UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Refactoring per Energy Consumption: Il Ruolo dei Bitwise Operator

RELATORE

Prof. Fabio Palomba

Dott. Stefano Lambiase

Dott.ssa Giulia Sellitto

Università degli Studi di Salerno

CANDIDATO

Leopoldo Todisco

Matricola: 0512110540

Anno Accademico 2022-2023

Questa tesi è stata realizzata nel

"Proverbi 16:3"

Abstract

Nell'epoca delle crescenti preoccupazioni ambientali, il campo della Green Software Engineering ha acquisito importanza come mezzo per mitigare l'impatto ecologico dei sistemi software.

Questa ricerca esplora il possibile ruolo degli operatori bitwise in tale contesto. In particolare, lo studio utilizza un esperimento empirico per indagare le possibili implicazioni degli operatori bitwise sul consumo di energia, concentrandosi sul loro impatto durante l'esecuzione di modelli di Machine Learning.

La domanda centrale che guida questa analisi è se l'utilizzo degli operatori bitwise possa non solo ridurre il tempo di esecuzione, ma anche contribuire a ridurre le richieste energetiche del processo. Dopo aver eseguito delle operazioni di refactoring alla libreria scikit-learn, l'esperimento prevede il confronto di modelli di Machine Learning con e senza l'inclusione di operatori bitwise, sia in termini di tempo di esecuzione che di consumo di energia.

Dall'analisi dei risultati ottenuti è emerso che mediamente, i modelli che hanno subito l'operazione di refactoring tendono a richiedere circa 7500 microJoule in meno, ma non sono state rilevate differenze nel tempo richiesto dai modelli.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto Applicativo

L'attenzione per l'ambiente e la sostenibilità stanno diventando sempre più importanti in molti settori. Con l'aumento dell'uso del software in molti aspetti della vita quotidiana è diventato essenziale considerare l'impatto ambientale di questo settore in continua crescita.

Il Machine Learning, in particolare il processo di addestramento dei modelli, richiede un enorme volume di dati e operazioni computazionali complesse. Queste operazioni richiedono server di alto livello e acceleratori hardware specializzati, che consumano quantità significative di energia elettrica. Addestrare un modello può richiedere giorni o settimane di tempo di calcolo su cluster di server, il che implica un consumo energetico paragonabile a quello di intere comunità. Come esempio concreto di ciò, basti pensare che la sola fase di training di ChatGPT-3 (il noto chatbot della compagnia OpenAI) ha consumato 700.000 litri di acqua dolce per il raffreddamento del data center, un quantitativo sufficiente a realizzare circa 370 automobili [1].

1.2 Motivazioni e obiettivi

La consapevolezza delle problematiche odierne sul consumo energetico può essere utile non solo per guidare gli sviluppatori nella scrittura di prodotti software a basso impatto ambientale, ma anche nell'ottimizzazione del codice legacy.

Come accennato nella sezione precedente, l'addestramento di modelli di Machine Learning è un'operazione estremamente costosa in termini di risorse e potenzialmente può durare per molti giorni.

Il fulcro centrale di questa tesi si concentra sull'esperimento empirico condotto con l'obiettivo di indagare se l'ottimizzazione attraverso l'uso di bitwise operators possa effettivamente ridurre il consumo energetico nei modelli di Machine Learning.

Per questo lavoro è stato preso in esame il codice sorgente della libreria scikit-learn [2], in particolare le classi LinearRegression e GaussianNaiveBayes. In tali classi le operazioni aritmetiche sono state riscritte utilizzando algoritmi già noti in letteratura che usano i bitwise operators.

I dettagli implementativi dell'operazione di refactoring sono presenti nella repository Github[1]

1.3 Risultati ottenuti

Al termine dell'esperimento empirico i risultati ottenuti sono stati analizzati con script appositi scritti in R.

Un primo risultato ottenuto da questa analisi è che l'uso dei bitwise operators non ha cambiato, mediamente, i tempi di esecuzione dei modelli di Machine Learning.

Spostando il focus sull'energia richiesta dai modelli è emerso che mediamente, i bitwise operators riescono a ridurre di circa 7500 microJoule il consumo energetico, con picchi fino a 37820 microJoule in meno.

In conclusione, è stato effettuato un T-Test per capire se reiettare o meno le ipotesi nulle, ma sia per le differenze nell'energia consumata che per le differenze nel tempo di esecuzione, non è stato possibile reiettare l'ipotesi nulla.

[1] https://github.com/leotodisco/scikit-learn

1.4 Struttura della tesi

Il documento di Tesi è strutturato come segue: nel Capitolo 2 viene presentato lo stato dell'arte riguardante il campo della Green Software Engineering, con un'analisi approfondita delle attuali metodologie, pratiche e tool di misurazione enegetica esistenti. Nel Capitolo 3 viene esposta la metodologia dell' esperimento empirico volto a valutare l'utilizzo dei bitwise operators nelle operazioni di training e predizione di modelli di Machine Learning. Successivamente, nel Capitolo 4, vengono presentati i risultati ottenuti dall'esperimento, con un'attenta analisi e interpretazione dei dati raccolti. Nel Capitolo 5, è stata presentata la fase di Decision Making. Successivamente, nel Capitolo 6, vengono esaminate le minacce alla validità dello studio, identificando potenziali fonti di errore o limitazioni metodologiche. Infine, nel Capitolo 7, vengono presentate le conclusioni generali derivate dalla ricerca condotta, insieme a possibili direzioni future di ricerca e sviluppo nell'ambito della Green Software Engineering.

CAPITOLO 2

Stato dell'arte

2.1 Green Software Engineering

McGuire et al. [3] hanno proposto un nuovo modello di sostenibilità nell'ingegneria del software che propone una serie di linee guida generali per integrare la sostenibilità nell'intero ciclo di vita del software, che potrebbero avere implicazioni tecnologiche specifiche. Tale modello prende il nome di Green Software Engineering.

2.1.1 "Il consumo energetico come requisito non funzionale"

Già nel 2013, Wilke et al. [4] hanno condotto uno studio empirico per capire quanto il consumo energetico delle applicazioni impattasse sull'esperienza utente. Lo studio consisteva nel collezionare una elevata mole di recensioni scritte su Google Play Store, mostrando che ben il 18% delle recensioni complessive riguardano la problematica dell'eccessivo consumo di batteria da parte delle applicazioni.
In conclusione, gli autori hanno provato a identificare le cause di consumi energetici elevati affermando che molte di esse sono legate a problematiche di programmazione come l'utilizzo eccessivo di risorse o l'adozione di antipattern specifici del linguaggio.

In generale, molte di queste problematiche potrebbero essere evitate con attività di test più precise.

Harman et al. [5] confermano il fatto che il consumo energetico si possa considerare un requisito non funzionale. Gli autori hanno dimostrato come i professionisti non tengano conto di un notevole numero di requisiti non funzionali, tra i quali il consumo energetico. Nello studio in questione si parla anche dell'importanza dell'ottimizzazione energetica in seguito all'incremento del *battery-powered computing*. Harman et al. sostengono che, a causa della sua rilevanza, l'ottimizzazione energetica dovrebbe assumere una specifica area del macroargomento "Search Based Software Engineering" [5], ma per fare ciò occorre che si trovi una metrica per il consumo energetico, da trasformare nella funzione di fitness. Nonostante ciò gli sviluppatori software non sono al corrente di queste problematiche; Pang et al. [6] hanno mostrato che pochi sviluppatori si preoccupano del consumo energetico del prodotto che producono.

Le lezioni più importanti apprese da questi studi è che problematiche di consumo energetico sono davvero importanti per gli utenti e per l'ambiente, rendendolo qualcosa che non si può più trascurare, e quindi, si può dire, senza dubbio, che il consumo energetico è un requisito non funzionale.

2.2 Come misurare il consumo energetico?

Precedentemente si è affermato che il consumo energetico è un requisito non funzionale, e per definizione, un requisito non funzionale deve essere misurabile. Harman et al. [5] citano due possibili modi di misurare i consumi energetici:

- **Hardware-Based** [7], dunque tramite l'ausilio di strumenti hardware appositi. Il modo in cui essi operano consiste nel determinare la corrente consumata da un device prima e durante l'esecuzione del software, al termine della misurazione si sottrae la misurazione di partenza da quella finale. Alcuni strumenti, detti *"Power Profiler"*, sono particolarmente avanzati e possono avere memoria interna, o potrebbero interagire con particolari registri delle CPU, detti Performance Counters. I performance counter sono registri speciali integrati nelle

CPU che forniscono informazioni sulle prestazioni del sistema, come ad esempio il numero di istruzioni eseguite, il numero di cache hit/miss, la frequenza di clock della CPU e statistiche simili.

Uno scenario potrebbe essere il seguente [7]:
il sistema sotto test esegue un task ed è connesso a un misuratore di potenza, che a sua volta è collegato a un *Energy Server* che interagisce con i performance counters e colleziona i dati.

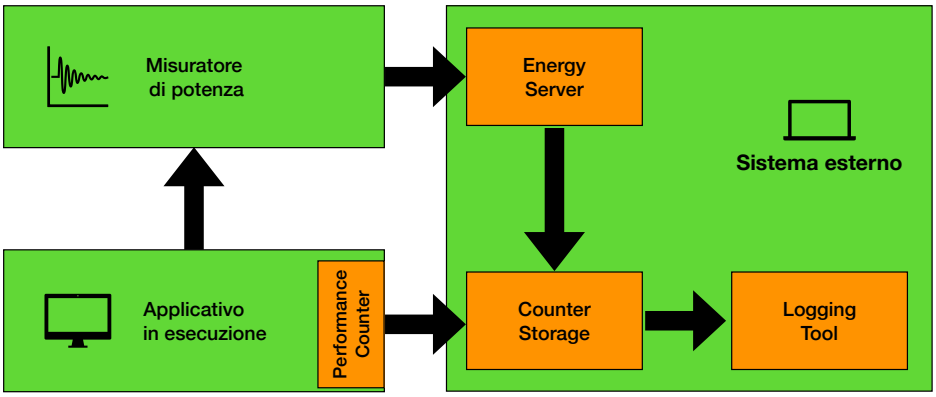

Figura 2.1: Anatomia di un tool di misurazione hardware

Alcuni dei tool più rilevanti appartenenti a questa categoria sono:

- POWERSCOPE [8], combina lo strumento hardware con il supporto del kernel per eseguire il campionamento. Si avvale di un software apposito che funge da analizzatore per mappare i dati campionati sulla struttura del programma, in modo da avere un profilo energetico ad-hoc per procedura. Quando la misurazione riguarda blocchi di codice (in questo caso procedure), parliamo di misurazione *mid-grained*, e questi livelli di precisione sono tipici di approcci hardware-based.
- GREENMINER [9], ha un'architettura di tipo client-server, il server si avvale di un Arduino, mentre il client è un Raspberry Pi che usa un dispositivo

Android per eseguire i test.

- **Model-Based**, una tecnica molto utile se non si è a disposizione di hardware specializzato, consiste nel creare un modello che stima il consumo energetico in tempo reale basandosi sui sensori relativi ai voltaggi integrati.
  T. Johann et al [7] si riferiscono a questa tipologia di misurazioni come misurazioni blackbox.
  POWERBOOTER [10] è un tool che fa affidamento ai voltaggi della batteria e al comportamento della batteria in fase di utilizzo del dispositivo Android e riesce a fornire in tempo reale delle stime.
  POWERAPI [11] Si tratta di una libreria di sistema che fornisce un'interfaccia di programmazione (API) per monitorare in tempo reale il consumo energetico del software con una granularità a livello di processi di sistema. La libreria offre anche valori di differenziazione dell'energia basati su risorse hardware, come ad esempio fornire l'energia consumata dal processo sulla CPU, sulla rete o su altre risorse hardware supportate. L'architettura di PowerAPI è modulare, poiché ciascuno dei suoi componenti è rappresentato come un modulo 2.2. I moduli comunicano tra di loro tramite il pattern publish/subscribe, e ogni modulo possiede sensori per ottenere dati dal sistema operativo, queste informazioni poi passano per il bus e arrivano ad ogni modulo. In una seconda fase i dati ottenuti sono utilizzati dai moduli "Formulas" per eseguire delle stime. Infine i moduli "Listeners" si occupano di "pubblicare" i dati elaborati.

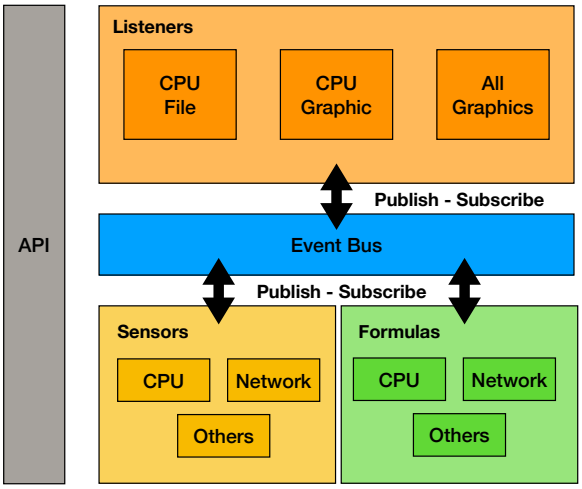

Figura 2.2: Architettura del tool PowerAPI

Alle due tecniche si può aggiungere anche un approccio **Software-Based** che consiste nello stimare il consumo energetico richiesto basandosi sulle caratteristiche del dispositivo, come ad esempio la frequenza della CPU. Un chiaro esempio di come operare con questo approccio si evince analizzando il tool GREENTRACKER [12], il quale raccoglie informazioni sulla CPU ed esegue alcuni compiti iniziali di "benchmarking".

Agli utenti viene richiesto di specificare quali sono le classi di software che desiderano testare e richiede di specificare in quali sistemi software ogni classe è attualmente installata sul proprio computer. GREEN TRACKER cerca di determinare il consumo medio della CPU per ciascuna delle classi specificate. Apre prima un programma specifico (ad esempio il browser) poi, in un intervallo specificato, salva un file di testo contenente i seguenti elementi: timestamp, ID del processo, utilizzo della CPU e nome del comando.

Tipologia	Pro	Contro
Hardware Based	Precisione elevata	Richiede componenti costosi
Model Based	Economica e usabile in ogni situazione	Richiede la perfetta calibrazione dei parametri per creare un modello che funzioni bene per uno specifico hardware
Software Based	Molto semplice da usare	Non si può essere molto precisi, e in alcuni ambiti, come l'embedded programming, non può essere usata

Tabella 2.1: Tabella riassuntiva dei vari metodi di misurazione

Una volta effettuata una misurazione, prima di esprimere un giudizio, occorre prestare attenzione ad alcuni fattori discriminanti, in particolare:

- **Effetto Hawthorne**: è l'insieme delle variazioni nei comportamenti di un fenomeno quando questo è sotto osservazione. Nel contesto dello studio in questione, bisogna tener conto che le proprietà che vogliamo misurare potrebbero essere influenzate dal processo di misurazione, in particolare, il consumo di una qualsiasi linea di codice potrebbe essere influenzato dalla strumentazione, per questo motivo vogliamo un'influenza minima, o per lo meno, costante, in modo da poterla ignorare.

- **Specificità**: si vuole che le misurazioni forniscano dettagli su quale componente e quale operazione consuma più energia, in questo modo l'ingegnere potrebbe trovare metodi alternativi.
  Si possono distinguere tre tipologie di misurazioni:

  - **a grana fine**: si considera il contributo individuale di ciascuna linea di codice per l'energia consumata;
  - **mid-grained**: si considera un blocco intero o un intero metodo/procedura;
  - **a grana grossa**: si considera l'energia consumata dall' esecuzione del programma in un dato periodo di tempo.

- **Instabilità delle componenti**: studi in letteratura hanno dimostrato che CPU nominalmente identiche presentano variazioni nel consumo energetico [13], specialmente nelle fasi di *idle* e di carico massimo.

2.3 Metriche per il consumo energetico

La presente sezione del capitolo ha lo scopo di presentare alcune metriche per quantificare il consumo energetico del software.

Secondo Johann et al. [7], dato che il software consiste di più moduli, ciascuno specializzato in task specifici, si può pensare di usare più metriche, in modo da misurare ogni modulo in modo più preciso ed adeguato.
Inoltre, gli autori lasciano una metrica generica:

$$Efficienza = Lavoro\ Eseguito/Energia\ Usata$$

Un esempio per quantificare l'efficienza energetica di un'applicazione che ordina dei numeri interi, seguendo la metrica suddetta, sarebbe:

$$Efficienza = Elementi\ Ordinati/Energia\ Usata$$

S. Kaxiras et al. [14], propongono l'**Energy Delay Product** (EDP). L'EDP si calcola come prodotto dell'energia e del tempo di esecuzione. Poiché l'energia è il prodotto di potenza e tempo, EDP è il prodotto di potenza e tempo al quadrato.

$$EDP = potenza \cdot (tempo)^2$$

Secondo questa metrica, dunque, riducendo il tempo di esecuzione di un task, l'efficienza energetica del prodotto software aumenterebbe notevolmente.

2.4 Come migliorare l'efficienza dei prodotti software?

Nelle sezioni precedenti si è parlato di perché e come misurare il consumo energetico di software. A questo punto, viene da chiedersi: "Una volta valutato il consumo energetico di un software, come è possibile migliorarlo?".

Nell'ingegneria del software esiste un termine per alcuni modelli e soluzioni che non sono ottimali e sono sintomi di cattive scelte di progettazione: *Bad Smell* (o Code Smell), coniato da Martin Fowler [15]. Secondo gli studi di Fowler [15], l'unico modo per risolvere questa problematica è il refactoring.

In letteratura ci si riferisce a cattive scelte di progettazione legata a problemi di consumo energetico come **Energy Smell**, e spesso sono legati al mondo delle mobile app come costatato dagli studi Palomba et al. [16].

2.4.1 I Bitwise Operators

Gli operatori bitwise operano a livello di bit su dati di tipo intero.
Esistono sei operatori bitwise:

1. $\gg$ Shift a destra, il valore di i $\gg$ j viene ottenuto facendo scorrere di j posizioni verso destra i bit di i; per questo motivo il risultato sarà equivalente alla divisione del valore originale per 2 alla j-esima potenza.

2. $\ll$ Shift a sinistra, il risultato di i $\ll$ j viene ottenuto facendo scorrere di j posizioni verso sinistra i bit di i, quindi il risultato sarà equivalente alla moltiplicazione del valore originale per 2 alla j-esima potenza;

3. & and bitwise, effettua l'operazione di and booleano su tutti i bit corrispondenti dei due operandi;

4. $\wedge$ or esclusivo;

5. $|$ or inclusivo;

6. $\sim$ complemento, produce il complemento del suo operando dove gli zeri sostituiscono gli uni e gli uni sostituiscono gli zeri;

2.4.2 Il ruolo dei Bitwise Operators nel Green Software Engineering

Nonostante il vasto interesse nei confronti delle prestazioni energetiche, un'area di ricerca trascurata riguarda il ruolo che i bitwise operators possono avere sul

consumo energetico. Un'investigazione dettagliata di questo aspetto potrebbe avere conseguenze significative sullo sviluppo di tecnologie energetiche innovative e ambientalmente sostenibili per il futuro.

Come è ben noto, la manipolazione dei bit e le altre operazioni a basso livello sono particolarmente utili per scrivere programmi per i quali sono importanti la velocità e/o l'uso efficiente della memoria [17]. In altre parole, l'uso di ottimizzazioni di basso livello di questo tipo potrebbe ridurre i tempi di esecuzione necessari al completamento di alcuni task e come emerge dallo studio di F. Palomba et al [16], esistono antipattern come il Data Transmission Without Compression, che causano un consumo energetico eccessivo proprio a causa della loro "pesantezza".

Allo stesso tempo, la metrica *EDP* afferma che l'efficienza energetica di un prodotto software è direttamente proporzionale al tempo di esecuzione [14], quindi possiamo affermare che i Bitwise Operators sono legati al concetto di efficienza energetica di applicazioni.

CAPITOLO 3

Metodologia

Nel presente capitolo ci si immerge nel cuore dell'indagine empirica, esaminando in dettaglio il processo di progettazione e preparazione dell'esperimento controllato. L'obiettivo dell'esperimento controllato è confrontare l'efficacia dell'ottimizzazione tramite operatori bitwise per ridurre il tempo di esecuzione dei progetti di Machine Learning, e di conseguenza rendere l'intero processo meno energy greedy a causa della relazione tra tempo di esecuzione e consumo energetico.

L'esperimento consiste nel test delle prestazioni degli algoritmi di apprendimento automatico con e senza operatori bitwise e il confronto dei risultati. Per effettuare l'esperimento si fa ricorso a un processo di sei fasi proposto da Fenton et al. [18], raffigurato nella figura 3.1.

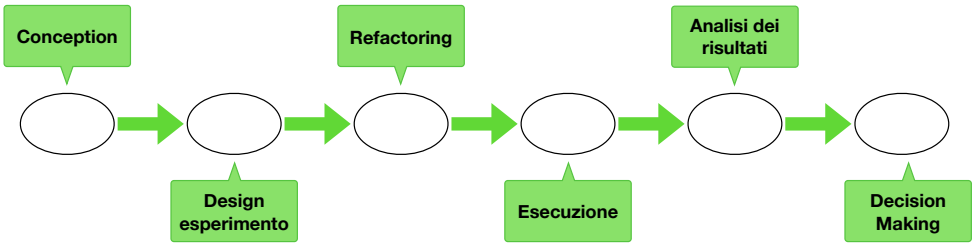

Figura 3.1: Overview del processo di sperimentazione empirica

3.1 Conception

In questa fase si definiscono gli obiettivi dell'esperimento il cui raggiungimento sarà valutato alla fine.

3.1.1 Definizione goal

Utilizzando il modello GQM (Goal Question Metric) [19], che rappresenta un approccio strutturato per stabilire obiettivi specifici, è possibile delineare e articolare il goal in questione in una maniera chiara e definita.

Object of Study: tecnica di ottimizzazione di codice sorgente tramite ausilio di bitwise operators;

Purpose: confrontare le differenze fra l'uso di bitwise operators e i tradizionali operatori aritmetici;

Quality Focus: tempo di esecuzione e la sua influenza sulla metrica EDP;

Perspective: ricercatore;

Context: l'esperimento viene eseguito su un computer con processore Intel Core i5 con architettura Haswell e sistema operativo Ubuntu 20.4. Tale scelta è dovuta al fatto che per ottenere le statistiche a runtime relative all'energia è necessario accedere

all'interfaccia RAPL (Running Average Power Limit) [20] del processore Intel e ciò è possibile solo utilizzando kernel Linux, dando i permessi e usando un processore con architettura x86 successiva all'architettura Sandy Bridge.
Gli oggetti dell'esperimento sono le classi GaussianNB e LinearRegression della libreria scikit-learn [2].
È quindi possibile articolare l'obiettivo dell'esperimento nel seguente modo:

◎ **Our Goal.** Obiettivo dell'esperimento controllato è confrontare l'utilizzo di operazioni aritmetiche tradizionali con l'implementazione di operazioni aritmetiche con l'ausilio di bitwise operators. In particolare, si mira a valutare se l'impiego dei bitwise operators può portare a una riduzione del consumo di risorse energetiche durante le fasi di addestramento e di predizione di tali modelli.

3.2 Definizione delle domande di ricerca

Le domande di ricerca in un esperimento empirico sono le interrogative chiave che un ricercatore cerca di rispondere attraverso il suo studio o la sua indagine. Queste domande guidano la progettazione dell'esperimento, la raccolta dei dati e l'analisi dei risultati.

🔍 **RQ$_1$.** *L'ottimizzazione di codice sorgente con bitwise operators può ridurre il tempo richiesto per l'esecuzione di modelli di Machine Learning?*

🔍 **RQ$_2$.** *L'ottimizzazione di codice sorgente con bitwise operators può ridurre la richiesta energetica per l'esecuzione di modelli di Machine Learning?*

Tali domande di ricerca sono giustificate dalla formula 2.3 secondo la quale l'efficienza energetica di prodotti software è data dal prodotto fra energia monitorata per tempo al quadrato, per questo motivo si intende indagare sugli effetti che si potrebbero avere effettuando operazioni di refactoring su modelli di Machine Learning.

3.3 Design dell'esperimento

In questa fase si formulano le ipotesi, si definiscono variabili e si progettano i test delle ipotesi.

3.3.1 Definizione delle ipotesi

In questa sezione vengono definite le ipotesi nulle e le ipotesi alternative.
Le ipotesi nulle assumono che non vi sia alcun effetto o differenza significativa nell'esperimento, e qualsiasi variazione osservata è attribuibile soltanto al caso o ad altri fattori non controllati. La verifica o la reiezione delle ipotesi nulle permette agli scienziati di determinare se i risultati ottenuti sono veramente significativi o se sono semplicemente il risultato di fluttuazioni casuali. Pertanto, la definizione accurata delle ipotesi nulle è un passo cruciale per assicurare la validità e l'integrità dell'intero processo sperimentale. Al contrario, le ipotesi alternative assumono che vi sia una significativa differenza fra i due gruppi osservati.

Ipotesi	Descrizione
H_{01}	Non c'è differenza nel tempo di esecuzione nell'uso dello stesso modello di Machine Learning prima e dopo il refactoring.
H_{A1}	Il tempo di esecuzione per il modello refactorizzato risulta essere significativamente inferiore.
H_{02}	Non c'è differenza nella quantità di energia richiesta dall'uso dello stesso modello di Machine Learning prima e dopo il refactoring.
H_{A2}	La quantità di energia richiesta per il modello refactorizzato risulta essere significativamente inferiore.

***Nota**: H_0 si riferisce alle ipotesi nulle, mentre H_A si riferisce alle ipotesi alternative*

Tabella 3.1: Ipotesi nulle ed alternative

3.3.2 Definizione delle variabili

In questa sezione vengono delineate le variabili dell'esperimento. Si hanno due tipi di variabili: indipendenti e dipendenti.

Le variabili indipendenti (o fattori) in un esperimento sono le variabili che vengono manipolate o controllate dal ricercatore per osservare gli effetti su altre variabili, tipicamente le variabili dipendenti. Sono la causa presunta che viene testata per vedere se produce un effetto specifico.

In questo studio è stata selezionata una singola variabile indipendente: "utilizzo dei bitwise operators". Questa variabile può assumere due valori (anche detti livelli):

- **NO_BIT** per indicare che il modello non usa operatori bitwise
- **SI_BIT** per indicare che si usano gli operatori bitwise

Le variabili dipendenti rappresentano l'elemento che in un esperimento o studio si intende prevedere o quantificare. Sono le variabili che si ritiene possano essere influenzate o determinate dalle variabili indipendenti. In questo studio sono state ritrovate le seguenti variabili dipendenti:

- **Tempo di esecuzione** della fase di training e della predizione, che influenza il valore della metrica EDP;
- **Consumo energetico** della fase di training e della predizione.

3.3.3 Tipo di Design

È stato scelto un esperimento appartenente alla classe *One Factor, Two Treatments*, più precisamente di tipo Paired Comparison. Questo vuol dire che tutti gli oggetti sono stati sottoposti a tutti i treatments. La scelta di questa tipologia di esperimento è dovuta al fatto che rappresenta un modo veloce di capire se la congettura è corretta, inoltre il numero di oggetti dell'esperimento è molto basso. Un approccio di tipo *Completely randomized* in questa situazione avrebbe reso più complesso notare le differenze fra i treatment.

3.4 Preparazione

Il primo step attuato per la preparazione dei test è stato quello di riscrivere alcune operazioni aritmetiche utilizzando i bitwise operators;
per l'elevamento a potenza si ha:

```
def bitwise_power(base, exponent):
    result = 1
    while exponent != 0:
        if exponent & 1:  # se ultimo bit di exponent vale 1
            result *= base
        base *= base  # eleva base al quadrato
        exponent >>= 1  # shift logico verso destra di exponent

    return result
```

Il funzionamento dell'algoritmo si basa sul fatto che ogni numero si può esprimere come somma di potenze di 2. Durante ogni iterazione del ciclo while, viene verificato se il bit meno significativo dell'esponente è 1 (mediante l'operatore AND bitwise con 1). Se è 1, moltiplichiamo il risultato parziale per il valore corrente della base. Successivamente, la base viene elevata al quadrato (usando l'operatore di moltiplicazione bitwise) e l'esponente viene diviso per 2 (tramite lo shift di un bit a destra). Questo processo viene ripetuto fino a che l'esponente non diventa 0.

Per la moltiplicazione è stato implementato il noto algoritmo della moltiplicazione russa che consiste nell'effettuare la moltiplicazione fra due numeri interi usando solo moltiplicazioni e divisioni per 2:

```
def bitwise_multiply(x, y):
    result = 0
    while y > 0:
        if y & 1:  # Se l'ultimo bit di y vale 1
            result += x
        x <<= 1  # Moltiplica per 2
        y >>= 1  # Dividi per 2

    return result
```

Il funzionamento è il seguente: il ciclo while si esegue fintanto che y è maggiore di 0, nel ciclo, tramite l'operatore and, si verifica se y è dispari, in tal caso si somma al risultato, x viene moltiplicato per 2 e y viene diviso per 2.

Per la somma si ha:

```
def bitwise_sum(num1, num2):
    while num2 != 0:
        carry = num1 & num2 # Calcola il riporto della somma tra
   bit
        num1 = num1 ^ num2 # Calcola l'addizione fra i due bit
        num2 = carry << 1 # Porta avanti il riporto

    return num1
```

L'algoritmo funziona sommando i bit dei due numeri uno per uno, tenendo traccia del riporto (carry) da un bit all'altro. Il riporto viene gestito spostando il carry in posizioni sempre più significative, in modo che possa essere considerato nella successiva iterazione.

Dopo aver completato l'implementazione delle funzioni, è stata presa la decisione di effettuare un processo di refactoring sulla libreria sklearn. Tale scelta è stata guidata dalla natura open source della libreria, dalla sua diffusa adozione all'interno della comunità e dalla sua semplicità. In particolare, ci si è concentrati sulle classi "LinearRegression" e "GaussianNaiveBayes". Si è optato per queste classi in virtù del loro coinvolgimento in numerose operazioni aritmetiche, il che ha favorito una valutazione più approfondita delle differenze tra il modello refactorizzato e quello standard.

Poichè gli operatori bitwise funzionano solo con numeri interi, è stato necessario eseguire cast a intero, questa operazione è stata fatta sia per il modello che usa i bitwise operators che in quello standard, in modo che il cast non influenzi sulle differenze nel tempo richiesto fra un modello e l'altro.

Per ottenere le statistiche a runtime, si è fatto ricorso a una modalità model-based, con specificità *mid-grained* poichè il focus è sul metodo "fit" che effettua il training del modello e sul metodo "predict" che si occupa di eseguire le predizioni. La libreria

usata a tale scopo è pyJoules [21], che utilizza POWERAPI (2.2) per ottenere statistiche di CPU, GPU e RAM. Di seguito si trova l'implementazione:

```
from pyJoules.energy_meter import EnergyContext
from pyJoules.device.rapl_device import RaplPackageDomain,
                                        RaplDramDomain
from pyJoules.handler.csv_handler import CSVHandler

csv_handler = CSVHandler('bayes_results_training.csv')
model = naive_bayes.GaussianNB()
with EnergyContext(handler=csv_handler,
                   domains=[RaplPackageDomain(0),
                            RaplDramDomain(0)]) as ctx:
    model.fit(x_train, y_train)
csv_handler.save_data()
```

Dunque l'EnergyContext misurerà l'energia richiesta da ogni RAPL Domain specificato, che è un aggregato di componenti hardware. In questo caso è stato monitorato il dominio Package, che consiste nella CPU e il dominio DRAM (fig 3.2). Si noti che pyJoules consente anche di monitorare il consumo della GPU, ma ciò è possibile solo se si dispone di GPU NVIDIA con architettura Volta.

Se si desidera replicare l'esperimento, la repository da clonare è la seguente: scikit-learn; per semplificare il processo è stato creato uno script di shell Bash chiamato *"permessi.sh"* che richiede al sistema operativo l'accesso all'interfaccia RAPL (a patto di avere un processore Intel compatibile e di star utilizzando un sistema Linux).

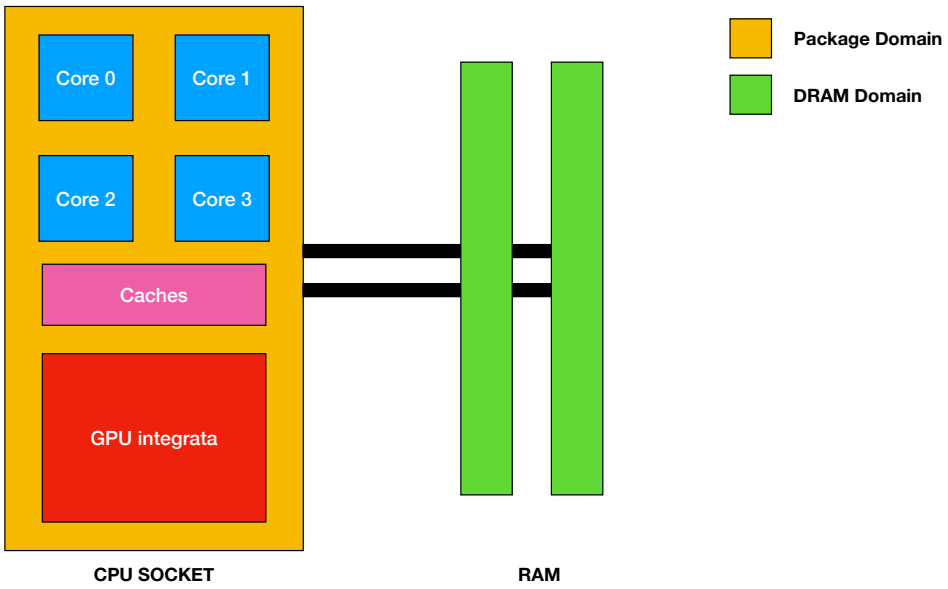

Figura 3.2: I Domains in pyJoules

Per eseguire i test sono stati presi in esame sette progetti distinti di Machine Learning:

- HeartCare, che esegue una classificazione Bayesiana per prevenire malattie cardiache;
- EnIA: tale progetto utilizza tecniche di classificazione Bayesiane per predire ottimali livelli di irrigazione di campi;
- DryBeans: qui si usano tecniche bayesiane per classificare differenti tipologie di fagioli;
- Student Grade Predictor: in questo progetto si usa una regressione lineare per predire i voti di studenti;
- Linear Regression
- Simple Linear Regression Project
- FitDiary: tale progetto fa uso di una regressione lineare per predire variazioni nelle percentuali di massa grassa e peso in periodi di allenamento fisico.

Infine, prima dell'esecuzione è stato eseguito un test preliminare, per assicurarsi che tutti i modelli funzionassero correttamente.

CAPITOLO 4

Risultati

In questo capitolo vengono presentati e analizzati i risultati ottenuti dalla nostra indagine. I dati raccolti durante il corso dell'esperimento rappresentano un punto fondamentale per rispondere alle domande di ricerca sollevate in precedenza.

4.1 Un'analisi preliminare

È stata condotta un'analisi preliminare tra le statistiche ottenute dai modelli standard e il modello con i cast a int. Questa analisi ha permesso di determinare il costo di tale operazione, consentendo così di considerarla come un elemento controllato che non influisce sullo studio. L'analisi è stata svolta prima prendendo in considerazione il tempo di esecuzione, poi la misurazione energetica.

Per il tempo di esecuzione si può affermare che non vi sono differenze nelle medie dei valori poichè la differenza dei valori medi è 0.

Per la misurazione energetica, invece, si ha una differenza di circa 1900 microJoule, questo vuol dire che il modello con i cast a int mediamente consuma 1900 microJoule in meno, ciò potrebbe essere dovuto al fatto che gli interi richiedono meno spazio in memoria.

4.2 Descrizione dei Dati

Alla conclusione dell'esperimento i dati sono stati collezionati sotto forma di sei CSV, il cui contenuto è descritto dalla tabella 4.1, cioè uno per ogni metodo di ciascun modello: regressione e classificazione standard, regressione e classificazione con i cast a int e infine regressione e classificazione dopo il refactoring.

Ogni test è stato ripetuto più volte per ridurre al minimo gli errori. Questo approccio è stato adottato in quanto il tool pyJoules, nel calcolare l'energia richiesta dal sistema, tiene conto non solo dell'energia associata all'oggetto di monitoraggio, ma anche dell'energia complessiva richiesta dall'intero sistema.

Nome	Descrizione
timestamp	Un'indicazione temporale del momento in cui è stato eseguito il test.
tag	un valore opzionale, utile se si vuole tenere traccia di più metodi.
duration	Il periodo di tempo necessario per completare il task, espresso in millisecondi.
package_0	Dati relativi all'energia richiesta dalla CPU sul socket 0, in microJoule
dram_0	Dati relativi all'energia richiesta dalla memoria DRAM, anch'essi in microJoule.

Tabella 4.1: Descrizione dei risultati

4.3 Analisi dei Risultati

Prima di procedere con l'analisi statistica mediante il t-test, si è effettuata un'analisi preliminare dei dati, permettendo di esaminare ad alto livello le differenze tra i modelli.

Nel contesto del metodo "predict", figura 4.1, l'uso degli operatori bitwise emerge come una soluzione efficace, evidenziando una riduzione sia nei tempi di esecuzione che nei consumi energetici, specialmente nel caso della classificazione con modello Gaussian Naive Bayes.

Tuttavia, la situazione cambia leggermente nel contesto della fase di "training", figura 4.2. Innanzitutto, si può notare un incremento generale sia nei tempi di esecuzione che nei consumi energetici, e più precisamente, l'impatto degli operatori bit a bit sembra essere quasi trascurabile nella maggior parte dei casi, e addirittura, nel caso della classificazione bayesiana, sembra richiedere anche più energia per la RAM.

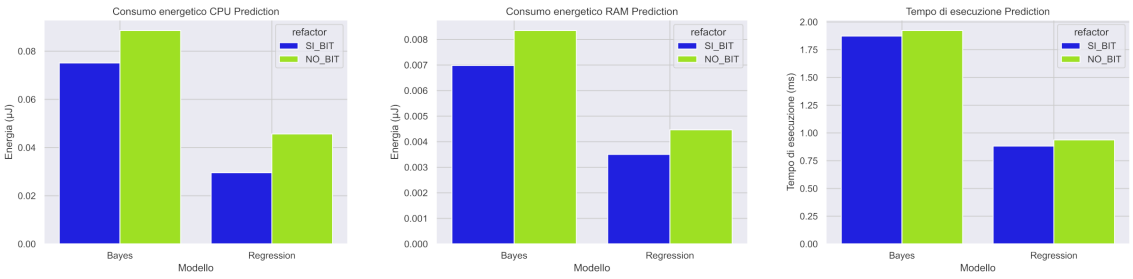

Figura 4.1: Confronto del consumo energetico e del tempo di esecuzione del metodo predict prima e dopo il refactoring

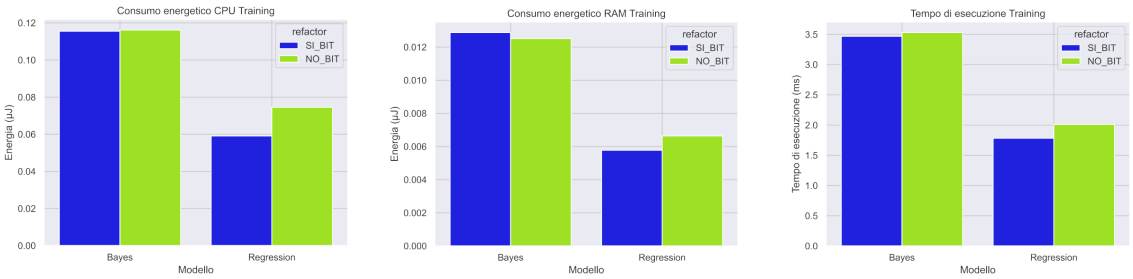

Figura 4.2: Confronto del consumo energetico e del tempo di esecuzione del metodo fit prima e dopo il refactoring

Si può fare un'analisi più dettagliata prendendo in considerazione i valori di media, mediana e deviazione standard, figura 4.2. Come si può notare prendendo in esame il tempo, si può dire che non ci sono differenze fra prima e dopo l'applicazione del treatment. Spostando il focus sull'energia consumata, invece, si nota che mediamente i modelli dopo il treatment richiedono 7520 µJ in meno, un valore che può sembrare basso, ma ci sono dei picchi di differenza fino a 37820 µJ nel caso migliore e di 46812 µJ nel caso peggiore.

Oggetto di Studio	Refactoring	Media	Mediana	Deviazione standard
Tempo	NO	0.002 ms	0.0016 ms	0.001 ms
Tempo	SI	0.002 ms	0.002 ms	0.001 ms
Energia	NO	85165.30 µJ	62540.8 µJ	49230.79 µJ
Energia	SI	77644.91 µJ	63839.5 µJ	50930.74 µJ

Tabella 4.2: Tabella riassuntiva delle statistiche prese in esame per l'analisi dei risultati

Prima di procedere con i due t-test relativi alle due ipotesi nulle, è stato necessario effettuare una ristrutturazione dei dati al fine di creare un nuovo dataset idoneo all'esecuzione dei t-test. Tale dataset presenta una colonna in cui si ritrova l'energia richiesta prima del refactoring, una colonna per l'energia necessaria dopo il refactoring, una colonna per la durata prima del refactoring e una colonna per la durata dopo il refactoring. Si noti che la colonna relativa all'energia è la somma di quelle che erano l'energia per la DRAM e per la CPU nel vecchio dataset.

4.4 T-Test

Il T-Test è un test statistico di tipo parametrico con lo scopo di verificare se il valore medio di una distribuzione si discosta significativamente da un altro.

Prima di eseguire i due T-Test sono stati eseguiti due test di correlazione col metodo di Pearson. Il p-value ottenuto dal primo test di correlazione fra la colonna

relativa al tempo di esecuzione prima del refactoring e quella dopo il refactoring è di 1.4112×10^{-7}, il che vuol dire che le colonne sono estremamente correlate tra di esse. Il p-value ottenuto dal secondo test di correlazione fra la colonna relativa all'energia totale richiesta dal modello prima del refactoring e quella dopo il refactoring è di 0. Valori così bassi per il p-value indicano che è molto probabile che le differenze notate tra i gruppi non siano dovute al caso.

Il primo T-Test è relativo alle differenze nell'energia richiesta. Per questo test è stato scelto un livello di confidenza del 95% e i risultati sono i seguenti:

- p-value: 0.2694;
- statistic: 1.1536;

Il p-value indica la probabilità che la differenza fra le media dei gruppi sia casuale, quindi, in un test con una confidenza del 95% si vuole che il p-value assuma un valore minore o uguale a 0.05; invece la statistica rappresenta la differenza media tra i due gruppi divisa per la deviazione standard delle differenze campionarie. In questo caso, essendo il p-value molto elevato, si può affermare che non ci sono evidenze statistiche per reiettare l'ipotesi nulla nonostante dall'analisi dei dati emerga che l'utilizzo degli operatori bitwise mediamente riduca la richiesta energetica di 7520.39 microJoule.

Il secondo T-Test è quello relativo alle differenze nel tempo di esecuzione, anche per esso è stato scelto un livello di confidenza del 95%, i risultati sono stati:

- p.value: 0.8097;
- statistic: 0.2458;

Anche in questo caso il p-value è molto elevato, pertanto non si può reiettare l'ipotesi nulla.

CAPITOLO 5

Discussion

5.1 RQ1: Relazione fra bitwise operators e tempo di esecuzione

🔍 **RQ$_1$.** *L'ottimizzazione di codice sorgente con bitwise operators può ridurre il tempo richiesto per l'esecuzione di modelli di Machine Learning?*

Dall'analisi dei risultati emerge che la differenza fra le medie dei due gruppi è uguale a 0, con una mediana di 0.0001.

Eseguendo il T-Test si è notato che il p-value è di 0.8 quindi un valore eccessivamente alto che non consente di reiettare l'ipotesi nulla.

👉 **Answer to RQ$_1$.** L'utilizzo della tecnica di ottimizzazione del codice sorgente mediante i bitwise operators non ha un impatto significativo nella riduzione del tempo di esecuzione per l'esecuzione di modelli di Machine Learning.

5.2 RQ2: Relazione fra bitwise operators ed energia

🔍 **RQ$_2$.** *L'ottimizzazione di codice sorgente con bitwise operators può ridurre la richiesta energetica per l'esecuzione di modelli di Machine Learning?*

In questo caso dall'analisi emerge che l'uso dei bitwise operators, mediamente, riduce l'energia richiesta di 7520 microJoule, nonostante ci siano dei picchi di differenza di 37820 microJoule nel caso migliore e di 46182 microJoule nel caso peggiore; con una mediana di 10950.5 microJoule.

Il t-test eseguito mostra un valore p dello 0.2 il che vuol dire che non può essere reiettata l'ipotesi nulla.

👉 **Answer to RQ_2.** L'utilizzo della tecnica di ottimizzazione del codice sorgente mediante i bitwise operators non ha un impatto significativo nella richiesta energetica per l'esecuzione di modelli di Machine Learning.

CAPITOLO 6

Minacce alla validità

In questo capitolo vengono esaminate le minacce alla validità dell'esperimento controllato. Le minacce alla validità di un esperimento si riferiscono a fattori o problemi che possono compromettere la capacità di un esperimento di fornire risultati significativi e validi. Si dividono in:

- **Conclusion**: influiscono sulla capacità di trarre conclusioni sulle relazioni tra le variabili indipendenti e le variabili dipendenti;
- **Internal**: influenzano le variabili indipendenti rispetto alla casualità;
- **External**: condizioni che limitano la capacità di generalizzare i risultati;

6.1 Conclusion

Nel corso dello studio ci si è trovati di fronte a diverse sfide che hanno potuto influenzare l'affidabilità dei profili energetici e dei dati raccolti.

Una delle principali sfide riguarda il rumore termico interno nei sistemi utilizzati (2.2). Questo può creare variazioni nei consumi energetici registrati, anche quando il codice eseguito rimane invariato. Per affrontare questa minaccia alla validità,

ogni test è stato eseguito più volte, cercando di ottenere una visione più stabile e rappresentativa dei consumi energetici reali.

Inoltre, un altro fattore che potrebbe influenzare i risultati è la temperatura esterna. Le variazioni di temperatura possono incidere sui consumi energetici dei dispositivi elettronici, rendendo essenziale tenere conto di questa variabile nelle analisi.

Va anche sottolineato che il framework di misurazione pyJoules tiene conto non solo della richiesta energetica dell'oggetto in esame ma anche dell'energia richiesta da tutto il sistema. Questo può avere potenzialmente distorto i risultati, poiché misura un consumo più ampio rispetto all'applicazione in esame. Tuttavia, per mitigare questo effetto, è stato minimizzato il numero di processi in backgroud.

In sintesi, le sfide connesse al rumore termico, alle variazioni di temperatura esterna e al framework di misurazione sono state affrontate attraverso l'esecuzione di misurazioni ripetute.

6.2 Internal

Storia Questo tipo di minacce si verifica quando eventi esterni o fattori che si verificano durante l'esperimento influenzano i risultati in modo significativo. Nel caso di questo esperimento, il rischio di incombere in queste problematiche è piuttosto elevato in quanto se i test fossero stati eseguiti mentre il computer stesse eseguendo altri task i risultati sarebbero fortemente influenzati da ciò. Il tutto è stato evitato ripetendo il test più volte e minimizzando il numero di processi in background.

Maturità Queste minacce si verificano quando i cambiamenti dei partecipanti nel corso del tempo influenzano i risultati dello studio in modo imprevisto o non controllato. Nel corso del tempo, i modelli di Machine Learning potrebbero migliorare le loro prestazioni naturalmente grazie all'ottimizzazione o all'accumulo di dati, in particolare, Jupyter Notebook [22] adotta un sistema di caching che fa sì che la prima esecuzione del modello sia sempre quella più lenta, le successive esecuzioni tendono a richiedere meno tempo a causa dei dati in cache. L'effetto del caching nel corso dello studio è stato annullato poichè dopo ogni test il server è sempre stato riavviato.

6.3 External

I risultati dello studio potrebbero essere influenzati da minacce alla validità esterna che limitano la generalizzabilità dei risultati stessi. In primo luogo, va notato che l'hardware utilizzato è stato un processore Intel Core i5 4460 con quattro core. Questa scelta potrebbe aver avuto un impatto significativo sui tempi di elaborazione e sulle prestazioni generali, rendendo difficile la trasposizione dei risultati su sistemi con configurazioni hardware diverse.

Inoltre, un'altra minaccia alla generalizzazione dei risultati è data dalla dimensione ridotta del campione utilizzato nello studio. La sample size limitata potrebbe influenzare la rappresentatività dei dati raccolti e la robustezza delle conclusioni tratte.

CAPITOLO 7

Conclusioni

Nel presente capitolo, verranno esaminate le conclusioni emerse dall'analisi dei dati ottenuti nell'ambito dell'esperimento empirico condotto, ma vengono anche esposte limitazioni e possibili sviluppi futuri.

7.1 Limitazioni e Sviluppi Futuri

Durante lo studio sono emerse alcune circostanze limitanti che necessitano di un'adeguata considerazione:

- **Sample size limitata**: la dimensione del campione di partecipanti utilizzato in questa ricerca è da considerarsi ridotta. Tale circostanza può influenzare la possibilità di generalizzare i risultati ottenuti, rendendo pertanto opportuno esaminare le conclusioni con la dovuta prudenza.
- **Errore nei Metodi di Misurazione**: i metodi di misurazione adottati manifestano tassi di errore. Questi possono aver introdotto variazioni nei dati raccolti, potenzialmente influenzando l'accuratezza delle analisi eseguite e delle conclusioni formulate.

- **Riproducibilità limitata**: la riproducibilità dello studio è soggetta a limitazioni, in quanto richiede l'accesso a specifiche risorse hardware e l'impiego di un sistema operativo Linux. In particolare, l'uso di pyJoules implica l'utilizzo di una CPU Intel o di una GPU NVIDIA. Ciò comporta restrizioni significative sulla generalizzabilità dello studio e sulla sua applicabilità in contesti vari

Inoltre, questo lavoro di tesi potrebbe motivare lavori futuri: potrebbero essere prese in considerazione ulteriori classi della libreria sklearn da analizzare e alle quali applicare il treatment qui proposto, oppure si potrebbe riconsiderare lo studio su altre architetture hardware, oppure si potrebbe considerare di applicare il refactoring anche alla libreria numpy, sulla quale si basa sklearn.

Cambiando totalmente la tipologia di studio, si potrebbe studiare un metodo di assegnazione di classe energetica di appartenenza al software, utilizzando metriche e antipattern già noti in letteratura.

7.2 Conclusioni

Questo lavoro di tesi analizza la relazione fra l'ottimizzazione con Bitwise Operators e l'efficienza energetica di Modelli di Machine Learning.

Per comprendere la relazione è stato effettuato un esperimento empirico che consiste nel confronto delle metriche emerse dall'esecuzione degli stessi modelli prima e dopo il refactoring con i bitwise operators. Il focus è stato su:

- Tempo richiesto dal modello;
- Energia richiesta dal modello;

In seguito sono stati analizzati i risultati ed è emerso che per il tempo di esecuzione non sono state ritrovate differenze fra i modelli prima e dopo il refactoring; per il consumo energetico invece, nonostante il T-Test abbia avuto esito negativo, è emerso che in media il tempo di esecuzione e il consumo energetico diminuiscono dopo l'applicazione del refactoring. Questo risultato apre la strada a possibili indagini future in cui potrebbero essere approfonditi gli aspetti legati a questa tendenza.

Bibliografia

[1] P. Li, J. Yang, M. A. Islam, and S. Ren, “Making ai less" thirsty": Uncovering and addressing the secret water footprint of ai models,” *arXiv preprint arXiv:2304.03271*, 2023. (Citato a pagina 1)

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. (Citato alle pagine 2 e 15)

[3] S. McGuire, E. Shultz, B. Ayoola, and P. Ralph, “Sustainability is stratified: Toward a better theory of sustainable software engineering,” *arXiv preprint arXiv:2301.11129*, 2023. (Citato a pagina 4)

[4] C. Wilke, S. Richly, S. Götz, C. Piechnick, and U. Aßmann, “Energy consumption and efficiency in mobile applications: A user feedback study,” in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, 2013, pp. 134–141. (Citato a pagina 4)

[5] M. Harman, Y. Jia, and Y. Zhang, “Achievements, open problems and challenges for search based software testing,” in *2015 IEEE 8th International Conference*

on Software Testing, Verification and Validation (ICST), 2015, pp. 1–12. (Citato a pagina 5)

[6] C. Pang, A. Hindle, B. Adams, and A. E. Hassan, "What do programmers know about software energy consumption?" *IEEE Software*, vol. 33, no. 3, pp. 83–89, 2015. (Citato a pagina 5)

[7] T. Johann, M. Dick, S. Naumann, and E. Kern, "How to measure energy-efficiency of software: Metrics and measurement results," in *2012 First International Workshop on Green and Sustainable Software (GREENS)*, 2012, pp. 51–54. (Citato alle pagine 5, 6, 7 e 10)

[8] J. Flinn and M. Satyanarayanan, "Powerscope: a tool for profiling the energy usage of mobile applications," in *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999, pp. 2–10. (Citato a pagina 6)

[9] A. Hindle, A. Wilson, K. Rasmussen, E. J. Barlow, J. C. Campbell, and S. Romansky, "Greenminer: A hardware based mining software repositories software energy consumption framework," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 12–21. [Online]. Available: https://doi.org/10.1145/2597073.2597097 (Citato a pagina 6)

[10] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES/ISSS '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 105–114. [Online]. Available: https://doi.org/10.1145/1878961.1878982 (Citato a pagina 7)

[11] Powerapi. [Online]. Available: https://powerapi.org (Citato a pagina 7)

[12] N. Amsel and B. Tomlinson, "Green tracker: A tool for estimating the energy consumption of software," in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '10. New York, NY, USA:

Association for Computing Machinery, 2010, p. 3337–3342. [Online]. Available: https://doi.org/10.1145/1753846.1753981 (Citato a pagina 8)

[13] J. von Kistowski, H. Block, J. Beckett, C. Spradling, K.-D. Lange, and S. Kounev, "Variations in cpu power consumption," in *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, ser. ICPE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 147–158. [Online]. Available: https://doi.org/10.1145/2851553.2851567 (Citato a pagina 10)

[14] S. Kaxiras and M. Martonosi, "Computer architecture techniques for power-efficiency," *Synthesis Lectures on Computer Architecture*, vol. 3, no. 1, pp. 1–207, 2008. (Citato alle pagine 10 e 12)

[15] M. Fowler, "Refactoring: Improving the design of existing code," in *11th European Conference. Jyväskylä, Finland*, 1997. (Citato a pagina 11)

[16] F. Palomba, D. Di Nucci, A. Panichella, A. Zaidman, and A. De Lucia, "On the impact of code smells on the energy consumption of mobile applications," *Information and Software Technology*, vol. 105, pp. 43–55, 2019. (Citato alle pagine 11 e 12)

[17] K. N. King, *C programming: a modern approach*. WW Norton & Co., Inc., 1996. (Citato a pagina 12)

[18] N. Fenton and J. Bieman, *Software metrics: a rigorous and practical approach*. CRC press, 2014. (Citato a pagina 13)

[19] V. R. B. G. Caldiera and H. D. Rombach, "The goal question metric approach," *Encyclopedia of software engineering*, pp. 528–532, 1994. (Citato a pagina 14)

[20] (Citato a pagina 15)

[21] (Citato a pagina 20)

[22] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks – a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players,*

Agents and Agendas, F. Loizides and B. Schmidt, Eds. IOS Press, 2016, pp. 87 – 90. (Citato a pagina 31)

Ringraziamenti

Desidero esprimere la mia sincera gratitudine a tutte le persone che hanno reso possibile la realizzazione di questa tesi. In primis sento di dover ringraziare il relatore Prof. Palomba perchè ha avuto fiducia in me quando gli parlai dell'idea di partenza per questo lavoro e i miei correlatori, il dottor Lambiase e la dottoressa Sellitto per i consigli, il materiale didattico, la pazienza e il supporto mostratomi nei mesi passati.

Un doveroso ringraziamento va alla mia famiglia, se oggi sono qui è sopratutto merito loro che mi hanno sostenuto emotivamente ed economicamente, sopratutto tre anni fa quando ho deciso di cambiare totalmente vita, in particolare un grazie va a mia madre e mia zia per aver sempre creduto in me, anche quando neanche io ci credevo.

Un caloroso ringraziamento va ai miei colleghi e amici ("goblin") universitari: Carlo, Alessandro, Mario, Paolo, Luigi, Vincenzo, e a coloro che si sono aggiunti più recentemente come Irene, Benedetto e Gianmario (dottor Voria). Grazie per i momenti condivisi, i ricordi costruiti, le sessioni di studio insieme, gli appunti scritti e grazie per avermi strappato una risata anche nei momenti più difficili del nostro percorso (i.e. la preghiera a Maradona per il progetto di IS). Mi auguro che il prossimo capitolo delle nostre vite sia pieno di successi.

Un ulteriore ringraziamento va ai ragazzi dell'Associazione Coscienze per aver sempre supportato noi studenti, con il Drive degli appunti, il Github e i seminari organizzati. In particolare un grazie va al Presidente Rebecca e a Leonardo.

Un ultimo ringraziamento va ai miei amici di vita Carmine e Nunzio, anche voi avete contribuito al raggiungimento di questo traguardo, grazie per il supporto e per avermi sempre capito ogni volta che ho dovuto sacrificare il tempo in vostra compagnia per studiare. Non posso esimermi dall'augurarvi un futuro luminoso e gioioso!

Desidero concludere la tesi con un impegno personale: questo piccolo traguardo rappresenta per me solo l'inizio di un nuovo percorso. L'entusiasmo e la passione che ho (ri)trovato in questi anni mi spingono a non fermarmi; al contrario, mi motivano a continuare a lavorare duramente.

Sono consapevole che il cammino sarà impegnativo, ma sono determinato a superare le sfide che incontrerò lungo la strada. Con il sostegno prezioso di coloro che mi hanno guidato finora e con la mia dedizione costante, sono fiducioso che nel prossimo futuro si apriranno avanti a me nuove porte.

Thank God