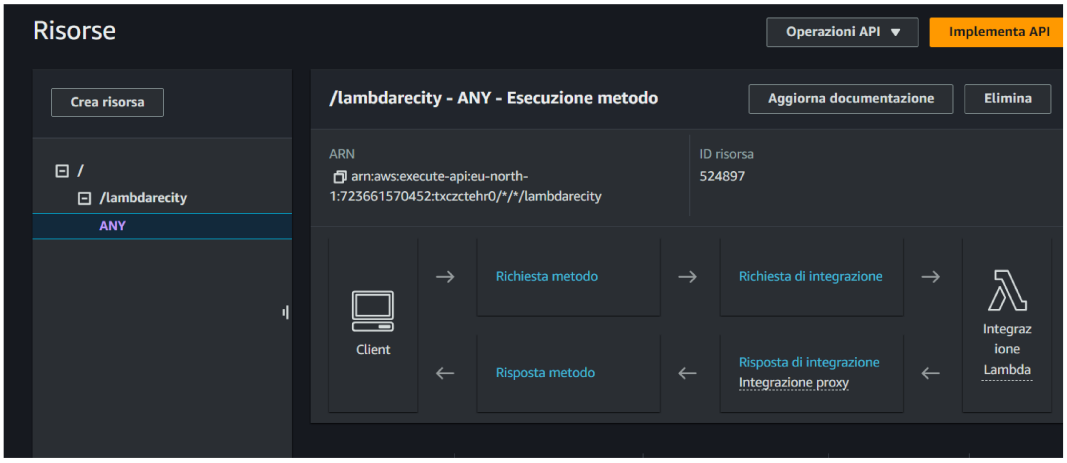

Figura 5.12: API funzione Lambda

Forecast Results

Date	Forecasted Value	Trend	Yearly Seasonality	Weekly Seasonality
2015-12-22 00:00:00	nan	nan	nan	nan
2015-12-23 00:00:00	0.004771675914525986	-0.030056409537792206	0.000744389311876148	-4.926873316435376e-06
2015-12-24 00:00:00	0.0047708675265312195	-0.030048195272684097	0.00073022968762368	-3.8364629517673166e-07
2015-12-25 00:00:00	0.004791982471942902	-0.03003998100757599	0.000717147602699697	3.627871819844586e-06
2015-12-26 00:00:00	0.004766847938299179	-0.03003176674246788	0.0007034339942038059	6.557294909725897e-06
2015-12-27 00:00:00	0.004759069532155991	-0.03002355247735977	0.0006897528073750436	4.2429401219123974e-06
2015-12-28 00:00:00	0.004773572087287903	-0.030015338212251663	0.0006761733093298972	1.5531275039393222e-06
2015-12-29 00:00:00	0.004756152629852295	-0.030007123947143555	0.0006627634284086525	-1.0670714800653514e-05
2015-12-30 00:00:00	0.00475693866610527	-0.029998909682035446	0.000649589637760073	-4.926873316435376e-06
2015-12-31 00:00:00	0.004756823182106018	-0.029990695416927338	0.0006367145688273013	-3.8364629517673166e-07
2016-01-01 00:00:00	0.00475652888417244	-0.02998248115181923	0.0006241982337087393	3.627871819844586e-06
2016-01-02 00:00:00	0.004755575209856033	-0.02997426688671112	0.0006120966863818467	6.557294909725897e-06
2016-01-03 00:00:00	0.0047498345375061035	-0.029966052621603012	0.0006004612660035491	4.2429401219123974e-06
2016-01-04 00:00:00	0.004744235426187515	-0.02995783835649490	0.0005893387133255601	1.5531275039393222e-06
2016-01-05 00:00:00	0.00472966209053993	-0.029949624091386795	0.0005787703557871282	-1.0670714800653514e-05
2016-01-06 00:00:00	0.00477876141667366	-0.02994140982627868	0.0005687918164767325	-4.926873316435376e-06
2016-01-07 00:00:00	0.0047821588814258575	-0.029933195561170578	0.0005594324320554733	-3.8364629517673166e-07
2016-01-08 00:00:00	0.00478566437959671	-0.02992498129606247	0.0005507150781340897	3.627871819844586e-06
2016-01-09 00:00:00	0.004788748919963837	-0.02991676703095436	0.0005426562274806201	6.557294909725897e-06
2016-01-10 00:00:00	0.00478726252913475	-0.029908552765846252	0.0005352655425667763	4.2429401219123974e-06
2016-01-11 00:00:00	0.004786062985658646	-0.02990038500738144	0.0005285450024530292	1.5531275039393222e-06
2016-01-12 00:00:00	0.004730880260467529	-0.029892124235630035	0.0005224902415648103	-1.0670714800653514e-05
2016-01-13 00:00:00	0.004784557968378067	-0.029883909970521927	0.0005170893273316324	-4.926873316435376e-06
2016-01-14 00:00:00	0.004747431725263596	-0.02987569570541382	0.0005123236333020031	-3.8364629517673166e-07
2016-01-15 00:00:00	0.004755500704050064	-0.02986748144030571	0.0005081670824438334	3.627871819844586e-06
2016-01-16 00:00:00	0.004763063043355942	-0.029859267175 1976	0.0005045871366746724	6.557294909725897e-06
2016-01-17 00:00:00	0.004765916615724564	-0.029851052910089493	0.0005015447386540473	4.2429401219123974e-06

Figura 5.13: Previsioni di serie temporali restituite dalla NeuralPorphet

2018-06-15 00:00:00	-0.04057847708463669	-0.040083080530166626	-0.0005667043733410537	3.627871819844586e-06
2018-06-16 00:00:00	-0.040578339248895645	-0.040093135088682175	-0.0005594415124505758	6.557294909725897e-06
2018-06-17 00:00:00	-0.04062863439321518	-0.04010318964719772	-0.000552246521692723	4.2429401219123974e-06
2018-06-18 00:00:00	-0.04063423350453377	-0.04011324420571327	-0.0005451010074466467	1.5531275039393222e-06
2018-06-19 00:00:00	-0.04064939543604851	-0.04012329876422882	-0.0005379862850531936	-1.0670714800653514e-05
2018-06-20 00:00:00	-0.04064660146832466	-0.04013335332274437	-0.000530881981831044	-4.926873316435376e-06
2018-06-21 00:00:00	-0.04064499959349632	-0.04014340788125992	-0.0005237675504758954	-3.8364629517673166e-07
2018-06-22 00:00:00	-0.04059877619147301	-0.04015346243977547	-0.0005166222690604627	3.627871819844586e-06
2018-06-23 00:00:00	-0.040598705410957336	-0.040163516998291016	-0.0005094248335808516	6.557294909725897e-06
2018-06-24 00:00:00	-0.04060380533337593	-0.04017357528209686	-0.0005021544639021158	4.2429401219123974e-06
2018-06-25 00:00:00	-0.04060918465256691	-0.04018362984061241	-0.0004947905545122921	1.5531275039393222e-06
2018-06-26 00:00:00	-0.0406239852309227	-0.04019368439912796	-0.0004873134312219918	-1.0670714800653514e-05
2018-06-27 00:00:00	-0.04066580906510353	-0.04020373895764351	-0.0004797042056452483	-4.926873316435376e-06
2018-06-28 00:00:00	-0.04061844199895859	-0.04021379351615906	-0.0004719454154837876	-3.8364629517673166e-07
2018-06-29 00:00:00	-0.040616560727357864	-0.040223848074674606	-0.000464020820800215	3.627871819844586e-06
2018-06-30 00:00:00	-0.04061558097600937	-0.040233902633190155	-0.000455916189821437	6.557294909725897e-06
2018-07-01 00:00:00	-0.04061965271830559	-0.040243957191705704	-0.0004476190079003572	54.2429401219123974e-06
2018-07-02 00:00:00	-0.040623895823955536	-0.04025401175022125	-0.0004391192051116377	1.5531275039393222e-06
2018-07-03 00:00:00	-0.0406825877726078	-0.0402640663087368	-0.0004304086323827505	-1.0670714800653514e-05
2018-07-04 00:00:00	-0.04067796841263771	-0.04027412086725235	-0.0004214818181935698	-4.926873316435376e-06
2018-07-05 00:00:00	-0.04067433625459671	-0.0402841754257679	-0.0004123362596146762	4-3.8364629517673166e-07
2018-07-06 00:00:00	-0.04067101329565048	-0.04029422998428345	-0.0004029715491924435	3.627871819844586e-06
2018-07-07 00:00:00	-0.04069111868739128	-0.040304284542798996	-0.0003933905391022563	6.557294909725897e-06
2018-07-08 00:00:00	-0.04067113623023033	-0.04031434282660484	-0.0003835988463833928	4.2429401219123974e-06
2018-07-09 00:00:00	-0.040673889219760895	-0.04032439738512039	-0.0003736049402505159	41.5531275039393222e-06
2018-07-10 00:00:00	-0.04070854187011719	-0.04033445194363594	-0.0003634201711975038	-1.0670714800653514e-05
2018-07-11 00:00:00	-0.040702491998672485	-0.04034450650215149	-0.0003530587418936193	-4.926873316435376e-06
2018-07-12 00:00:00	-0.040697481483221054	-0.04035456106066704	-0.0003425373288337141	3-3.8364629517673166e-07
2018-07-13 00:00:00	-0.04069286212325096	-0.04036461561918259	-0.0003318756353110075	3.627871819844586e-06

Figura 5.14: Previsioni di serie temporali restituite dalla NeuralProphet

Metrica	Errore
Mean Absolute Error (MAE)	0.00287006971548763
Mean Squared Error (MSE)	1.731536841469247e-05
R-squared (R2)	-0.5752688883725035

Tabella 5.1: Calcolo errore MAE, MSE e R2 restituito della NeuralProphet

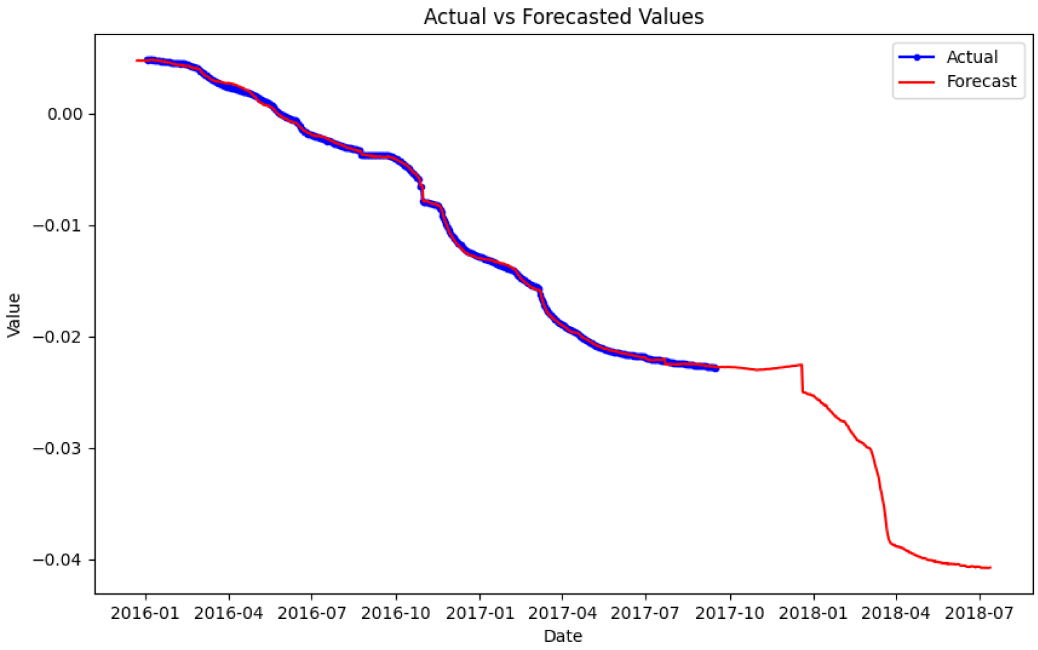

Figura 5.15: Grafico del confronto valori delle predizioni e valori reali restituito della NeuralProphet

Capitolo 6

Conclusioni e sviluppi futuri

6.1 Conclusioni

Lo sviluppo del progetto di sistema di architetture serverless per la predizione di serie temporali a partire da materiali self-sensing nell'ambito del progetto ReCity ha dato le basi per la creazione di un sistema basato su cloud per agevolare la gestione e l'analisi di dati temporali complessi.

Le architetture serverless, implementate tramite servizi come AWS Lambda, Step Functions, e altri componenti AWS, hanno dimostrato di essere efficaci nel contesto del progetto della predizione delle serie temporali basate su dati provenienti dai materiali self-sensing. Queste architetture hanno offerto flessibilità, scalabilità e affidabilità, consentendo un'implementazione efficiente delle procedure di predizione.

NeuralProphet ha svolto un ruolo fondamentale, fornendo un'interfaccia semplice ed efficiente per la previsione delle serie temporali. L'utilizzo di NeuralProphet ha consentito di modellare le complesse dinamiche temporali presenti nei dati dei materiali self-sensing, migliorando significativamente la precisione delle previsioni.

L'integrazione di Docker e Amazon Elastic Container Registry (ECR) ha semplificato la gestione e la distribuzione delle applicazioni create, offrendo un ambiente di esecuzione consistente e portatile. Questo approccio ha facilitato lo sviluppo, il test e il rilascio delle soluzioni, contribuendo alla coerenza e all'affidabilità del sistema creato.

I risultati delle analisi condotte hanno messo in luce l'importanza dell'approccio serverless e degli strumenti come NeuralProphet nel contesto della previsione delle serie temporali basate sui dati dei materiali self-sensing. Lo studio emerso offre importanti indicazioni per migliorare la comprensione e la gestione di tali materiali, oltre a sviluppare approcci più efficienti per la predizione delle serie temporali.

Tuttavia, durante il corso del progetto, sono emerse diverse sfide che hanno richiesto attenzione e risoluzione. È stato evidente che vi sono limitazioni e aree specifiche in cui ulteriori ricerche e sviluppi potrebbero portare a notevoli miglioramenti. In particolare, è stato identificato il bisogno di affrontare le complessità legate alla gestione dei dati provenienti dai materiali self-sensing e di ottimizzare le tecniche utilizzate per la predizione delle serie temporali.

L'argomento di tesi ha rappresentato per me una fonte di grande interesse e stimolo, in quanto mi ha fornito l'opportunità di esplorare e approfondire le conoscenze relative a nuove tecnologie nel campo della programmazione. Questa esperienza non solo mi ha permesso di ampliare le mie competenze, ma mi ha anche offerto l'occasione di mettere in pratica le nozioni apprese, consolidando il mio percorso universitario.

6.2 Sviluppi futuri

Prevedere lo sviluppo futuro di AWS e del cloud computing, nonché del serverless computing in generale, comporta una certa dose di speculazione, ma è possibile formulare ipotesi basate sulle tendenze attuali e sulle esigenze del mercato.

L'espansione dei servizi cloud è prevista continuare nel tempo. AWS e altri fornitori sembrano intenti ad impegnarsi ad ampliare la loro gamma di servizi per soddisfare una varietà sempre crescente di esigenze aziendali, con possibilità di sviluppare soluzioni specializzate per settori specifici come la sanità, l'automotive, e l'industria manifatturiera.

Il miglioramento delle prestazioni e dell'affidabilità dei servizi cloud sarà un obiettivo costante. Con l'aumentare della concorrenza nel settore, ci si attende un costante incremento delle prestazioni, con tempi di risposta più rapidi, maggiore disponibilità e una migliore gestione delle risorse.

Il serverless computing continuerà a crescere in popolarità grazie alla sua capacità di consentire alle aziende di concentrarsi sullo sviluppo delle applicazioni senza preoccuparsi della gestione dell'infrastruttura sottostante. Si prevede che sempre più carichi di lavoro verranno spostati verso architetture serverless per ridurre i costi operativi e semplificare lo sviluppo e la gestione delle applicazioni.

L'integrazione con l'edge computing diventerà sempre più importante, data la crescente adozione di tecnologie come l'Internet of Things (IoT). AWS e altri fornitori potrebbero espandere le loro offerte per includere servizi edge computing che consentono alle aziende di elaborare i dati più vicino alla loro origine, riducendo la latenza e migliorando le prestazioni complessive delle applicazioni.

Infine, il focus sulla sicurezza e sulla conformità continuerà ad essere prioritario. Con l'aumento delle minacce alla sicurezza informatica e l'introduzione di nuove normative sulla privacy dei dati, ci si aspetta che AWS e altri fornitori mettano sempre più l'accento sulla sicurezza e sulla conformità dei loro servizi, introducendo nuove funzionalità di sicurezza e implementando procedure di audit più rigorose.

In sintesi, il futuro di AWS, del cloud computing e del serverless computing sembra essere caratterizzato da un continuo sviluppo tecnologico e dall'adattamento alle esigenze in evoluzione del mercato e delle aziende.

Per quanto riguarda lo sviluppo futuro di NeuralProphet, si prevedono diverse evoluzioni:

- **Miglioramenti nell'algoritmo di previsione**
  Si prevede che NeuralProphet continuerà a perfezionare la precisione delle previsioni temporali mediante l'ottimizzazione degli algoritmi di apprendimento automatico utilizzati e l'integrazione di nuove tecniche di modellazione.

- **Espansione delle funzionalità**
  Potrebbero essere introdotte nuove funzionalità per gestire dati temporali complessi, come la gestione di dati mancanti, la rilevazione automatica di anomalie e la gestione di serie storiche multivariabili.

- **Integrazione con altre librerie e strumenti**
  NeuralProphet potrebbe essere integrato con altre librerie e strumenti di data science e machine learning per fornire una suite completa di funzionalità per l'analisi e la previsione dei dati.

- **Miglioramento della scalabilità e della velocità di esecuzione**
  Considerando l'aumento delle dimensioni dei dati e delle richieste di previsione in tempo reale, NeuralProphet potrebbe concentrarsi sull'ottimizzazione della scalabilità e della velocità di esecuzione dei suoi algoritmi.

- **Supporto per nuovi tipi di dati e domini applicativi**
  Potrebbero essere introdotti nuovi modelli e tecniche per gestire tipi di dati e domini applicativi specifici, come le serie temporali geospaziali o le previsioni finanziarie.

In generale, NeuralProphet rimarrà una risorsa importante per coloro che lavorano con l'analisi e la previsione dei dati temporali. Il suo sviluppo futuro sarà influenzato dall'evoluzione delle esigenze del settore e dalle innovazioni nel campo del Machine Learning e dell'analisi dei dati.

In conclusione, questo studio ha esplorato l'utilizzo delle architetture serverless per la predizione di serie temporali da materiali self-sensing nel contesto del progetto ReCity. L'integrazione di strumenti come AWS, NeuralProphet e Docker ha dimostrato l'efficacia di tali approcci, nonostante le sfide legate alla complessità dei dati e alla precisione delle previsioni.

Bibliografia

[1] Oskar Triebe. Neuralprophet: A simple forecasting package. https://github.com/ourownstory/neural_prophet, 2023.

[2] Fortinet. What is serverless computing? https://www.fortinet.com/it/resources/cyberglossary/serverless-computing, 2024.

[3] Chiradeep BasuMallick. What is serverless? definition, architecture, examples, and applications. https://www.spiceworks.com/tech/devops/articles/what-is-serverless/, 2022.

[4] Sachin Mamoru. Serverless computing with aws lambda: Harnessing the power of event-driven architecture. https://medium.com/cloud-native-daily/serverless-computing-with-aws-lambda-harnessing-the-power-of-\\event-driven-architecture-ec55322a62c9, 2023.

[5] neuralmagic. Deploy serverless machine learning inference on aws with deepsparse. https://neuralmagic.com/blog/deploy-serverless-machine-learning-inference-on-aws-with-deepsparse/, 2023.

[6] Rutan Bhattacharyya Ashish Kumar Srivastav Dheeraj Vaidya, CFA, FRM. Time series analysis. https://www.wallstreetmojo.com/time-series-analysis/, 2023.

[7] Prakhar Gajendrakar Raisa Ali Dheeraj Vaidya, CFA, FRM. Time series. https://www.wallstreetmojo.com/time-series/, 2023.

[8] Shailesh Shukla. Various techniques to detect and isolate time series components using python. https://www.analyticsvidhya.com/blog/2023/02/various-techniques-to-detect-and-isolate-time-series-components\\-using-python/, 2023.

[9] Dr. Ganapathi Pulipaka. Neuralprophet - a neural network based time-series model. https://www.facebook.com/photo/?fbid=2869337873350131&set=pcb.2869338103350108, 2020.

[10] DataTechNotes. Regression model accuracy (mae, mse, rmse, r-squared) check in r. https://www.datatechnotes.com/2019/02/regression-model-accuracy-mae-mse-rmse.html, 2019.

[11] David. Docker announces new local + cloud products to accelerate delivery of secure apps. https://cloudcow.com/content/docker-announces-new-local-cloud-products-to-accelerate-delivery\-of-secure-apps/, 2023.

[12] Saurabh Singh. How to build application inside and outside docker. dockerfile structure and commands. https://medium.com/@saurabh.singh0829/how-to-build-application-inside-and-outside-docker-dockerfile\-structure-and-commands-f542b58cd830, 2018.

[13] The python logo. https://www.python.org/community/logos/.

[14] Riccardo Esposito. Linguaggi di programmazione: più usati, richiesti e ben pagati oggi. https://seoriented.it/linguaggi-programmazione-richiesti/, 2022.

[15] Wikipedia. Python. https://it.wikipedia.org/wiki/Python, 2023.

Elenco delle figure