



# Università del Salento

Dipartimento di Ingegneria dell'Innovazione

Corso di Laurea Triennale in Ingegneria  
dell'Informazione

---

TESI DI LAUREA IN PRINCIPI DI INGEGNERIA DEL  
SOFTWARE

**Una tecnica di load-shifting basata su architetture Cloud  
a supporto di applicazioni di Green AI**

Relatore

*Prof. Roberto Vergallo*

Laureando

*Profilo Marco*

*Matricola n° 20054679*

---

ANNO ACCADEMICO 2022/2023

A te che sei andato via prima di vedere il mio progetto realizzato.

A te che più di chiunque altro avresti voluto esserci quest'oggi nel giorno del tuo compleanno.

A mio nonno Prof. Calogero Di Salvo.

# Indice

<b>1</b>	<b>Introduzione</b>	<b>5</b>
1.1	Contesto . . . . .	5
1.2	Obiettivi . . . . .	6
<b>2</b>	<b>Stato dell'Arte</b>	<b>8</b>
2.1	Review Scientifica . . . . .	8
2.2	Review Tecnologica . . . . .	11
2.2.1	Machine Learning . . . . .	11
2.2.2	Architetture Cloud . . . . .	16
2.2.3	Amazon Web Services . . . . .	18
2.2.4	Linguaggio di Programmazione . . . . .	19
2.2.5	Moduli di serializzazione per Machine Learning . . . . .	20
2.2.6	API Carbon Intensity . . . . .	21
2.2.7	Secure Shell . . . . .	21
<b>3</b>	<b>Studio di fattibilità</b>	<b>22</b>
3.1	Architettura logica . . . . .	22
3.2	Scelte tecnologiche . . . . .	23
3.2.1	Linguaggio di Programmazione . . . . .	23
3.2.2	Librerie . . . . .	23
3.2.3	AWS . . . . .	24
3.2.4	SSH . . . . .	24
3.3	Architettura Fisica . . . . .	24

<b>4</b>	<b>Progettazione</b>	<b>27</b>
4.1	Diagramma dei casi d'uso . . . . .	27
4.2	Diagramma delle classi . . . . .	28
4.3	Diagrammi di sequenza . . . . .	29
<b>5</b>	<b>Sviluppo</b>	<b>31</b>
5.1	Package Joblib Estimator Function . . . . .	31
5.1.1	Preparazione dei Dati . . . . .	31
5.1.2	Inizio dell'addestramento . . . . .	32
5.1.3	Ripresa dell'addestramento . . . . .	33
5.2	Implementazione nel contesto principale . . . . .	35
5.3	Gestione delle eccezioni . . . . .	38
<b>6</b>	<b>Test e analisi delle performance</b>	<b>40</b>
6.1	Test Funzionali . . . . .	40
6.2	Unit Test . . . . .	43
<b>7</b>	<b>Conclusioni e Sviluppi futuri</b>	<b>44</b>
7.1	Conclusioni finali . . . . .	44
7.2	Sviluppi futuri . . . . .	45
	<b>Ringraziamenti</b>	<b>47</b>

# Capitolo 1

## Introduzione

### 1.1 Contesto

Negli ultimi anni l'interesse per l'integrazione dell'Intelligenza Artificiale (AI), cioè il campo di studio e sviluppo che si occupa di creare macchine e sistemi capaci di emulare l'intelligenza umana, e della sostenibilità sta crescendo rapidamente.

L'applicazione dell'AI, in particolare delle tecniche di apprendimento automatico (Machine Learning), richiede notevoli risorse computazionali, il che comporta un consumo significativo di energia. Questo elevato consumo energetico è rappresentato da diverse ragioni quali: calcoli intensivi, i quali richiedono un'elevata potenza di calcolo, hardware specializzato, grandi dimensioni dei dataset, scalabilità.

Le tecnologie Cloud sono un insieme di servizi, risorse e infrastrutture informatiche fornite in rete. Aniché ospitare e gestire fisicamente server in loco, queste ultime consentono di accedere a tali risorse in remoto predisponendo di una connessione Internet. L'adozione di tecnologie Cloud sta crescendo in maniera costante, offrendo una soluzione per la gestione e l'elaborazione scalabile di grandi quantità di dati.

La tesi si propone di esplorare e sviluppare una tecnica di load-shifting

basata su architetture Cloud, che supporti applicazioni di green AI. Il load-shifting, o spostamento del carico, è la pratica di distribuire in modo ottimale il carico di lavoro tra diverse risorse di elaborazione al fine di massimizzare l'efficienza energetica complessiva, consentendo di ridurre il consumo di energia e le emissioni di carbonio associate alle attività di elaborazione delle applicazioni AI.

Grazie alle architetture Cloud, il load-shifting prevede la capacità di spostare dinamicamente il carico di lavoro delle applicazioni tra diverse macchine virtuali disponibili nella rete Cloud. Attraverso la corretta distribuzione di esso, è possibile ridurre l'utilizzo di risorse che richiedono l'uso intensivo di CPU<sup>1</sup> e GPU<sup>2</sup>.

In questa ricerca il load-shifting verrà utilizzato per spostare il lavoro di un algoritmo di Machine Learning su un server con una minore carbon intensity<sup>3</sup>, espressa come la quantità di CO<sub>2</sub><sup>4</sup> emessa per generare energia.

## 1.2 Obiettivi

L'obiettivo della tesi è sviluppare e valutare un algoritmo di load-shifting che usi la carbon intensity per scegliere il server su cui trasferire il lavoro di Machine Learning, precedentemente interrotto.

In particolare, gli obiettivi di questo lavoro di tesi sono:

- Avviare l'algoritmo di training, basato su un Isolation Forest, per il rilevamento di frodi bancarie, su un server con un'intensità di carbonio ben precisa;

---

<sup>1</sup>Central Processing Unit; è l'unità centrale di elaborazione di un computer o di un sistema informatico

<sup>2</sup>Graphics Processing Unit; è un processore specializzato nella gestione e nell'elaborazione delle immagini e dei grafici

<sup>3</sup>intensità di carbonio

<sup>4</sup>diossido di Carbonio

- Ricerare un server con una carbon intensity minore di quella del server precedente;
- Se la ricerca ha esito positivo, mettere in pausa lo stato di training<sup>5</sup> e salvare i dati del processo fino al momento dell'interruzione;
- Spostare i dati e l'algoritmo sul nuovo server per riprendere il processo di addestramento;
- Concludere l'addestramento ed eseguire le previsioni del modello.

---

<sup>5</sup>addestramento

# Capitolo 2

## Stato dell'Arte

### 2.1 Review Scientifica

Nel presente paragrafo sono riportati esempi di articoli volti ad approfondire scientificamente la problematica legata alla sostenibilità delle tecnologie ICT<sup>1</sup> e il loro impatto sull'ambiente. Secondo un articolo pubblicato sul "*Journal of Cleaner Production*", una celebre rivista scientifica specializzata in tematiche di sostenibilità ambientale, se non controllate, le emissioni di gas serra associate alle tecnologie dell'informazione e della comunicazione potrebbero aumentare dall'1.6% nel 2007 a oltre il 14% entro il 2040 [1]. Questo articolo mette in evidenza l'importante ruolo dei data center nel panorama mondiale delle emissioni di gas serra, poiché sono essenziali per il funzionamento e la distribuzione dell'AI. Tuttavia, misurare le emissioni dei data center può risultare complesso e non uniforme a causa delle diverse regioni in cui si trovano e dei diversi tipi di energia utilizzata (rinnovabile e non rinnovabile). Inoltre, l'articolo delinea due approcci legati all'intelligenza artificiale: GreenAI e RedAI.

GreenAI si concentra sull'ottimizzazione dei risultati tenendo conto del costo computazionale, ponendo pari importanza all'accuratezza e all'efficienza<sup>2</sup>, al

---

<sup>1</sup>Information and Communication Technology; insieme di strumenti e sistemi utilizzati per acquisire, memorizzare, elaborare e trasmettere informazioni

<sup>2</sup>velocità e costo energetico



fine di sviluppare modelli con un impatto ambientale ridotto. D'altra parte, RedAI mira a migliorare l'accuratezza utilizzando una potenza computazionale considerevole, senza preoccuparsi dei relativi costi energetici [2].

Un articolo citato: "*Follow-the-sun (2021). 9 Advantages of FTS*" [3], descrive la strategia del FTS<sup>3</sup> come modello evoluto che consente di spostare il carico di lavoro da un sito all'altro, sfruttando i fusi orari e garantendo assistenza continua ai clienti. L'approccio FTS si basa sul principio di consegnare il lavoro non finito alla sede successiva alla fine della giornata lavorativa, consentendo di sfruttare il ciclo lavorativo diurno di diverse squadre in diverse zone del mondo, riducendo i tempi di attesa e utilizzando energie rinnovabili derivanti dal sole.

Nell'articolo [4], del 20 giugno 2022, si pone l'attenzione su come si stia scoprendo che, man mano che i modelli di apprendimento automatico diventano sempre più grandi, aumenta anche la loro corrispondente impronta di carbonio. Nell'articolo sopra citato si evince anche come un team interdisciplinare dell'Allen Institute for AI, Microsoft, Hebrew University, Carnegie Mellon University e la startup di AI Hugging Face ha scoperto che se si addestrasse il modello NLP<sup>4</sup> in paesi come la Norvegia o la Francia, dove si utilizzano fonti di energia rinnovabile, le emissioni di carbonio potrebbero essere ridotte della metà.

---

<sup>3</sup>Follow The Sun

<sup>4</sup>Natural Language Processing, è un ramo dell'AI che consente ai computer di comprendere, generare e manipolare il linguaggio umano

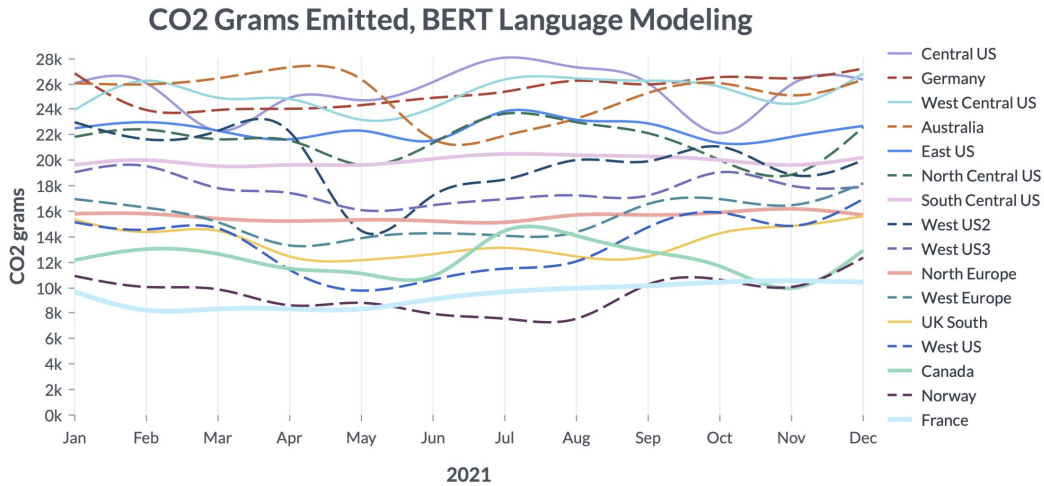


Figura 2.1: CO2 Grams Emitted, BERT Language Modeling

Green AI si concentra sullo sviluppo di tecnologie e pratiche di intelligenza artificiale che riducano il più possibile l'impatto ambientale e promuovano la sostenibilità. L'obiettivo è ridurre il consumo energetico e l'impronta di carbonio, associati ai sistemi di intelligenza artificiale. Gli aspetti chiave della Green AI sono:

- *Efficienza energetica:* promuove lo sviluppo di algoritmi, modelli e infrastrutture hardware a basso consumo energetico;
- *Efficienza dei dati:* riducendo l'elaborazione, la memorizzazione e la trasmissione di dati ridondanti o non necessari si contribuisce a limitare il consumo di energia e le emissioni di carbonio ad esso associate;
- *Ottimizzazione delle risorse:* esplora tecniche per ottimizzare l'uso delle risorse di calcolo, come server e data center, al fine di ridurre il consumo di energia e lo spreco;
- *Gestione del ciclo di vita:* si concentra sull'intero ciclo di vita dei sistemi di intelligenza artificiale.

L'articolo [5] evidenzia, ad esempio, come la quantità di calcolo utilizzata per addestrare i modelli di deep learning è aumentata 300.000 volte in sei anni, come mostrato nel seguente grafico:

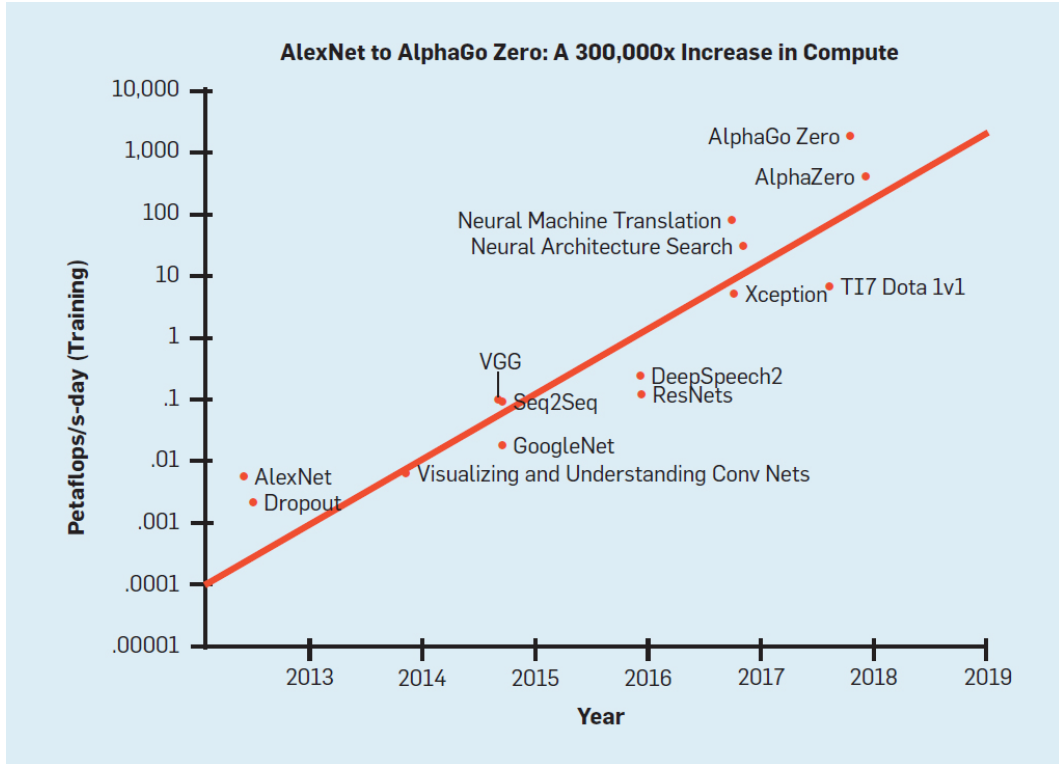


Figura 2.2: Grafico aumento negli anni dei processi di deep learning

## 2.2 Review Tecnologica

In questa sezione vengono descritti gli strumenti e le risorse con i quali è stata effettuata questa ricerca.

### 2.2.1 Machine Learning

I modelli di Machine Learning sono algoritmi informatici che usano dei dati per effettuare stime o decisioni. Quando bisogna migliorare un normale software, il compito viene dato agli utenti. Al contrario, un algoritmo di Machine Learning usa i dati per migliorare un'attività specifica. Sono algoritmi che migliorano man mano che acquisiscono esperienza.

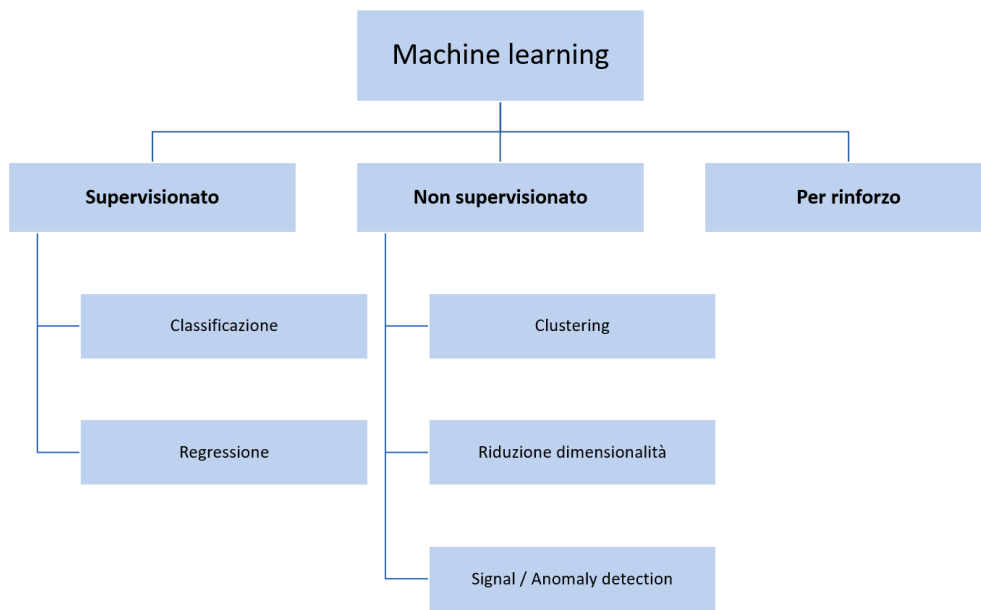


Figura 2.3: Tipi di algoritmi di Machine Learning

### Apprendimento Supervisionato

L'apprendimento supervisionato viene usato quando si vuole mappare l'input alle etichette di output. L'obiettivo è quello di insegnare al modello una funzione che può essere utilizzata per fare previsioni o prendere decisioni su nuovi dati di input. Questi algoritmi vengono utilizzati per risolvere due problemi:

- *Classification*: si occupa di assegnare gli oggetti a una o più categorie predefinite in base alle loro caratteristiche o attributi;
- *Regression*: si occupa di stimare un valore continuo in base alle variabili indipendenti.

### Apprendimento non supervisionato

Nell'apprendimento non supervisionato il modello viene addestrato utilizzando dati di input non strutturati o non etichettati. Con questa tecnica si è in grado di monitorare la struttura dati e di raccogliere informazioni cariche di significato. Le tecniche più comuni sono: *clustering*, *association rule mining*, *dimensionality reduction*, *anomaly detection*.

## Apprendimento per rinforzo

L'apprendimento per rinforzo [12] è un paradigma in cui un agente interagisce con un ambiente e impara a prendere decisioni per massimizzare una ricompensa cumulativa nel corso del tempo. La qualità di un'azione è determinata dalla sua ricompensa, che aumenta all'aumentare della bontà del comportamento. È utilizzato in una vasta gamma di applicazioni, come giochi, robot autonomi, gestione delle risorse ecc.

## Fasi del Machine Learning

Il percorso di un processo di Machine Learning è rappresentato dal seguente schema:

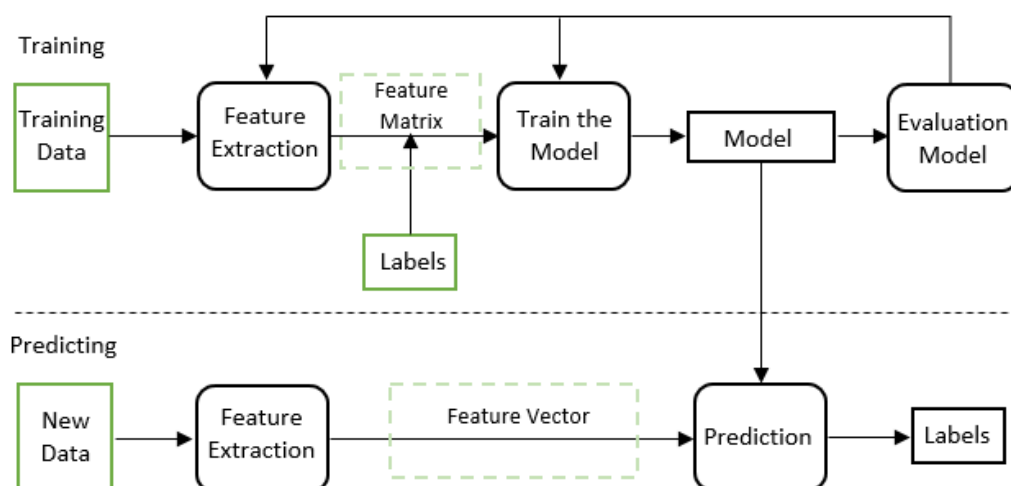


Figura 2.4: Flusso di lavoro del processo di ML

La fase iniziale è caratterizzata dalla raccolta dei dati, che possono essere acquisiti da diverse fonti, come database o file di testo. I dati raccolti potrebbero richiedere una fase di preparazione prima di poter essere utilizzati per il training<sup>5</sup> del modello. Questa fase comprende l'estrazione delle caratteristi-

<sup>5</sup>addestramento

che rilevanti dai dati, la pulizia del dataset, la gestione dei dati mancanti e anomali, la normalizzazione dei dati e la suddivisione dei dati in set di addestramento e test. Si procede alla scelta e all'addestramento del modello. Una volta terminato il processo, il modello viene valutato utilizzando il set di test, mediante metriche appropriate, come l'accuratezza, la precisione e l'F1-score.

## **Algoritmi di rilevamento delle anomalie**

Il rilevamento delle anomalie all'interno di un dataset svolge un ruolo cruciale nell'identificazione di eventi o modelli rari e potenzialmente dannosi. Le tecniche di rilevamento delle anomalie sono ampiamente utilizzate in diversi settori al fine di migliorare la sicurezza e l'affidabilità dei sistemi.

Nell'articolo [9] vengono esaminati diversi tipi di tecniche per il rilevamento delle anomalie, compresi metodi statici, algoritmi di apprendimento automatico come macchine a vettori di supporto e reti neurali, e approcci di insieme. Vengono discussi per ogni tecnica vantaggi e svantaggi. Inoltre, si espone il caso in cui si ha un set di dati sbilanciato, dove il numero di casi di frode è notevolmente inferiore rispetto alle transazioni legittime. Le CNN<sup>6</sup>, ad esempio, sono uno strumento molto accurato, però hanno dei notevoli svantaggi, quali: tuning dei parametri, design molto complessi, alta sensibilità nella scelta di iperparametri<sup>7</sup> e nella struttura dell'architettura.

## **Isolation Forest**

L'algoritmo Isolation Forest (IF) [10] è un algoritmo non supervisionato di anomaly detection. Questo modello propone l'impiego dell'isolamento come metodo più efficace ed efficiente per individuare le anomalie, rispetto alle misure

---

<sup>6</sup>Convolutional Neural Network: reti neurali composte da strati di neuroni artificiali organizzati in modo tale da riconoscere automaticamente caratteristiche gerarchiche complesse nelle immagini

<sup>7</sup>parametri esterni ai modelli di ML che vengono utilizzati per regolare il processo di addestramento dei modelli stessi

di distanza e densità. L'IF si focalizza sul fatto che, gli outliers<sup>8</sup> sono minori, in numero, rispetto agli altri dati del dataset. Il principio di funzionamento è il seguente:

1. L'IF crea un insieme di alberi decisionali binari detti iTree;
2. Una volta che gli alberi sono stati creati, viene calcolata l'anomalia per ciascun esempio nel dataset;
3. Gli esempi con un valore di anomalia superiore ad una soglia prefissata in precedenza vengono considerati outlier;
4. I valori anomali identificati vengono restituiti come output dell'algoritmo

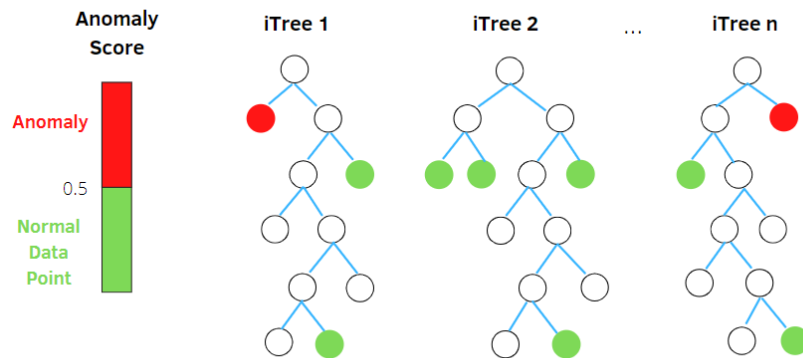


Figura 2.5: Anomaly detection with Isolation Forest

Come mostra l'articolo [10], il funzionamento di un IF può essere descritto come segue: dati in input un numero specificato di alberi e una dimensione di campionamento, che determina la quantità di dati forniti a ciascun albero, viene addestrata e restituita una foresta come mostrato in Figura 2.6.

Successivamente si valutano le istanze, definendo la Path Length<sup>9</sup>. Dati un'istanza e un iTree, si calcola quanti nodi attraversa ogni istanza prima di essere isolata o raggiungere la massima profondità.

<sup>8</sup>valori anomali

<sup>9</sup>lunghezza del percorso

---

**Algorithm 1** :  $iForest(X, t, \psi)$

---

**Inputs:**  $X$  - input data,  $t$  - number of trees,  $\psi$  - sub-sampling size

**Output:** a set of  $t$  *iTrees*

- 1: **Initialize**  $Forest$
- 2: set height limit  $l = ceiling(\log_2 \psi)$
- 3: **for**  $i = 1$  to  $t$  **do**
- 4:    $X' \leftarrow sample(X, \psi)$
- 5:    $Forest \leftarrow Forest \cup iTree(X', 0, l)$
- 6: **end for**
- 7: **return**  $Forest$

---

Figura 2.6: Pseudocode Isolation Forest

### 2.2.2 Architetture Cloud

Un Cloud è un server con dati e programmi a cui le organizzazioni possono accedere virtualmente. L'architettura Cloud è il modo in cui vari componenti tecnologici, sistemi e funzionalità interagiscono tra loro per creare una piattaforma online. I vantaggi di passare ad una tale architettura sono i seguenti:

- *Elasticità e scalabilità*: è possibile aumentare le risorse in modo dinamico e rapido per soddisfare i picchi di domanda e ridurle quando la richiesta diminuisce;
- *Riduzione dei costi*: poiché le risorse sono fornite come servizio su richiesta, non è necessario investire in infrastrutture costose;
- *Affidabilità e resilienza*: i fornitori di servizi Cloud implementano infrastrutture ridondanti e misure di backup per garantire che i dati siano protetti;
- *Accessibilità e flessibilità*: l'architettura Cloud consente l'accesso ai servizi e alle risorse attraverso una connessione Internet da qualsiasi luogo;
- *Sicurezza*: crittografia dei dati, autenticazione degli utenti, monitoraggio dei log e sicurezza a più livelli.



## Tipologie di architetture Cloud

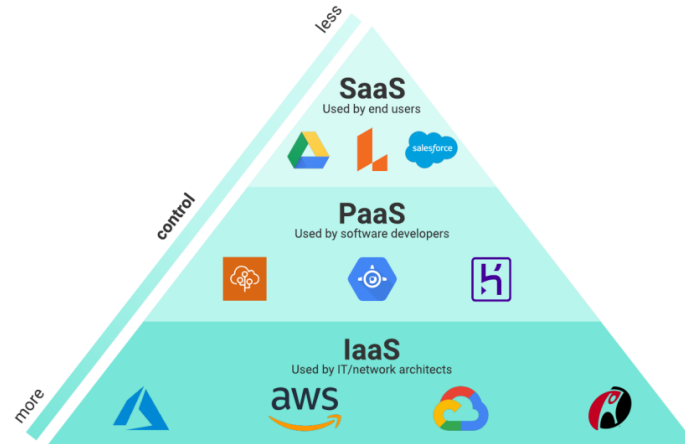


Figura 2.7: Architetture Cloud; fonte [6]

### Infrastructure as a Service (IaaS)

È un modello di servizio Cloud in cui l'infrastruttura di base, come server virtuali, storage e rete, viene fornita come servizio su richiesta tramite Internet. Gli utenti possono accedere e gestire queste risorse virtuali senza dover acquistare o prendersi cura dell'hardware.

### Platform as a Service (PaaS)

È un modello di servizio Cloud che fornisce un ambiente di sviluppo e di esecuzione completo per le applicazioni. Al contrario di IaaS, con PaaS vengono offerti anche i componenti aggiuntivi necessari per sviluppare, testare e gestire le applicazioni. Queste architetture semplificano il processo di sviluppo e deployment delle applicazioni, consentendo agli sviluppatori di concentrarsi sul codice e sulla logica dell'applicazione, senza doversi preoccupare dell'infrastruttura sottostante.

### Software as a Service (SaaS)

È un modello di servizi Cloud in cui un'applicazione software viene fornita agli utenti come servizio su Internet. Gli utenti possono accedere all'applicazione

attraverso browser web. I provider di Cloud Computing gestiscono quasi tutte le funzionalità Cloud per i propri utenti.

Articoli come questo [7], mostrano come il Cloud Computing (CC) sia molto usato per algoritmi di apprendimento automatico di ML. Scalabilità, latenza ridotta, throughput<sup>10</sup> più elevato sono alcuni degli elementi essenziali del Cloud Computing per IaaS. L'implementazione di algoritmi di ML su Cloud, secondo l'articolo sopra citato, offre varie opportunità per utilizzare questa tecnologia per una gestione più efficiente delle risorse.

### **2.2.3 Amazon Web Services**

Amazon Web Services (AWS) è una piattaforma di servizi di Cloud Computing fornita da Amazon. Offre un'ampia gamma di servizi: gestione dei database, analisi e archiviazione di dati, calcolo distribuito, virtualizzazione dei server e molti altri. AWS consente agli utenti di pagare solo per i servizi che vengono utilizzati. Con l'uso di tali servizi, si può aumentare o ridurre le risorse di calcolo o la capacità di archiviazione in base alle proprie esigenze. Sono presenti, inoltre, una serie di misure di sicurezza, tra cui controlli di accesso, crittografia dei dati in transito e a riposo, monitoraggio dei log e soluzioni di conformità.

#### **Amazon EC2**

Il servizio Amazon Elastic Compute Cloud (Amazon EC2) è un servizio di calcolo flessibile offerto da Amazon Web Services. Permette di eseguire server virtuali per far girare applicazioni e servizi web come in un tradizionale data center. Offre il vantaggio di poter effettuare il provisioning<sup>11</sup> di server e risorse immediatamente, in base alle proprie esigenze.

---

<sup>10</sup>capacità di un sistema o di un dispositivo di gestire un carico di lavoro specifico

<sup>11</sup>processo di creazione, configurazione e fornitura di server e risorse necessarie per supportare le applicazioni e i servizi

È possibile trovare in letteratura documenti come questo [8], in cui si utilizzano le infrastrutture IaaS, e quindi anche i server EC2, per valutare l'esecuzione dei processi di addestramento. Questo tipo di infrastruttura fornisce una comoda astrazione delle VM<sup>12</sup> eseguendo il multiplexing sulle macchine fisiche sottostanti. Quindi i server EC2 sono comunemente utilizzati per il Machine Learning poiché le istanze sono scalabili e possono facilmente essere configurate per soddisfare le esigenze specifiche richieste dai modelli. Tali server, inoltre, consentono di distribuire i carichi di lavoro in modo efficiente riducendo i tempi di elaborazione. È possibile selezionare istanze con una minore intensità di carbonio o che utilizzano energia proveniente da fonti rinnovabili, contribuendo a ridurre l'impatto ambientale del carico di lavoro di ML.

## 2.2.4 Linguaggio di Programmazione

Un linguaggio di programmazione è un insieme di regole e istruzioni utilizzato per programmi informatici. Consente ai programmatori di comunicare con la macchina e di specificare le operazioni che il computer deve eseguire. Questa comunicazione avviene tramite un compilatore, che traduce il codice sorgente in codice eseguibile dalla macchina. Esistono diversi linguaggi di programmazione: Java, C, C++, JavaScript, Python ecc.

Per gli algoritmi di Machine Learning e per il lavoro di load-shifting viene usato principalmente Python. Quest'ultimo è un linguaggio di programmazione ad alto livello, creato nel 1991 da Guido van Rossum e si è guadagnato una grande popolarità grazie alla sua semplicità di lettura del codice e alla vasta gamma di librerie disponibili. Python inoltre ha la peculiarità di essere interpretato, ossia non è necessario compilare il codice prima di eseguirlo, e di essere compatibile con diversi sistemi operativi come Windows, macOS e Linux. Infine supporta il paradigma object-oriented, consentendo agli svilup-

---

<sup>12</sup>Virtual Machine: software che emula un computer fisico consentendo di eseguire un sistema operativo e applicazioni su di esso

patori di organizzare il codice in classi e oggetti. Python offre una vasta gamma di librerie e framework, come TensorFlow, Keras, PyTorch e Scikit-Learn. Uno degli IDE<sup>13</sup> più usato per interfacciarsi con il linguaggio Python è *PyCharm*, sviluppato da JetBrains e disponibile in due versioni: la Community Edition (open-source) e la Professional Edition (a pagamento).

Nel campo dell'analisi dei dati è molto usato anche *Jupyter Notebook*, un'applicazione web open source che consente di creare e condividere documenti interattivi contenenti codice, testo e immagini. È suddiviso in celle che possono essere eseguite in modo interattivo, quindi è possibile scrivere codice ed eseguirlo immediatamente.

### 2.2.5 Moduli di serializzazione per Machine Learning

Una delle librerie che consente di serializzare oggetti in Python, ossia convertirli in un file binario e salvarli su disco, è *Pickle*. È ampiamente utilizzato per salvare modelli addestrati utilizzati nelle applicazioni di Machine Learning. Si noti che il modulo appena descritto, ha alcune limitazioni, ad esempio, i file pickle possono essere specifici della versione di Python utilizzata e potrebbero non essere compatibili tra diverse versioni del linguaggio.

Una libreria molto simile in termini di funzionalità è *Joblib*. Principalmente, la differenza tra le due è la diversa gestione della memoria; *Joblib* è progettato per gestire oggetti di grandi dimensioni in modo efficiente, riducendo l'impatto sulla memoria durante il processo di serializzazione e deserializzazione. *Joblib*, inoltre, è ampiamente utilizzato in combinazione con *scikit-learn*, una delle librerie di Machine Learning più popolari in Python. Se si lavora con modelli di *scikit-learn*, *Joblib* può offrire una migliore integrazione e compatibilità.

---

<sup>13</sup>ambiente di sviluppo integrato

### 2.2.6 API Carbon Intensity

Le API<sup>14</sup> Carbon Intensity forniscono dati sull'intensità delle emissioni di carbonio associate alla generazione di energia elettrica in diverse regioni. I dati sono espressi in termini di grammi di CO2 emessi per kilowattora (gCO2/kWh).

Electricity Map è un servizio online che fornisce informazioni sull'origine dell'elettricità e sull'intensità di carbonio in tempo reale. Il servizio, inoltre, offre un'API che consente agli sviluppatori di accedere a questi dati per creare applicazioni. L'applicazione è disponibile gratuitamente agli utenti, limitatamente con un massimo di 1000 chiamate. Si riporta il link al sito: [app.electricitymaps.com/map](http://app.electricitymaps.com/map)

### 2.2.7 Secure Shell

Secure Shell (SSH) è un protocollo di rete ampiamente utilizzato per consentire l'accesso remoto e la gestione sicura dei sistemi. Lo scopo principale di SSH è fornire un canale protetto attraverso il quale gli utenti possono autenticarsi e accedere in remoto ad un sistema, eseguire comandi e trasferire file in modo affidabile tra i computer coinvolti.

Di seguito è spiegato il funzionamento di SSH:

Inizialmente, avviene l'autenticazione tra Server<sup>15</sup> e Client<sup>16</sup> mediante l'uso di password o tramite una coppia di chiavi pubblica e privata. Successivamente, viene stabilita una connessione crittografata tra i due. Viene generata una chiave di sessione per la crittografia simmetrica, che viene usata per crittografare e decriptare i dati scambiati. Da default, le connessioni SSH utilizzano la porta 22, modificabile con il port forwarding.

---

<sup>14</sup>Application Programming Interface, regole e protocolli che consentono a diverse applicazioni di comunicare tra loro

<sup>15</sup>computer remoto che accetta le connessioni SSH dai Client

<sup>16</sup>computer locale da cui si desidera stabilire una connessione al Server

# Capitolo 3

## Studio di fattibilità

### 3.1 Architettura logica

In questa sezione si presenta uno schema logico che descrive lo spostamento di un algoritmo di Isolation Forest, usato per rilevamento delle anomalie, in questo caso delle transazione fraudolente, su macchine virtuali situate in zone del mondo in cui la carbon intensity è minore rispetto a quella su cui parte l'addestramento.

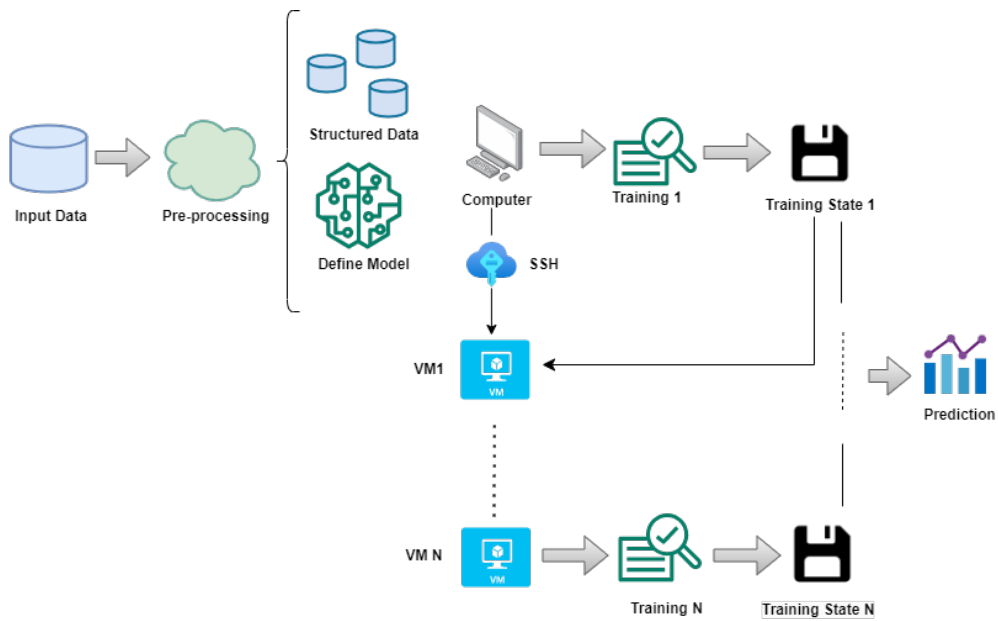


Figura 3.1: Schema logico del processo di training e load-shifting

Come mostrato in figura 3.1, inizialmente vi è una fase di pre-processing dei dati dove viene suddiviso il processo di training in più fasi. Successivamente si avvia il primo processo di addestramento sulla macchina locale. Una volta terminato, si salvano i dati necessari per riprendere il processo in seguito in un file. Dopo di che, si cerca una macchina virtuale EC2 fornita da Amazon Web Service, ubuntu, con una carbon intensity minore. Una volta trovata, la macchina che ha eseguito il training precedentemente (nella prima iterazione il computer locale), avvia una connessione SSH per copiare il file sulla nuova macchina in modo da poter riprendere l'esecuzione del training. Questo processo viene ripetuto un numero di volte uguale al numero in cui è stato suddiviso il processo di training. A termine dell'ultimo addestramento si esegue la previsione utilizzando il modello appena addestrato.

## 3.2 Scelte tecnologiche

### 3.2.1 Linguaggio di Programmazione

Il linguaggio di programmazione selezionato per sviluppare questo software è Python. È stato scelto per la semplicità con cui si possono creare e monitorare i processi di Machine Learning, per la vasta gamma di librerie disponibili e per l'integrazione con altre tecnologie.

### 3.2.2 Librerie

I moduli utilizzati sono:

- *Pandas*: libreria utilizzata nella fase di pre-processing per leggere il set di dati e analizzarlo;
- *Os*: modulo utilizzato per l'interazione con il sistema operativo;
- *Joblib*: libreria che fornisce strumenti per la serializzazione degli oggetti in Python e caching delle funzioni. In questo progetto si è scelto di

utilizzare la libreria *Joblib* invece di *Pickle* per salvare lo stato del training in un file, quando il processo viene interrotto, perché a differenza di *Pickle*, *Joblib* mantiene la precisione e l'accuratezza del modello anche quando viene eseguito su più macchine virtuali;

- *Paramiko*: libreria che fornisce un'implementazione del protocollo SSH per la creazione di connessioni sicure e l'esecuzione di comandi su server remoti;
- *Subprocess*: modulo usato per l'esecuzione di processi interni e interazione con essi. Utile quando si vuole eseguire comandi da terminale, avviare programmi esterni o comunicare con processi in esecuzione.

### 3.2.3 AWS

Per spostarsi su server con un'intensità di carbonio minore si utilizzano le istanze EC2 di Amazon Web Services poiché, quest'ultimo, è il Cloud Provider con più servizi disponibili, e perché grazie al programma AWS Academy sono stati concessi dei crediti per sviluppare il lavoro.

### 3.2.4 SSH

Per connettersi ad una Virtual Machine di tipo EC2 di AWS si utilizza una connessione SSH (Secure Shell).

## 3.3 Architettura Fisica

In questa sezione si approfondisce la gestione di una connessione SSH e il processo di copia di un file da una macchina all'altra.

In figura 3.2 è mostrata la connessione SSH tra un computer (Client) e una Virtual Machine. Il processo si può riassumere nelle seguenti fasi:



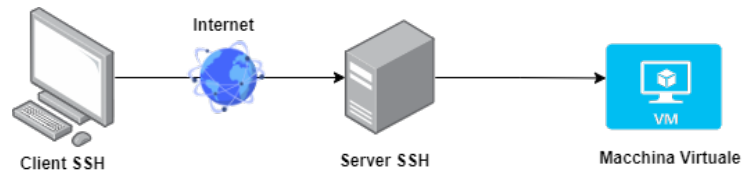


Figura 3.2: Connessione SSH tra due client

1. **Generazione delle chiavi:** Sul Client, viene generata una coppia di chiavi crittografiche: una privata e una pubblica;
2. **Invio della chiave pubblica al Server:** La chiave pubblica viene inviata al Server a cui si desidera connettere. Questo può essere fatto in diversi modi, ad esempio inserendo manualmente la chiave pubblica sul Server, oppure utilizzando comandi come *ssh-copy-id*;
3. **Autenticazione del Client:** Il Client invia la sua chiave pubblica al Server come prova d'identità. Il Server verifica se la chiave pubblica è presente nella lista di chiavi autorizzate. In caso affermativo il Client è autenticato;
4. **Scambio delle chiavi di sessione:** Viene stabilita una connessione sicura tra Client e Server;
5. **Crittografia e comunicazione sicura:** Tutti i dati trasmessi tra Client e Server vengono crittografati con la chiave di sessione, che garantisce la riservatezza e l'integrità dei dati durante la trasmissione;
6. **Gestione della sessione:** il Client può eseguire comandi o inviare richieste al Server. Il Server risponde alle richieste del Client e invia i risultati desiderati.

La figura 3.3 descrive l'architettura del processo di addestramento, salvataggio dello stato, copia mediante ssh del file e ripresa dell'addestramento. Nel momento in cui un ciclo di addestramento è completato viene creato un

file binario *training\_state.pkl*, che contiene il numero di estimatori addestrati ( $n\_estimators$ ), i dati di addestramento o features sotto forma di matrice o dataframe ( $X\_train$ ), il modello (*model*), i target corrispondenti ai dati di addestramento ( $y\_train$ ) e i target corrispondenti ai dati di test ( $y\_test$ ), raccolti in un *dictionary*<sup>1</sup>. Un file con estensione *.pkl* è un file dati serializzato utilizzando il modulo *Joblib*. Dopo aver instaurato una connessione SSH con la VM EC2, ubuntu, fornita da Amazon Web Services si copia il file *training\_state.pkl*, con il comando *scp -i*, su quest'ultima pronta a riprendere il training. Questo ciclo si ripete tante volte quante sono le suddivisioni in base al numero di estimatori totali.

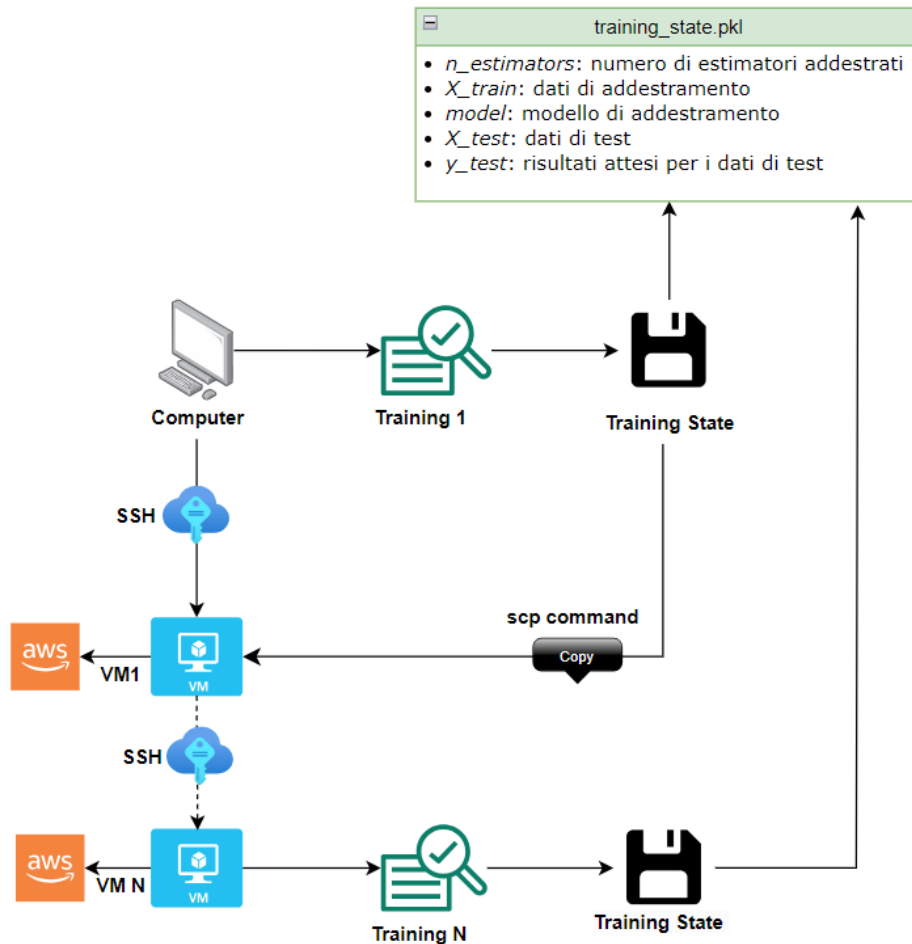


Figura 3.3: Architettura fisica del progetto

<sup>1</sup>struttura dati in Python che contiene coppie di elementi chiave-valore

# Capitolo 4

## Progettazione

In questo capitolo vengono descritte le funzionalità del software con l'uso dei diagrammi UML<sup>1</sup>, realizzati con il software web open source *draw.io*.

### 4.1 Diagramma dei casi d'uso

Il diagramma dei casi d'uso è uno dei diagrammi UML, utilizzato per modellare le funzionalità del sistema e le interazioni tra gli attori (utenti o sistemi esterni) e il sistema stesso.

In figura 4.1 è rappresentato il diagramma dei casi d'uso. Si possono identificare due attori: il computer locale, sul quale inizia l'addestramento, e la macchina virtuale EC2. Si identificano due casi d'uso principali:

1. **Training**: nel quale viene avviato il processo. In questo caso d'uso è incluso lo *Stop Training* dopo 10 estimatori completati e il *Save State* dove vengono salvati i dati per riprendere l'addestramento;
2. **Connessione alla VM in SSH**: che include i due casi d'uso *Copy State* e *Resume Training*.

---

<sup>1</sup>Unified Modeling Language

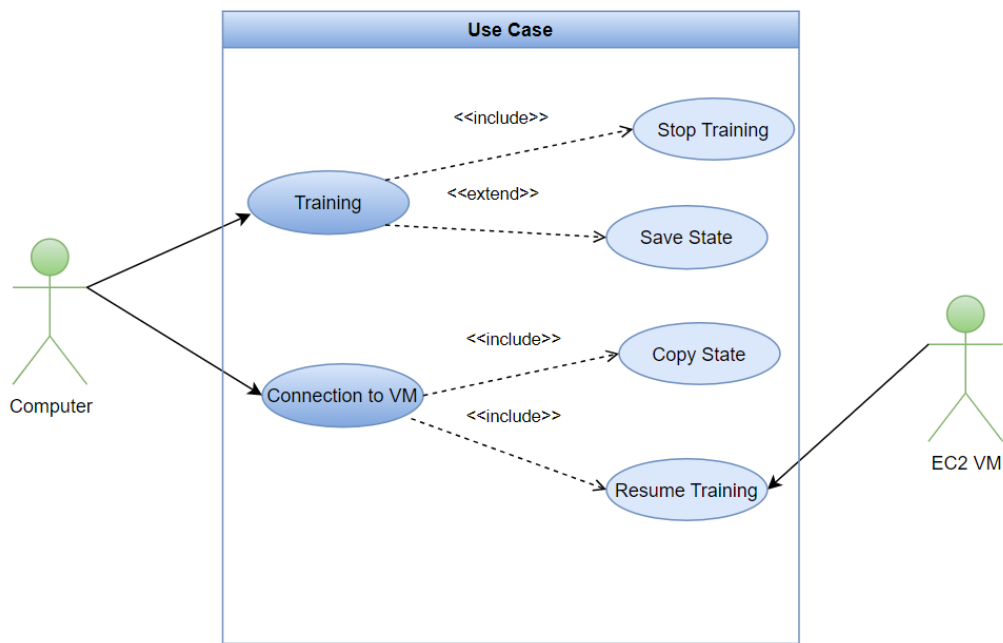


Figura 4.1: Diagramma dei casi d'uso

## 4.2 Diagramma delle classi

Il diagramma delle classi rappresenta le classi di un sistema, le loro relazioni, attributi e metodi. È uno strumento utilizzato per visualizzare la struttura statica del sistema.

In questo progetto, si distinguono 2 diverse classi: **Script\_Tesi\_Full\_New** e **Funzioni\_Script\_New**, che contengono una serie di attributi e funzioni, e un Package **Joblib\_Estimators\_Function** nel quale sono presenti altre 3 classi, utilizzate per il pre-processing dei dati e le esecuzioni delle fasi di training. Il diagramma è il seguente:

N.B. Dato che il progetto di load-shifting fa parte di un progetto più ampio, si noti che il Package *Joblib\_Estimators\_Function* e le funzioni in grassetto sono state implementate dal sottoscritto.

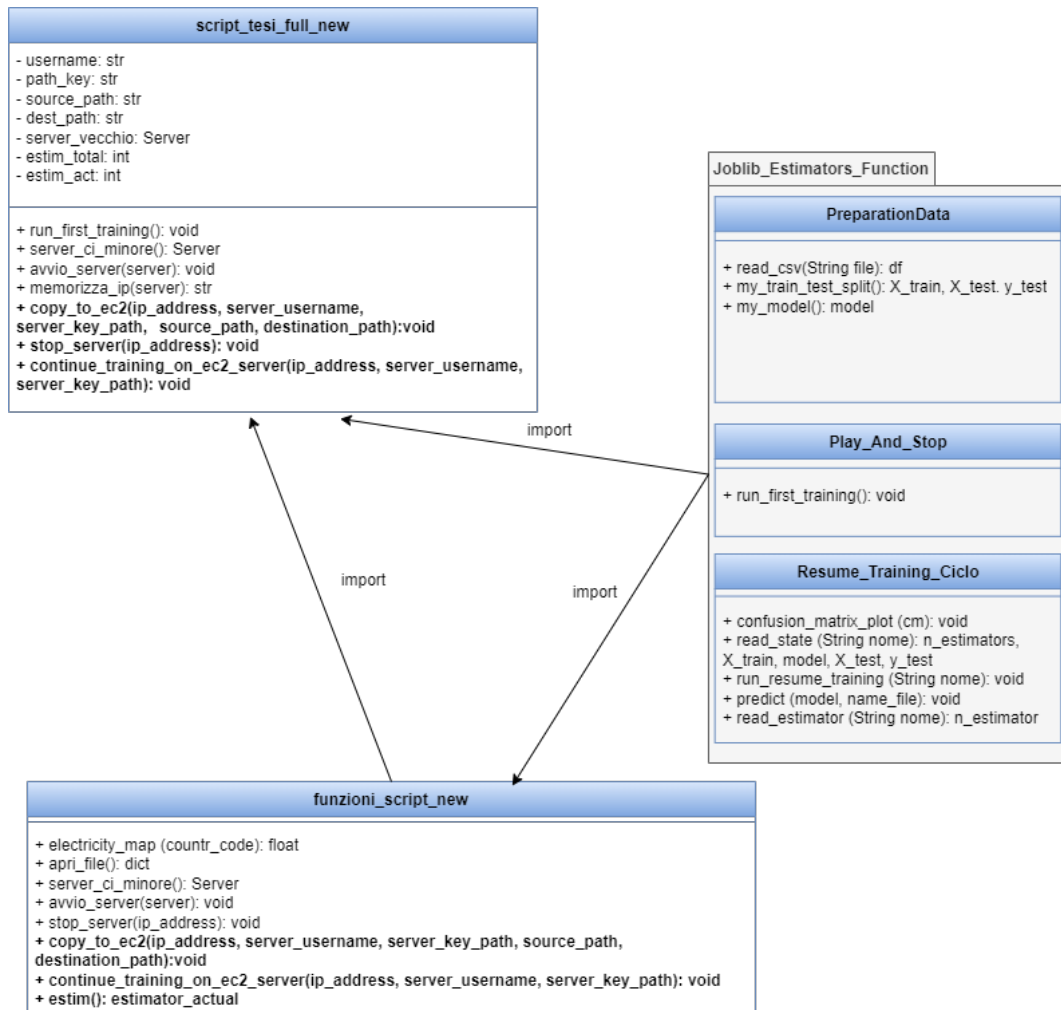


Figura 4.2: Diagramma delle Classi

## 4.3 Diagrammi di sequenza

Il diagramma di sequenza mostra l'interazione tra gli oggetti o le entità all'interno di un sistema in un ordine sequenziale. È utilizzato per modellare, comprendere e comunicare il comportamento dinamico di un sistema software o di un processo. Di seguito sono mostrati i diagrammi di sequenza per alcuni casi d'uso.

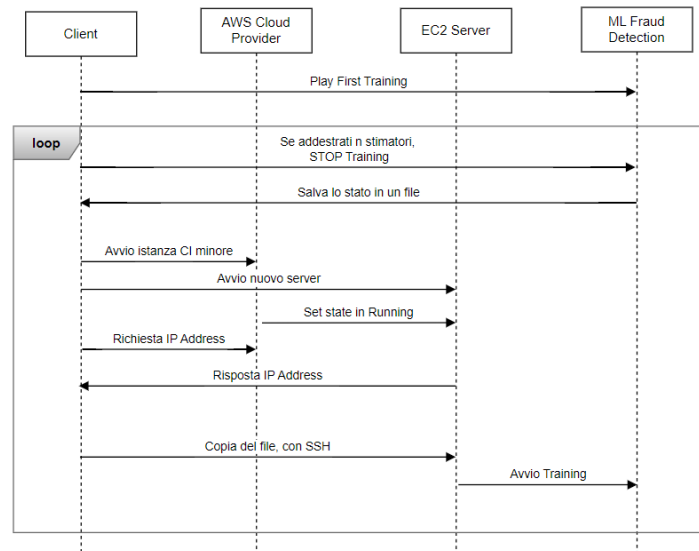


Figura 4.3: Diagramma di sequenza intero processo

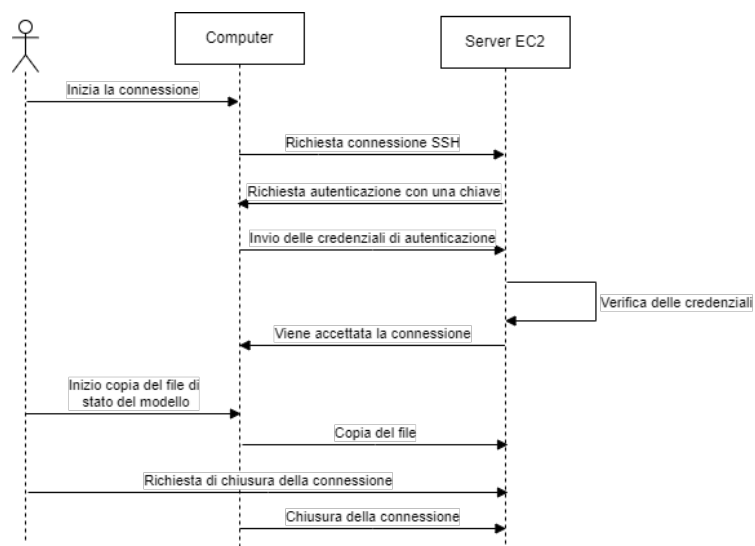


Figura 4.4: Diagramma di sequenza della connessione SSH e copia del file *training\_state.pkl*

# Capitolo 5

## Sviluppo

In questa sezione viene implementato e spiegato lo scheletro illustrato nel capitolo precedente.

### 5.1 Package Joblib Estimator Function

È stato creato un Package<sup>1</sup> che contiene i moduli necessari all'implementazione del codice. All'interno di questo pacchetto si trovano i moduli: *Preparation\_Data.py*, *Play\_And\_Stop.py* e *Resume\_Training\_Ciclo.py*.

#### 5.1.1 Preparazione dei Dati

Nella fase di pre-processing dei dati viene in primo luogo acquisito il dataset contenente un certo numero di transazioni normali e un certo numero di transazioni fraudolente, in formato CSV. Successivamente vengono standardizzate le colonne **Time** e **Amount** utilizzando la classe *StandardScaler()* del modulo *sklearn.preprocessing* e convertite in un array bidimensionale per scalare i valori in modo che abbiano media zero e varianza unitaria. Così facendo le colonne interessate sono simili alle altre del dataset e si può iniziare l'addestramento.

---

<sup>1</sup>directory Python

In questa fase sono presenti due funzioni:

- *my\_train\_test\_split*: in cui viene eseguita la suddivisione dei dati in un set di addestramento e uno di test, estratte le features, rimuovendo la colonna *Class*, ossia il target, e restituiti gli oggetti che poi verranno usati per addestrare e valutare il modello di ML;
- *my\_model*: in cui viene costruito il modello di Isolation Forest. Il modello viene creato con un numero di estimatori pari a 30.

### 5.1.2 Inizio dell'addestramento

Nel secondo modulo è presente solo la funzione **run\_first\_training** che esegue il primo ciclo di addestramento del modello. Nel dettaglio: viene richiamata la funzione *my\_train\_test\_split*, controllato se esiste un file contenente uno stato di addestramento, e in caso positivo caricato utilizzando il metodo *joblib.load*. Se, invece, il file non esiste viene creato un nuovo modello richiamando la funzione *my\_model* e viene inizializzato il numero di estimatori addestrati a 0. Dopo aver impostato il numero di estimatori da addestrare nel primo ciclo (10 in questo caso) e calcolato il numero di cicli necessari per addestrare tutti gli estimatori, viene avviato un ciclo di addestramento come mostra il seguente codice:

```
1     ...
2     for i in range(n_estimators+1, total_estimators+1):
3         model.fit(X_train)
4         print("Addestramento completato per l'estimatore", i)
5     ...
```

Successivamente vengono impostati gli estimatori nel modello al numero totale desiderato per stoppare l'addestramento e ne viene salvato lo stato:

```
1     ...
2     model.estimators_ = model.estimators_[:total_estimators]
3     model.n_estimators = total_estimators
```



```

print("Fine primo ciclo di Training")
5 state = {
    'model': model,
7     'n_estimators': total_estimators,
    'training_data': X_train,
9     'n_estimators_total': model.n_estimators,
    'X_test': X_test,
11    'y_test': y_test
}
13 with open("training_state.pkl", "wb") as f:
    joblib.dump(state, f)

```

### 5.1.3 Ripresa dell'addestramento

Nel modulo *Resume\_Training\_Ciclo* le funzioni che sono state implementate sono:

**read\_state:** consente di leggere lo stato di addestramento da un file passato in input e di ottenere le informazioni necessarie per continuare il training.

```

def read_state(name_file):
2   try:
        with open(name_file, "rb") as f:
4           loaded_state = joblib.load(f)
            print("Stato di addestramento caricato
correttamente.")
6   except FileNotFoundError:
            print("Il file del modello non e' stato trovato.")
8           return None
        except joblib.UnpicklingError as e:
10            print("Errore nel caricamento del modello:", str(e))
            return None
12    n_estimators = loaded_state['n_estimators']
    X_train = loaded_state['training_data']
14    model = loaded_state['model']

```

```

X_test = loaded_state['X_test']
y_test = loaded_state['y_test']

return n_estimators, X_train, model, X_test, y_test

```

**read\_estimator:** legge lo stato di addestramento da un file passato in input e restituisce solo il numero di estimatori.

**run\_resume\_training:** permette di riprendere l'addestramento del modello dal punto in cui è stato interrotto precedentemente. Essenzialmente è simile alla funzione *run\_first\_training* con la differenza che in input le viene passato il nome del file da cui prendere le informazioni grazie alla funzione *read\_state* sopra descritta.

**predict:** esegue la previsione utilizzando il modello addestrato e fornisce diverse metriche per valutare le prestazioni del modello sulla base dei risultati previsti e attesi. Le metriche utilizzate sono:

- *accuracy\_score*: è una metrica che calcola la percentuale di predizioni corrette rispetto al totale delle predizioni;
- *f1\_score*: è il punteggio F1, definito come la media armonica tra precisione e richiamo;
- *classification\_report*: è una funzione che restituisce un report dettagliato che include diverse misure di valutazione della classificazione, come precisione, richiamo, valore F1 e supporto per ciascuna classe target;
- *confusion\_matrix*: è una matrice quadrata che mostra il numero di predizioni corrette e errate suddivise per classe target. La riga rappresenta le etichette effettive, mentre la colonna rappresenta le etichette predette. Si usa per visualizzare le prestazioni del modello in termini di falsi positivi, falsi negativi, veri positivi e veri negativi.

## 5.2 Implementazione nel contesto principale

Come è stato detto nei capitoli precedenti, il programma realizzato in questo lavoro di tesi va ad implementare un progetto più ampio. Lo scopo è fermare un processo di addestramento di un Isolation Forest, spostarsi su un server virtuale creato con energia rinnovabile, o comunque con una carbon intensity minore, e riprendere l'addestramento. A tal fine si modificano i moduli, già implementati da terzi, aggiungendo alcune funzioni e si rinominano come segue:

- *funzioni\_script\_new.py*
- *Script\_Tesi\_Full\_New.py*

In *funzioni\_script\_new* sono state aggiunte le funzioni: **copy\_file\_to\_ec2()**, **continue\_training\_on\_ec2\_server()**, **predict()** ed **estim()**.

Di seguito sono spiegati alcuni passi di queste funzioni. La prima funzione copia un file da una macchina a un server EC2 remoto di AWS, utilizzando la connessione SSH. Richiede in input l'indirizzo IP del server, il nome utente, il percorso del file chiave SSH, il percorso del file da copiare e il percorso di destinazione sulla nuova macchina. Di seguito è mostrato il codice per aprire una connessione SSH e copiare il file:

```
1      ...
      # Creazione della connessione SSH
3      ssh = paramiko.SSHClient()
      ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy
      ())
5      # Imposta la politica di conferma automatica per l'
autenticita' dell'host
      ssh.load_host_keys(filename="C:\\Users\\mprof\\.ssh\\
known_hosts")
7      # Connessione al server remoto
      ssh.connect(ip_address, username=server_username,
key_filename=server_key_path)
9      print("Connessione SSH stabilita con successo")
```

```

# Percorso remoto del server EC2 in cui copiare il file
11 remote_path = f"{server_username}@{ip_address}:{
destination_path}"

# Comando SCP per copiare il file
13 scp_command = ["scp", "-i", server_key_path,
source_path, remote_path]

# Esegui il comando SCP
15 subprocess.check_call(scp_command)

print("Trasferimento avvenuto con successo")
17 ...

```

N.B. È stata omessa la gestione delle eccezioni che verrà esaminata successivamente.

La seconda funzione, `continue_training_on_ec2_server()`, esegue la connessione a un server EC2 remoto utilizzando il protocollo SSH e continua l'addestramento del modello utilizzando la funzione `run_resume_training()` del modulo `Resume_Training_Ciclo`. In input le si passa l'indirizzo IP del server, il nome utente e il percorso del file chiave SSH. Il codice seguente mostra come riprendere il comando dopo aver aperto la connessione:

```

1 model = Resume_Training_Ciclo.run_resume_training("
training_state.pkl")

print("Training stop sul server: ", ip_address)

```

La funzione `predict()`, effettua la predizione del modello sul server remoto previa connessione SSH, richiamando la funzione `predict` del modulo `Resume_Training_Ciclo`.

L'ultima funzione implementata in questo modulo, `estim()`, restituisce il numero di estimatori addestrati fino al momento della chiamata richiamando la funzione `read_estimator` del modulo `Resume_Training_Ciclo`.

Il secondo modulo `Script_Tesi_Full_New` è stato modificato con l'aggiunta di alcune righe di codice necessarie a far funzionare il programma. Per iniziare vengono definite alcune variabili di istanza:

- **username**: nome utente della macchina EC2 che viene avviata, in questo caso si usa `ubuntu`;
- **path\_key**: percorso dove è salvato il file `key.pem` contenente le chiavi SSH;
- **source\_path**: percorso del file da trasferire sulla macchina virtuale;
- **dest\_path**: percorso sul quale si vuole copiare il file;
- **estim\_total**: numero di estimatori totali del modello, in questo caso 30;
- **estim\_act**: numero dell'estimatore addestrato da aggiornare ogni volta, inizializzato a 0.

Dopo aver creato un'istanza del server di partenza, viene eseguita la prima parte del training sul server locale.

All'interno di un ciclo *while* viene richiamata la funzione per cercare un server tra quelli in stato di stop con carbon intensity minore. Si procede all'avvio del nuovo server, impostando successivamente lo stato in *running*.

Dopo di che si acquisisce l'IP del nuovo server, dopo aver atteso 30 secondi necessari alla VM per avviarsi.

Si esegue la copia del file `training_state.pkl` come mostra il seguente codice:

```

...
2  f.copy_file_to_ec2(indirizzo_ip, username, path_key,
    source_path, dest_path)
...

```

Dopo aver atteso 30 secondi si continua l'addestramento sul nuovo server EC2, si aggiorna la variabile che tiene traccia dell'estimatore addestrato e si esegue il controllo per verificare se questa variabile è uguale al numero degli estimatori totali. Se la risposta è positiva si procede alle previsioni, altrimenti si ripete il ciclo. Il seguente codice mostra la parte appena descritta:

```

1 while True:
    server_nuovo = f.server_ci_minore()
3     if (server_nuovo != None and server_vecchio.
        carbon_intensity >= server_nuovo.carbon_intensity):
        ...
5         # Continuazione training
        print("Ricomincio il training sul nuovo server tra 30
            secondi")
7         time.sleep(30)
        model = f.continue_training_on_ec2_server(indirizzo_ip,
            username, path_key)
9         estim_act = f.estim()
        if (estim_act == estim_total):
11             f.predict(indirizzo_ip, username, path_key, model)
            break
13     else:
        print("Nessun VM avviata")

```

## 5.3 Gestione delle eccezioni

La gestione delle eccezioni consente di gestire eventuali errori o situazioni impreviste durante l'esecuzione del programma. Nella scrittura del codice è stata posta particolare attenzione alla gestione delle eccezioni nella funzione **read\_state**, nella quale vengono gestite diverse eccezioni durante il caricamento dello stato di addestramento da un file usando la libreria *Joblib*. Ecco come può essere implementata:

```

    try:
2        with open(name_file, "rb") as f:
            loaded_state = joblib.load(f)
4            print("Stato di addestramento caricato
                correttamente.")
        except FileNotFoundError:

```

```

6         print("Il file del modello non e' stato trovato.")
        return None
8     except joblib.UnpicklingError as e:
        print("Errore nel caricamento del modello:", str(e))
10        return None
        ...

```

È stata implementata anche la gestione delle eccezioni per l'apertura e la chiusura della connessione con il server EC2 mediante SSH. Di seguito si riporta il codice:

```

1     try:
        # Creazione della connessione SSH
3         ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy
        ())
5         # Imposta la politica di conferma automatica per l'
        autenticita' dell'host
        ssh.load_host_keys(filename="C:\\Users\\mprof\\.ssh\\
        known_hosts")
7         # Connessione al server remoto
        ssh.connect(ip_address, username=server_username,
        key_filename=server_key_path)
9         print("Connessione SSH stabilita con successo")
        ...
11    except paramiko.AuthenticationException as auth_error:
        print("Errore di autenticazione:", str(auth_error))
13    except paramiko.SSHException as ssh_error:
        print("Errore SSH:", str(ssh_error))
15    except Exception as ex:
        print("Errore generico:", str(ex))
17    finally:
        if ssh:
19        ssh.close()

```

# Capitolo 6

## Test e analisi delle performance

La fase di test nella stesura del codice è estremamente importante perché permette di identificare e risolvere errori, malfunzionamenti e comportamenti indesiderati del software.

Prima di eseguire il codice bisogna fare delle operazioni preliminari:

- Inserire la chiave API Electricity Map nella funzione *electricity\_map()* del modulo *funzioni\_script\_new.py*;
- Copiare gli *AWS Details CLI* nella directory *.aws/credentials*;
- Scaricare, se non è già presente, il file *Key.pem* e tenere traccia della directory in cui lo si salva.

### 6.1 Test Funzionali

I test funzionali vengono eseguiti per verificare se l'applicazione soddisfa i requisiti funzionali specificati.

Tenendo spente tutte le istanze EC2 come mostrato in figura 6.1, si procede all'esecuzione del modulo *Script\_Tesi\_Full\_New.py*. All'inizio viene eseguito il primo ciclo di training che si ferma dopo 10 estimatori, viene avviato il server EC2 di nome *ubuntu\_1*, stampato l'IP e trasferito il file contenente lo stato del modello sulla VM, come mostrato in figura 6.2:



<input type="checkbox"/>	Name ▾	ID istanza	Stato dell'ista... ▾	Tipo di istanza ▾	Verifica dello stato	Stato dell'allarme	Zona di disponi... ▾
<input type="checkbox"/>	ubuntu_1	i-04315fda53f635b4	⊖ Arrestato @Q	t2.micro	–	Nessun allarme +	us-east-1b
<input type="checkbox"/>	ubuntu_2	i-01458ad1bc300e0b4	⊖ Arrestato @Q	t2.micro	–	Nessun allarme +	us-east-1a
<input type="checkbox"/>	ubuntu_3	i-02ae1867cfffab0245	⊖ Arrestato @Q	t2.micro	–	Nessun allarme +	us-east-1b

Figura 6.1: Stato delle istanze prima dell'avvio del Training

```

Addestramento completato per l'estimatore 10
Fine primo ciclo di Training
E' stato avviato il server ubuntu_1
Attendi 30 secondi per permettere l'avvio della VM EC2
52.7.96.199
Attendi 10 secondi per iniziare la copia del file
Connessione SSH stabilita con successo
Trasferimento avvenuto con successo

```

Figura 6.2: Test addestramento iniziale

Successivamente viene eseguito il secondo ciclo di addestramento sulla nuova istanza (figura: 6.3), previa connessione SSH, fino all'estimatore numero 20, come mostrato in figura 6.4:

<input type="checkbox"/>	Name ▾	ID istanza	Stato dell'istanza ▾	Tipo di istanza ▾	Verifica dello stato	Stato dell'allarme	Zona di disponi... ▾
<input type="checkbox"/>	ubuntu_1	i-04315fda53f635b4	⊕ In esecuzione @Q	t2.micro	⊕ Inizializzazione in corso	Nessun allarme +	us-east-1b
<input type="checkbox"/>	ubuntu_2	i-01458ad1bc300e0b4	⊖ Arrestato @Q	t2.micro	–	Nessun allarme +	us-east-1a
<input type="checkbox"/>	ubuntu_3	i-02ae1867cfffab0245	⊖ Arrestato @Q	t2.micro	–	Nessun allarme +	us-east-1b

Figura 6.3: Stato delle istanze dopo il primo ciclo di Training

N.B. Si noti che la dicitura in rosso, nella figura 6.4, è un **Warning** e indica che il codice è impostato per utilizzare l'elaborazione sequenziale con un singolo worker e non si sta sfruttando la capacità di elaborazione parallela.

```

Ricomincio il training sul nuovo server tra 30 secondi
Connessione SSH stabilita con successo
Stato di addestramento caricato correttamente.
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
Building estimator 1 of 10 for this parallel run (total 10)...
Building estimator 2 of 10 for this parallel run (total 10)...
Building estimator 3 of 10 for this parallel run (total 10)...
Building estimator 4 of 10 for this parallel run (total 10)...
Building estimator 5 of 10 for this parallel run (total 10)...
Building estimator 6 of 10 for this parallel run (total 10)...
Building estimator 7 of 10 for this parallel run (total 10)...
Building estimator 8 of 10 for this parallel run (total 10)...
Building estimator 9 of 10 for this parallel run (total 10)...
Building estimator 10 of 10 for this parallel run (total 10)...

```

Figura 6.4: Addestramento degli altri 10 estimatori

Il test continua avviando il terzo ciclo di addestramento su un'altra VM nello stesso modo del secondo.

Una volta finito l'ultimo ciclo di training vengono eseguite le predizioni e tirate le somme, come è mostrato nella seguente figura:

```

Addestramento completato per l'estimatore 30
Fine ciclo di Training
Training stop sul server: 52.6.155.223
Stato di addestramento caricato correttamente.
Connessione SSH stabilita con successo
Predection:
Stato di addestramento caricato correttamente.
Accuracy Score:
0.9888346617042941
Classification Report:

```

	precision	recall	f1-score	support
0	1.00	0.99	0.99	85302
1	0.08	0.53	0.14	141
accuracy			0.99	85443
macro avg	0.54	0.76	0.57	85443
weighted avg	1.00	0.99	0.99	85443

```

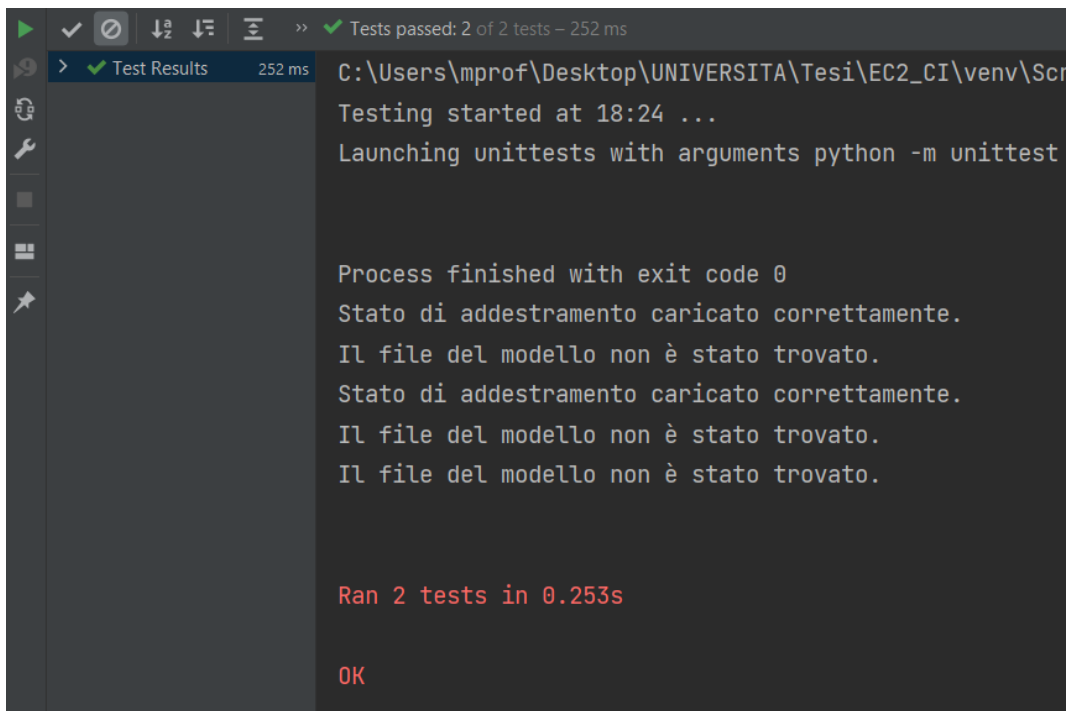
f_score:
0.1358695652173913

```

Figura 6.5: Conclusione del training e previsioni

## 6.2 Unit Test

I test di unità sono usati per verificare che parti specifiche di un programma funzionino correttamente. È stato utilizzato il modulo built-in *unittest* per verificare il funzionamento delle funzioni *read\_state* e *read\_estimator*, che riguardano la lettura del file di stato del modello. Il risultato è mostrato nella seguente figura:



```

> ✓ Test Results 252 ms C:\Users\mprof\Desktop\UNIVERSITA\Tesi\EC2_CI\venv\Scr
Testing started at 18:24 ...
Launching unittests with arguments python -m unittest

Process finished with exit code 0
Stato di addestramento caricato correttamente.
Il file del modello non è stato trovato.
Stato di addestramento caricato correttamente.
Il file del modello non è stato trovato.
Il file del modello non è stato trovato.

Ran 2 tests in 0.253s

OK
```

Figura 6.6: Unit Test funzioni *read\_state* e *read\_estimator*

# Capitolo 7

## Conclusioni e Sviluppi futuri

### 7.1 Conclusioni finali

In conclusione, si può evidenziare la buona riuscita del progetto. L'idea alla base era quella di acquisire un dataset di una banca e usare un algoritmo di rilevamento delle anomalie per identificare le transazioni fraudolente. Questo algoritmo però, potrebbe richiedere molto tempo in base al numero di transazioni presenti nel dataset, allora si è pensato ad un modo per spostare l'addestramento del modello su più server con una intensità di carbonio minore di quello precedente, salvando periodicamente lo stato di training per poter far ripartire l'addestramento sulla nuova istanza senza dover ricominciare dall'inizio. Prima di far partire il processo è stato acquisito il dataset e sono state standardizzate le due colonne che non lo erano. Successivamente, questo dataset, è stato diviso in dati di train e dati di test. Di seguito è stato avviato il processo di training, sulla prima macchina, utilizzando come algoritmo un Isolation Forest costituito da un numero specifico di estimatori. Dopo aver addestrato un numero, precedentemente deciso, di estimatori si è messo in pausa il processo e salvato lo stato, sotto forma di un dictionary Python, in un file con estensione *.pkl*. In seguito ad una ricerca di un server che presenti un'intensità di carbonio inferiore, è stata instaurata una connessione SSH tra la macchina

e il nuovo server. Successivamente è stato trasferito il file sulla nuova istanza ed è stato ripreso il processo di training dal momento dell'interruzione. Questo processo si ripete un numero di volte pari al numero in cui si è deciso di suddividere gli estimatori totali del modello di ML. Completato l'addestramento è stata eseguita la previsione del modello con delle metriche precise e con la stampa di una matrice di confusione. Per eseguire questo progetto sono stati scelti i server virtuali offerti dal servizio di Cloud Computing di *Amazon Web Service (AWS)* chiamati *Elastic Compute Cloud (EC2)*.

## 7.2 Sviluppi futuri

Attualmente l'applicazione del Green Software agli algoritmi di Machine Learning, che diventano sempre più pesanti, sta prendendo sempre più piede, quindi si è costantemente alla ricerca di metodologie che possano ridurre il consumo energetico dei software. Ulteriori sviluppi del progetto possono essere:

- Utilizzare diverse tecniche di suddivisione dell'algoritmo, ad esempio in base al tempo di esecuzione, oppure con una strategia di partizione dei dati di addestramento, implementando un algoritmo di addestramento gerarchico in cui si addestra il modello su sottoinsiemi più piccoli dei dati di train e si combinano i modelli addestrati.
- Utilizzare metodi di connessione per copiare il file di stato del modello e riprendere l'addestramento, diversi da SSH, ad esempio mediante l'uso dell'implementazione di API: insieme di regole e protocolli che consentono a diversi software di comunicare tra loro.
- Effettuare uno studio dettagliato sull'impatto ambientale di questo approccio di load-shifting basato su architetture Cloud, misurando l'impatto in termini di emissioni di carbonio e consumo di energia al fine di valutare l'efficacia della tecnica.

- Esplorare diverse architetture Cloud ad esempio Microsoft Azure, Google Cloud Platform e valutare quale di essi offre le opzioni più sostenibili dal punto di vista ambientale.
- Implementare una strategia di auto-scaling dinamico per le architetture Cloud in modo da adattare automaticamente le risorse in base al carico di lavoro e alle esigenze di addestramento dei modelli.

# Ringraziamenti

Vorrei riservare lo spazio finale di questo elaborato a tutti coloro che con instancabile supporto mi hanno sostenuto durante il mio percorso universitario.

In primis desidero ringraziare il mio relatore, **prof. Roberto Vergallo** per avermi permesso di realizzare un progetto interessante e allo stesso tempo all'avanguardia.

Esprimo la mia più sincera e completa gratitudine ai miei **familiari**, che mi hanno incoraggiato e stimolato a intraprendere questo percorso di studi fino al completo raggiungimento di esso.

Infine, ma non meno importanti, sono stati fondamentali per il mio percorso gli amici e compagni di studio, **Davide Calosso, Andrea Giannotta** e **Francesco Gabrieli** con i quali ho collaborato per superare insieme ogni difficoltà e condividere gioie e successi raggiunti con impegno e studio.

# Elenco delle figure

2.1	CO2 Grams Emitted, BERT Language Modeling . . . . .	10
2.2	Grafico aumento negli anni dei processi di deep learning . . . . .	11
2.3	Tipi di algoritmi di Machine Learning . . . . .	12
2.4	Flusso di lavoro del processo di ML . . . . .	13
2.5	Anomaly detection with Isolation Forest . . . . .	15
2.6	Pseudocode Isolation Forest . . . . .	16
2.7	Architetture Cloud; fonte [6] . . . . .	17
3.1	Schema logico del processo di training e load-shifting . . . . .	22
3.2	Connessione SSH tra due client . . . . .	25
3.3	Architettura fisica del progetto . . . . .	26
4.1	Diagramma dei casi d'uso . . . . .	28
4.2	Diagramma delle Classi . . . . .	29
4.3	Diagramma di sequenza intero processo . . . . .	30
4.4	Diagramma di sequenza della connessione SSH e copia del file <i>training_state.pkl</i> . . . . .	30
6.1	Stato delle istanze prima dell'avvio del Training . . . . .	41
6.2	Test addestramento iniziale . . . . .	41
6.3	Stato delle istanze dopo il primo ciclo di Training . . . . .	41
6.4	Addestramento degli altri 10 estimatori . . . . .	42
6.5	Conclusione del training e previsioni . . . . .	42
6.6	Unit Test funzioni read_state e read_estimator . . . . .	43



# Bibliografia

- [1] Titolo: "Green Software Engineering: A Systematic Mapping Study",  
Autori: Jorge Luis Pérez-Medina, Félix García, Mario Piattini, Pubbli-  
cato su: Journal of Systems and Software, Volume: 136, Numero: C,  
Pagine: 91-112, Anno: 2018
- [2] Titolo: "Green AI", Autori: Schwartz, Roy and Dodge, Jesse and Smith,  
Noah A. and Etzioni, Oren, Publisher: "Association for Computing  
Machinery", Pubblicato su Journal: "Commun. ACM", Mese e Anno  
di pubblicazione: "Novembre 2020", URL: [https://doi.org/10.1145/  
3381831](https://doi.org/10.1145/3381831)
- [3] Titolo sito Web: Follow The Sun, URL: [https://follow-the-sun.  
github.io/](https://follow-the-sun.github.io/)
- [4] Titolo: "Measuring the Carbon Intensity of AI in Cloud Instances" Au-  
tori: "Dodge, Jesse and Prewitt, Taylor and Tachet des Combes, Remi  
and Odmark, Erika and Schwartz, Roy and Strubell, Emma and Luccio-  
ni, Alexandra Sasha and Smith, Noah A. and DeCario, Nicole and Bu-  
chanan, Will", Publisher:"Association for Computing Machinery", URL:  
<https://doi.org/10.1145/3531146.3533234>
- [5] Titolo sito Web: Green AI, Autori: "Roy Schwartz, Jesse Dodge, Noah  
A. Smith, Oren Etzioni", URL: [https://cacm.acm.org/magazines/  
2020/12/248800-green-ai/fulltext](https://cacm.acm.org/magazines/2020/12/248800-green-ai/fulltext)

- [6] Titolo sito Web: O que é IaaS, PaaS e SaaS, URL: <https://www.linkedin.com/pulse/parte-61-o-que-%25C3%25A9-iaas-paas-e-saas-fl%25C3%25A1vio-silva/?trackingId=RvnzBdHCQRGbot7nH08Z5Q%3D%3D>
- [7] Titolo: "Machine learning (ML)-centric resource management in cloud computing: A review and future directions", Autori: "Tahseen Khan, Wenhong Tian, Guangyao Zhou, Shashikant Ilager, Mingming Gong, Rajkumar Buyya", Pubblicato su: "Journal of Network and Computer Applications", Volume: 204, Anno: 2022
- [8] Titolo: "Distributed Machine Learning with a Serverless Architecture", Autori: "Wang, Hao and Niu, Di and Li, Baochun", Book Title:"IEEE INFOCOM 2019 - IEEE Conference on Computer Communications",Anno: 2019, Pagine: 1288-1296
- [9] Titolo: "A comprehensive survey on graph anomaly detection with deep learning", Autori: "Ma, Xiaoxiao and Wu, Jia and Xue, Shan and Yang, Jian and Zhou, Chuan and Sheng, Quan Z and Xiong, Hui and Akoglu, Leman", Pubblicato su: "IEEE Transactions on Knowledge and Data Engineering", Anno 2021
- [10] Titolo: "Isolation Forest", Autori: "Liu, Fei Tony and Ting, Kai Ming and Zhou, Zhi-Hua", Pubblicato su: "2008 Eighth IEEE International Conference on Data Mining", Anno 2008, Pagine: 413-422
- [11] Titolo sito Web: What is SSH (Secure Shell)?, URL: <https://www.ssh.com/academy/ssh>
- [12] Titolo sito Web: Introduzione all'apprendimento per rinforzo, URL: <https://comeaprire.com/definizioni/introduzione-allapprendimento-per-rinforzo/>