UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Informatica

Corso di Laurea Triennale in Informatica

TESI DI LAUREA

Implementazione di un Sistema di Intrusion Detection per l'IoT tramite Machine Learning

RELATORE

Prof. **Fabio Palomba**

Università degli Studi di Salerno

CANDIDATO

Matteo Cicalese

Matricola: 0512106871

Anno Accademico 2021-2022

Nessun vincitore crede al caso

Sommario

L'Internet of Things è un paradigma comunicativo oramai affermato, che mira a connettere diversi tipi di oggetti e dispositivi alla rete in uno o più sistemi, con lo scopo di automatizzare operazioni e collezionare ed elaborare informazioni, senza richiedere alcuna interazione da parte dell'uomo. Con la crescente diffusione dei sistemi IoT, sono aumentate e migliorate le piattaforme e le tecnologie su cui si basano, i possibili casi d'utilizzo, insieme con i livelli di accessibilità ed usabilità, arrivando oggigiorno a costituire un riferimento fisso nella quotidianità di chiunque.

Sono innumerevoli i vantaggi per i quali sempre più utenti scelgono di adottare dispositivi e sistemi di questo tipo, ma sono purtroppo altrettanti gli svantaggi e le criticità presenti di cui sono ignari. A causa della superficialità in diversi aspetti dello sviluppo software e della progettazione hardware, del costante miglioramento dei malware e delle tecniche di violazione, uniti ad una scarsa consapevolezza degli utenti in merito a tali questioni, i problemi e le vulnerabilità di sicurezza nell'IoT sono enormi e molto più delicati rispetto ad un normale dispositivo, a causa dell'alto grado di interoperabilità e dell'elevata mole di dati interscambiati tra i dispositivi di un sistema di questa tipologia.

A tal proposito, in questo documento sarà discussa l'anatomia dei sistemi IoT e le relative questioni di privacy e sicurezza, e sarà valutato l'impiego di Intelligenza Artificiale in ottica di una risoluzione definitiva di tali problematiche. Per sopperire in maniera diretta alle criticità discusse, sarà progettata e implementata la pipeline di un modello di machine learning in grado di fare Intrusion Detection, utilizzando l'approccio ingegneristico CRISP-DM.

Il sistema intelligente ottenuto é completo di tecniche di data cleaning, normalization e preprocessing, e potendo sfruttare diversi algoritmi di classificazione, ha la capacità di segnalare la presenza di un eventuale botnet della famiglia Mirai o BASHLITE insediata in diversi dispositivi IoT. Il software implementato supera il 99% in tutte le metriche di riferimento, e fa riflettere sulla facilità di impiego di una soluzione del genere per rendere sicuro il proprio ecosistema IoT, oltre ad offrire delle considerazioni sulle modalità di utilizzo di un'API per il monitoraggio del traffico per l'ottenimento di dati che consentano di aumentare le tipologie di malware identificabili.

Indice

Elenco delle figure

Elenco delle tabelle

CAPITOLO 1

Introduzione

1.1 Contesto applicativo

Per *Internet of Things* (*IoT*), o Internet delle Cose, si intende quel percorso nello sviluppo tecnologico in base al quale, attraverso la rete Internet, ogni oggetto dell'esperienza quotidiana acquista una sua identità nel mondo digitale. Esso si basa sull'idea di oggetti "intelligenti" tra loro interconnessi su una rete, che tramite apposite tecnologie, sono in grado di scambiare le informazioni possedute, raccolte e/o elaborate, ed eseguire azioni di conseguenza [45].
L'Internet of Things è un paradigma che non conosce, potenzialmente, confini applicativi: dall'autovettura che dialoga con l'infrastruttura stradale per prevenire incidenti, agli elettrodomestici di casa che si coordinano per ottimizzare il consumo di energia, agli impianti di produzione che scambiano dati con apposite apparecchiature per la gestione del loro ciclo di vita, fino ad arrivare ai sistemi di sicurezza che avvisano automaticamente le autorità in caso di furti e intrusioni. Il successo di questa tipologia di software e dispositivi è specialmente dovuto alla loro capacità di controllare ed automatizzare qualsiasi processo ed operazione, riducendone costi e tempi, e questo ha fatto sì che moltissime compagnia ripensassero e rimodernassero il loro approccio al business mediante l'adozione di queste tecnologie, a partire dal settore dell'entertainment fino ad arrivare a quello militare ed ospedaliero.

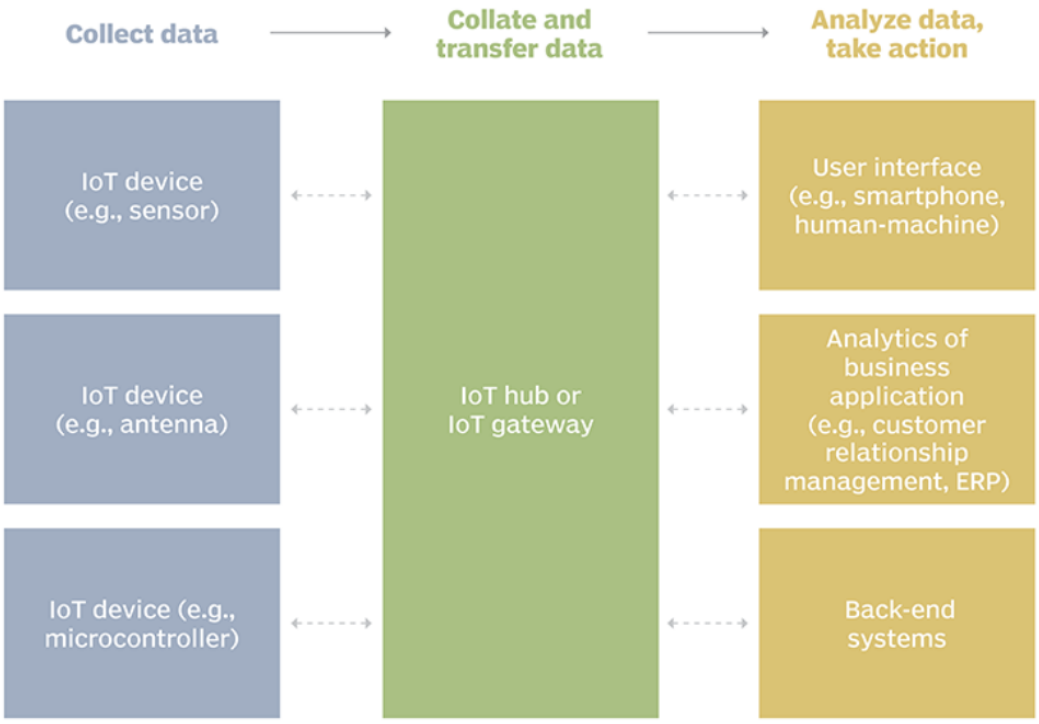

Figura 1.1: Immagine riassuntiva del principio di funzionamento di un sistema IoT [42]

La crescente tendenza nell'utilizzo di questi dispositivi presenta però un problema, purtroppo ancora troppo trascurato, ovvero quello della sicurezza e della privacy degli utenti che ne usufruiscono. La maggioranza dei dispositivi IoT chiede agli utenti di fornire informazioni private sensibili per personalizzare e specializzare i servizi che vengono forniti, e questo traffico di dati non è quasi mai coperto da meccanismi di sicurezza, che appunto risultano essere inadeguati o persino assenti, specie nel contesto di utilizzo commerciale. In altri casi d'utilizzo, ci sono dispositivi che tracciano movimenti, argomenti di discussione e persino transazioni bancarie, e talvolta queste operazioni sono effettuate in presenza di criteri per la tutela degli utenti inadeguati, che mettono in grave pericolo la loro integrità.
Per dare un'idea del potenziale pericolo che l'utilizzo di queste tecnologie porta con sé, basta guardare al 21 Ottobre del 2016, quando avvenne uno degli attacchi informatici più pericolosi mai registrati [15]. In questa data, un grandissimo numero di dispositivi IoT infetti sotto il controllo di una botnet, chiamata *Mirai*, colpì il provider DNS Dyn (uno dei maggiori provider Internet americani) con una attacco DDoS senza precedenti, con conseguenze che si riflessero su varie aziende di alto profilo, fra cui GitHub, Twitter, Reddit, Netflix, rendendo inutilizzabili temporaneamente i lori servizi, e arrivando a creare gravissimi problemi anche al provider Internet OVH, il più importante d'Europa.

Per rendere questi dispositivi più resilienti ed affidabili rispetto alla pericolosità dei furti di dati e gli attacchi informatici a cui possono essere sottoposti, la questione della sicurezza deve essere affrontata già a partire dalla fase di sviluppo di tali dispositivi. Inoltre, a causa dell'eterogeneità dei dispositivi IoT, insieme al limitatezza in termini di hardware e risorse computazionali degli stessi, un sistema di sicurezza universale risulta ancora più difficile da realizzare e diffondere.
Considerando i problemi e i limiti sopra citati, l'applicazione di tecniche di crittografia o la modifica di hardware risultano assolutamente inapplicabili ai dispositivi già presenti sul mercato e in utilizzo da parte degli utenti. Tenendo conto che sistemi di antivirus e similari non possono essere applicati a dispositivi con una capacità computazionale ed energetica limitata, l'unica strada percorribile è quella di sviluppare un sistema di Intrusion Detection, che si propone come un software leggero che può essere facilmente accostato ad un dispositivo IoT per tracciare parametri come utilizzo della CPU, consumo della memoria, traffico dei pacchetti di rete, con lo scopo di segnalare la presenza di malware e/o un eventuale attacco.
In considerazione della varietà e del polimorfismo dei dispositivi del mondo IoT, delle dinamicità delle reti in cui vengono installati, e della velocità con cui gli stessi malware si evolvono, un'ottima soluzione può essere quella di proteggere in modo altrettanto dinamico i sistemi dell'Internet of Things sfruttando l'Intelligenza Artificiale. Tramite un modello intelligente, è infatti possibile creare un sistema che si adatti facilmente ai nuovi tipi di attacchi, e che tramite 'esperienza' sappia addirittura rilevare attacchi di cui un sistema non è mai stato vittima.

1.2 Motivazioni e Obiettivi

Allo stato attuale, l'uso di l'Intelligenza Artificale applicata alla cybersecurity è un campo piccolo e ancora in fase di rodaggio, ma in rapida crescita. In tal senso, alcuni tentativi di utilizzo sono già stati effettuati per migliorare la sicurezza dei dispositivi IoT, ad esempio sono stati utilizzati i Decision Tree e reti neurali per l'identificazione di attacchi DoS, la Support Vector Machine per la simulazione di attacchi e il testing di integrità dei dispositivi [22]. Una della principali motivazioni per cui tecniche di questi tipo risultano essere poco diffuse è a causa del loro elevato costo di realizzazione e della loro difficoltà di applicazione, in particolare a piccoli sistemi.

In considerazione delle mancanze discusse in letteratura e del grave problema di sicurezza di cui soffre il mondo IoT, si è scelto di agire in maniera diretta, implementando un software ad-hoc che utilizzi l'Intelligenza Artificiale per identificare la presenza di malware nei dispositivi IoT. Più in particolare, sarà progettato, implementato e testato un sistema di *Intrusion Detection*, che a partire da un set di dati, sia in grado di rilevare la presenza di uno o più malware a partire da diversi dispositivi di IoT, e che sia facilmente aggiornabile ed estendibile. Il tipo di dati a partire dai quali sarà effettuata l'implementazione saranno rigorosamente relativi al traffico di dati in entrata e in uscita dei dispositivi. Questa scelta è stata fatta in quanto ogni dispositivo IoT è basato su piattaforme, software e hardware diversi, mentre l'analisi del traffico è un operazione che può essere effettuata universalmente e senza particolari problemi.
Per lo sviluppo dello stesso, l'intenzione è di produrre un software affidabile, fruibile e di qualità, motivo per il quale si é scelto di seguire l'approccio ingegneristico CRISP-DM, con lo scopo di ottimizzare le fasi di progettazione e massimizzare il risultato ottenuto. In merito al suddetto approccio, sono stati definiti i seguenti *obiettivi di business*:

- i dati su cui l'agente intelligente sarà addestrato dovranno essere dati reali e autentici, ottenuti da un'autentica presenza di malware, per ottenere un modello affidabile in casi d'uso reali
- inoltre, l'agente sarà addestrato con dati relativi ad almeno 3 diversi dispositivi, anch'essi reali, per avere un modello quanto più flessibile possibile
- considerando le metriche più diffuse, quali accuracy, precision, recall e F1 score, si deve ottenere una media dei valori non inferiore al 90%
- a prescindere dalla combinazione ottima di algoritmi di normalizzazione, preprocessing e classificazione, saranno implementati e messi a disposizione diversi algoritmi per tipologia, in modo da fornire diverse tecniche fruibili e testabili all'utilizzatore
- il sistema metterà a disposizione diversi tipi di grafici, come matrici di confusione e matrici di correlazione, per dare la possibilità di poter effettuare ulteriori valutazioni in modo rapido ed intuitivo

Dopo un'opportuna ricerca, per l'addestramento è stato individuato un dataset contenente dati relativi a 9 diversi dispositivi IoT, ottenuti a partire dall'analisi del traffico dei pacchetti di rete di ognuno di essi [27]. Utilizzando questi dati, è possibile individuare 10 diversi classi di attacchi, appartenenti alle famiglie Mirai e BASHLITE.

1.3 Risultati ottenuti

Per ottenere dei risultati affidabili per il nostro sistema di Intrusion Detection, sono state eseguite in totale 9 run per ogni classificatore implementato, tre per ognuno dei dispositivi selezionati (Provision, Philips ed Ennio). Il dataset si è rivelato essere ben bilanciato, e con un minimo intervento di Data Cleaning e la giusta combinazione di algoritmi di normalizzazione e preprocessing è stato possibile ottenere buoni risultati con tutti i classificatori. In totale, sono stati implementati 7 diversi classificatori, ed il migliore considerando il tipo di dataset ed il problema in esame si è rivelato essere il **K-Nearest Neighbors**. Utilizzando questo algoritmo, combinato con gli algoritmi Z-Score e K-Best Selection rispettivamente per normalizzazione e preprocessing, si è raggiunto l'ottimo risultato di 99.85% fra accuracy, precision, recall ed F1 score, dunque per la sua natura di funzionamento è risultato essere perfetto per il nostro problema.
I risultati ottenuti ci danno uno spunto di riflessione su come, utilizzando un'apposita API per l'analisi del traffico dei pacchetti (*pcap* nel caso del dataset scelto ed utilizzato per l'addestramento), sia possibile ottenere dati sullo stato attuale della rete e determinare se il proprio sistema IoT sia infetto da una botnet. Tuttavia, la mancanza di apposite infrastrutture e iniziative che si pongano come obiettivo quello di studiare lo stato delle reti in presenza delle molteplici famiglie di malware che affliggono il mondo IoT, insieme alle politiche conservative delle aziende produttrici, costituiscono l'unico fattore limitante per questo e per tutti i sistemi di Intrusion Detection, in quanto é proprio la mancanza di dati utili alla prevenzione e alla salvaguardia la causa delle numerose criticità di sicurezza presenti in questa tipologia di dispositivi.

1.4 Struttura della tesi

Nei successivi capitoli della tesi verranno coperte le varie problematiche, questioni e dettagli relativi al mondo dell'IoT.
Nel capitolo 2, vengono coperte tematiche relative allo stato dell'arte e le informazioni presenti in letteratura sugli aspetti di ricerca trattati nel contesto di privacy e sicurezza dei sistemi IoT. Sarà analizzata e descritta la struttura di questi sistemi insiemi alle piattaforme su cui si basano, passando per gli aspetti comunicativi e le relative problematiche, per poi discutere dei problemi di privacy e sicurezza, insieme alle modalità con le quali l'Intelligenza Artificiale può essere utilizzata come soluzione a queste mancanze.
Nel capitolo 3, sono descritte tutte le tecniche di validazione, normalizzazione, preprocessing e classificazione che sono state utilizzate per lo sviluppo ed il perfezionamento del sistema di Intrusion Detection, insieme con tutte le informazioni man mano scoperte in merito ai dati utilizzati, le scelte implementative effettuate e le problematiche via via riscontrate e risolte, il tutto strutturato attraverso le varie fasi del modello CRISP-DM.
Nel capitolo 4 sono presenti tutte le informazioni relative ai risultati ottenuti dal modello. In particolare verranno descritte e motivate le metriche utilizzate, saranno confrontati tutti i classificatori implementati, e verranno discusse le motivazioni per il quale il K-Nearest Neighbour risulta essere perfetto per il nostro problema, passando infine per i punti di forza e debolezza del modello costruito.
Nel capitolo 5, è presente una descrizione sommaria di ciò che è stato fatto a partire dalla progettazione del sistema di Intrusion Detection fino alla discussione dei risultati finali, e vengono effettuate delle considerazioni in merito alla fruibilità del software e in quale direzione muoversi per poterlo estendere e migliorare.

CAPITOLO 2

Stato dell'arte

In questo capitolo é illustrato lo stato dell'arte e i lavori presenti in letteratura sugli aspetti di ricerca trattati nel contesto di privacy e sicurezza dei sistemi IoT. Sarà anzitutto analizzata la struttura di questi sistemi, passando per le relative tipologie e le piattaforme su cui si basano, per poi passare agli aspetti di comunicazione, e la discussione dei principali problemi di questo mondo. Saranno poi discussi nel dettaglio gli aspetti di sicurezza e privacy, analizzando i principali problemi in quest'ambito, ed esaminando i malware che affliggono l'IoT. Infine, valuteremo l'utilità data dall'impiego di diverse tecniche di Intelligenza Artificiale per risolvere i problemi di privacy e sicurezza presenti.

2.1 Anatomia dei sistemi IoT

L'Internet of Things (IoT) è un paradigma comunicativo emergente, che mira a connettere diversi tipi di oggetti e dispositivi alla rete, con lo scopo di recuperare dati generati da sensori, controllare macchinari e apparecchiature, monitorare ambienti, veicoli e infrastrutture, e così via, il tutto senza richiedere alcuna interazione da parte dell'uomo. Il numero e la varietà di dispositivi IoT, insieme alle tecnologie su cui si basano, sono esponenzialmente aumentate negli ultimi anni, e si stima una crescita ancora maggiore nel futuro [45]. Se è vero che tutti gli oggetti possono diventare "intelligenti" connettendosi alla rete e scambiando informazioni su di sé e sull'ambiente circostante, è altrettanto vero che questo processo non avviene in tutti gli ambiti con la stessa velocità: ciò dipende dall'esistenza di soluzioni tecnologiche consolidate, dagli equilibri competitivi in un determinato mercato, e dal bilancio tra il valore dell'informazione e il costo di creazione della rete di oggetti intelligenti.

Ciò nonostante, chiunque al giorno d'oggi dispone di un almeno un dispositivo IoT nelle proprie case, ed il loro utilizzo è ormai consolidato nella attività quotidiane di chiunque, a partire dai comuni assistenti vocali, passando per telecamere smart in grado di notificarci eventuali presenze indesiderate, fino a passare per complesse apparecchiature ospedaliere che avvisano in caso di pericoli per la salute di un paziente. I vantaggi quanto a interconnettività, automatizzazioni, velocizzazione di qualsivoglia attività sono un fattore a cui oramai non possiamo fare a meno [52].

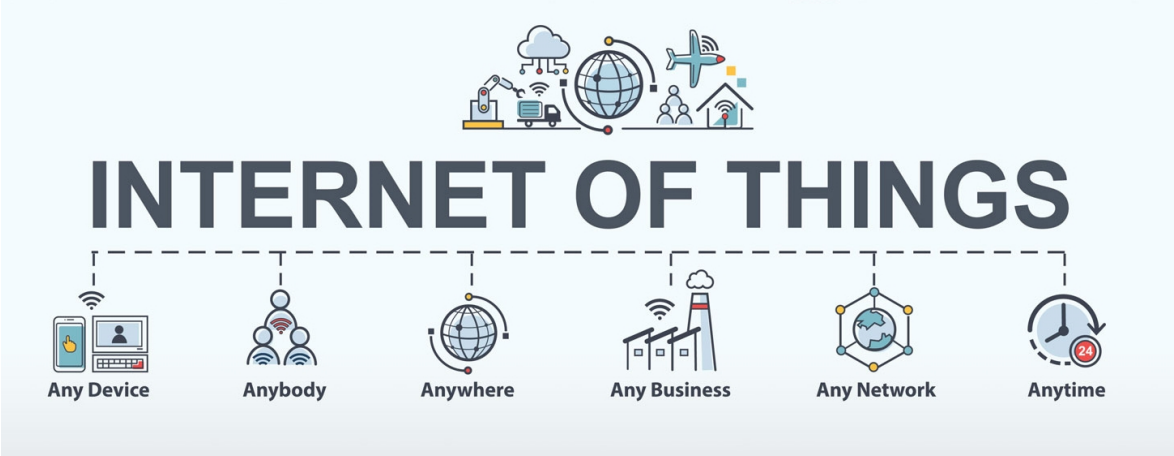

Figura 2.1: Immagine riassuntiva dell'Internet of Things

2.1.1 Struttura dei sistemi IoT

Le applicazioni per l'IoT, come visto nella precedente sottosezione, si estendono fra una grandissima varietà di casi d'uso. Tuttavia, i sistemi IoT presentano la medesima struttura, in quanto rappresentano l'integrazione di quattro componenti distinti [48], quali **sensori/dispositivi**, **connettività**, **data processing**, e **interfaccia utente** (vedi Figura 2.2).
Di seguito, vedremo più nel dettaglio ogni componente di cosa si occupa e come interagisce con le altre:

- **Sensori/Dispositivi** costituiscono la componente che raccoglie dati dall'ambiente, i quali possono essere semplici, come la lettura della temperatura, o complessi, come un stream di frame di un video.
  Sono definiti "sensori/dispositivi" perché multipli sensori possono essere uniti insieme, oppure i sensori possono essere solo parte di un dispositivo che va oltre la semplice rilevazione di determinate situazioni e eventi.

- **Connettività** è la componente responsabile di inviare i dati al cloud una volta che questi sono stati raccolti dai sensori/dispositivi. Affinché avvenga quest'invio di dati, sensori/dispositivi devono essere connessi al cloud, e tale connessione può avvenire in diverse modalità (che vedremo nella sottosezione successiva), da scegliere in base ai tradeoff fra consumo energetico, raggio, e larghezza di banda.

- **Data processing** costituisce la componente in grado di processare i dati una volta che questi sono arrivati al cloud. Questo processo può essere semplice, nel caso della lettura della temperatura, per cui va controllato solo se il range è accettabile, o complesso nel caso di un video per cui vanno identificati eventuali oggetti/persone presenti nello stesso.

- L'**interfaccia utente** è la componente in cui l'informazione è resa utile per l'utente finale. Un utente possiede appunto un'interfaccia, la quale permette di controllare in modo pro-attivo il sistema, ad esempio permette di leggere eventuali alert relativi alla temperatura, o controllare i feed video della propria abitazione. In relazione al tipo di sistema IoT, l'utente può anche agire, attraverso l'interfaccia, per "modificare" il sistema, ad esempio l'utente potrebbe modificare la temperatura di un frigorifero smart da remoto. Alternativamente, un'operazione di questo tipo potrebbe anche essere effettuata in modo automatico, e l'utente in tal caso riceverà semplicemente una notifica dell'avvenuto cambiamento della temperatura.

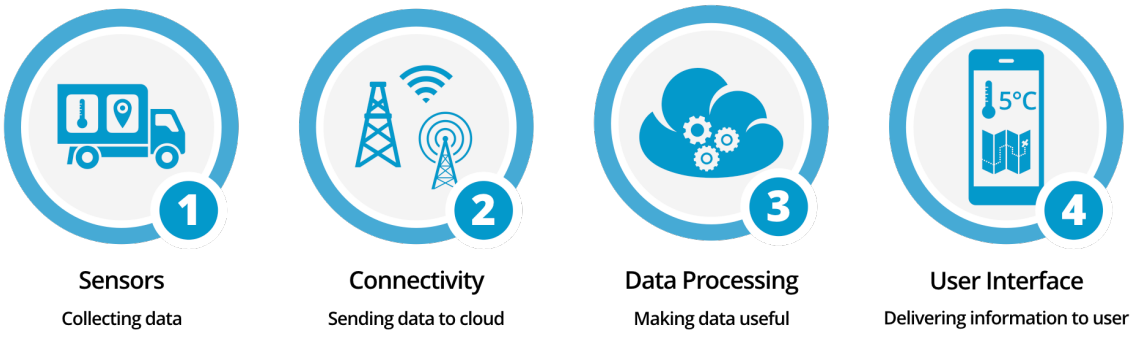

Figura 2.2: Componenti di un sistema IoT

Il vero potenziale di un sistema IoT é rilasciato quando lo stesso viene integrato con sistemi e flussi di dati esterni, ed è dunque fondamentale che tutti questi componenti diversi fra loro si leghino insieme in modo efficace e gestibile, e soprattutto che ci sia una coordinazione efficiente fra gli stessi.
A tal proposito, le componenti sopra descritte comunicano e sono mediate attraverso una **piattaforma IoT** [18]. Le piattaforme IoT collegano i dati raccolti alle applicazioni degli utenti finali, facilitandone la raccolta, l'archiviazione, l'elaborazione, la gestione e la visualizzazione, e garantendo il flusso di comunicazione tra le diverse componenti, l'integrazione dei diversi sistemi IoT, il monitoraggio dei punti di accesso alla rete, e la gestione delle applicazioni e la sicurezza dell'architettura. Ad alto livello, il vantaggio più rilevante offerto dalle piattaforme IoT è l'implementazione di un'unica architettura per la gestione di dati e servizi, che porta come risultato ad una migliore gestione delle risorse con conseguente aumento dell'efficienza e miglioramento degli aspetti di manutenzione e sicurezza.
Più nello specifico, una piattaforma IoT permette di [17]:

- connettere hardware, come sensori e dispositivi
- rendere efficiente la comunicazione e il flusso dei dati
- supportare la gestione dei dispositivi e le funzionalità dell'applicazione
- gestire diversi protocolli di comunicazione hardware e software per interfacciarsi con l'esterno
- fornire sicurezza e tecniche di autenticazione per i dispositivi e per gli utenti
- integrare tutto quello descritto sopra con sistemi esistenti e altri servizi web

Le piattaforme non sono uniche nel mondo dell'IoT, infatti al giorno d'oggi, sul mercato, sono già presenti aziende "dominanti" che offrono questo tipo di servizio, quali Amazon con AWS IoT Core, Microsoft con Azure IoT Hub, e Google con Google IoT Core (ovviamente ne esistono altre centinaia, seppur meno note). Queste piattaforme, però, sono più concentrate sul livello di infrastruttura, e richiedono una conoscenza avanzata ed un certo grado di personalizzazione da parte degli sviluppatori che vogliono costruire applicazioni IoT con scopi di business. Nella maggior parte dei casi, le piattaforme IoT sono costruite al di sopra di questi provider di infrastrutture, offrendo tool e servizi addizionali e complementari per lo sviluppo di applicazioni.

Potendo dunque affermare che un sistema IoT sia un "sistema di sistemi", ovvero una rete di dispositivi e software, ed essendo quasi impensabile che una certa organizzazione fornisca servizi relativi a tutti i domini applicativi, le piattaforme IoT costituiscono una componente fondamentale per le aziende che necessitano di una o più infrastrutture pronte all'uso, per evitare i problemi ed i costi legati all'ingaggio di un team di sviluppo specializzato nei vari campi dell'IoT per lo sviluppo di una propria soluzione. Ciò nonostante, va considerato il tradeoff dato dal costo necessario ad usufruire di una certa piattaforma rispetto al tempo necessario a sviluppare quello stesso servizio in maniera indipendente, che dipenderà ovviamente dalla complessità del particolare servizio di cui si necessita e dal costo nel tempo dell'infrastruttura eventualmente utilizzata [18].

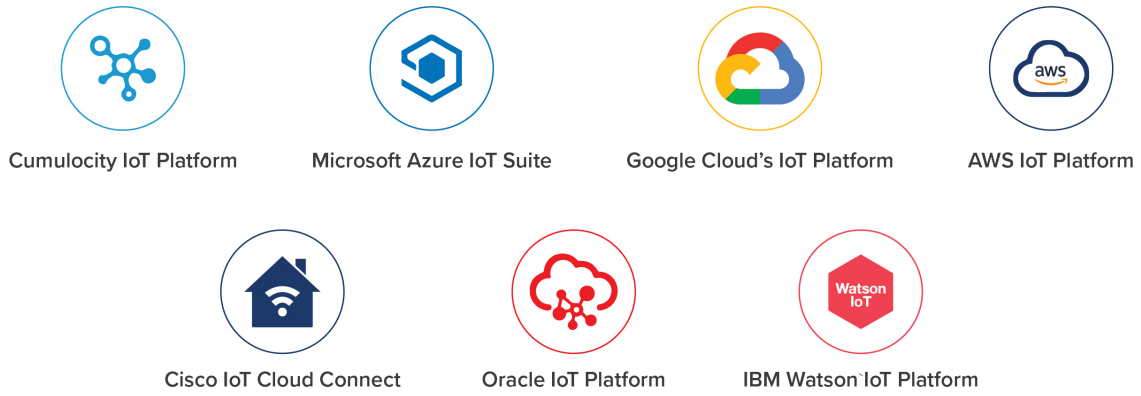

Figura 2.3: Le principali piattaforme IoT attualmente presenti sul mercato

2.1.2 Comunicazione nei sistemi IoT

L'efficienza e la validità di un sistema IoT sono fortemente basati sul tipo di connettività utilizzata dalle componenti di cui il sistema è costituito. Essendo l'Internet of Things un mondo vario e con moltissime declinazioni, è impossibile trovare un protocollo di comunicazione universalmente valido ed utile in tutte le situazioni.
A tal proposito esistono diverse soluzioni, ognuna della quali presenta i suoi punti di forza e le sue debolezze, ed é migliore piuttosto di un altro in relazione allo specifico caso d'uso. Scegliere la tecnologia migliore da utilizzare per lo specifico caso d'uso richiede una valutazione accurata dei tradeoff tra larghezza di banda, qualità del servizio, consumi di energia e gestione della rete. Di seguito, sono elencati i tipi di comunicazione più comuni [5]:

- **Ethernet** [44] è la tecnologia utilizzata quando è richiesto il massimo della velocità di trasferimento e la larghezza di banda, che non si potrebbe ottenere con nessuna tecnologia wireless. Costituisce una soluzione a basso costo, sicura, affidabile, e veloce (con l'Ethernet si arriva addirittura a 400 Gbps, a differenza del Wi-Fi che si aggira attorno ai 10 Gbps), che ove applicabile permette una facile connessione dei dispositivi di un sistema IoT, oltre ad essere utilizzata in situazioni in cui una connessione wireless non risulterebbe sufficiente per collegare tutti i dispositivi necessari, in quanto interferenze e alte latenze comprometterebbero le performance.
  È ampiamente utilizzata per applicazioni IoT basate su Wi-Fi, che connettono access point wireless tramite un'infrastruttura collegata via cavo.

- **LPWAN** (**Low Power Wide Area Networks**) costituisce un protocollo nuovo nel mondo IoT, in grado di fornire una comunicazione versatile a lungo raggio utilizzando batterie durevoli, piccole ed economiche, ed è stato ideato per supportare reti IoT su larga scala ed applicabili in contesti industriali e commerciali. Purtroppo, con questo protocollo è possibile inviare solo piccoli blocchi di dati a bassa velocità, e dunque sono da utilizzare nei casi in cui non è richiesta una grande larghezza di banda.
  Le LPWAN sono utilizzabili per collegare letteralmente qualsiasi tipo di sensore, facilitando applicazioni che si occupano di tracciamento di risorse, monitoraggio dell'ambiente, gestione di strutture, etc.

- **Cellulari** (3G, 4G, **5G**) costituiscono un protocollo largamente utilizzato sul mercato attuale, che offre una comunicazione a banda larga affidabile in grado di supportare chiamate vocali e videochiamate. Di contro, presentano costi operazionali e requisiti di

energia molto alti.

Mentre le reti cellulari non sono utilizzabili per la maggior parte delle applicazioni IoT con dispositivi che utilizzano batterie, sono ideali nel caso di auto connesse, gestione di flotte e logistica.

Un focus particolare va fatto per le reti **5G**, che grazie all'elevatissima velocità e alla bassissima latenza, si candida ad essere il futuro per i veicoli autonomi e la realtà aumentata. Inoltre, permette la realizzazione di sistemi di sorveglianza in tempo reale per la sicurezza pubblica, l'invio in tempo reale di dati per la sanità, l'automazione industriale, etc.

- **ZigBee** è uno standard wireless a corto raggio e bassa potenza, rilasciato tipicamente in topologie a maglia, con lo scopo di estendere la copertura tramite l'inoltro dei dati di un sensore sui nodi di una rete di dispositivi IoT.

  Confrontata con la LPWAN, ZigBee presenta velocità di trasmissione più elevate, ma allo stesso tempo un'efficienza minore a causa della configurazione a maglia. A causa del corto raggio (minore di 100 metri), ZigBee è più indicato per applicazioni IoT a medio raggio con una distribuzione uniforme di nodi. A tal proposito, costituisce un perfetto complemento ai sistemi WiFi per le abitazioni domotiche, per gestire le varie componenti smart.

- **Bluetooth** è una nota tecnologia di comunicazione wireless a corto raggio. Permette uno scambio di dati di tipo point-to-point e point-to-multipoint, è ottimizzata per il consumo di energia, ed è particolarmente utilizzata, insieme a Zigbee, nel contesto del Consumer IoT (che vedremo dopo). I dispositivi Bluetooth sono quasi sempre utilizzati in congiunzione con i dispositivi elettronici (tipicamente smartphone), che fungono da hub per il trasferimento dei dati verso il cloud.

  Il Bluetooth oggi è ampiamente integrato nei dispositivi indossabili per la salute e lo sport, così come dispositivi smart usati nelle abitazioni, dove è conveniente che i dati siano appunto trasferiti e visualizzati su smartphone.

- **Wi-Fi** è una tecnologia wireless a medio/corto raggio, ed ha un ruolo critico nel fornire una velocità di trasferimento elevata sia in contesti domestici che aziendali. Con il progredire di questa tecnologia, specie con l'avvento del Wi-Fi 6, il numero di soluzioni che la sfruttano è esponenzialmente aumentato, e ciò la rende oggigiorno la tecnologia in assoluto più utilizzata nel contesto dell'IoT. In particolare, il Wi-Fi risulta essere più diffuso su reti di dispositivi che sono connessi ad una fonte elettrica, come gadget

domestici smart, telecamere di sicurezza, contrassegni digitali, etc., sebbene si stia diffondendo anche sui dispositivi a batteria grazie ad un'efficienza sempre migliore nel consumo di energetico. Ciò nonostante, a causa dei limiti in merito a copertura e scalabilità, risulta essere la tecnologia meno utilizzata nel contesto dell'Industrial IoT (vedi sottosezione successiva).

- **RFID** (**Radio Frequency Identification**) è una tecnologia a corto raggio, che utilizza onde radio per trasmettere piccole quantità di dati da un tag RFID verso un lettore a distanza ravvicinata. Questa tecnologia ha dato luogo ad una vera e propria rivoluzione nel settore della logistica e della vendita al dettaglio. Infatti, attaccando un tag RFID ad un qualsiasi prodotto, le aziende possono tener traccia del proprio inventario e delle proprie risorse in tempo reale, permettendo una migliore pianificazione della produzione ed un'ottimizzazione della gestione delle catene di approvvigionamento.

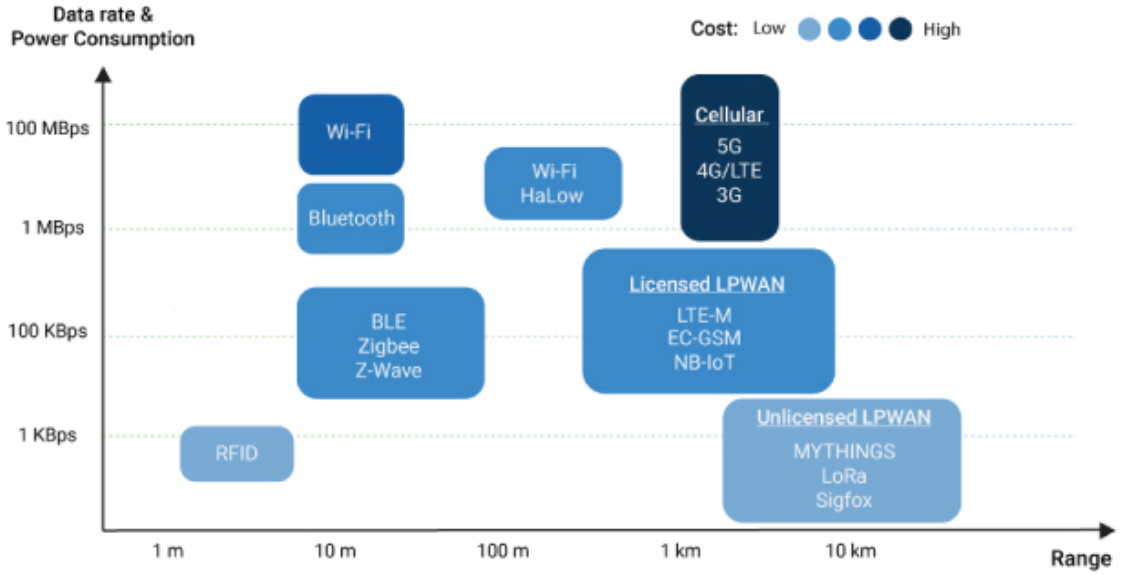

Figura 2.4: Rapporto fra copertura e velocità di trasferimento/consumo d'energia delle varie tecnologie di comunicazione

2.1.3 Tipologie di sistemi IoT

Come abbiamo visto, i sistemi IoT hanno un potenziale tale da poter essere applicati in quasi tutte le attività svolte dall'uomo, a partire dal settore entertainment fino ad arrivare ad impieghi in ambito ospedaliero e militare.
Dato che i sistemi IoT condividono la stessa struttura e gli stessi principi di funzionamento, questi ultimi andranno differenziati e specializzati in relazione al tipo di utenti al quale sarà destinato lo stesso, per cui presenteranno standard diversi di sicurezza, diversi tipi di interfacce, determinate tipologie di tecnologie comunicative, etc.
A tal proposito, è possibile individuare le seguenti categorie [29]:

- **Consumer IoT** (**CIoT**), che si riferisce all'uso di tecnologie IoT per dispositivi e applicazioni per consumatori. Alcuni prodotti di CIoT comprendono smartphone, assistenti intelligenti, elettrodomestici, etc. Tipicamente, le soluzioni CIoT utilizzano prevalentemente WiFi, Bluetooth e ZigBee per la connettività, in quanto offrono una comunicazione a corto raggio in luoghi piccoli.

- **Commercial Internet of Things**, che piuttosto che concentrarsi sul miglioramento degli ambienti personali e casalinghi, porta i benefici dell'IoT a luoghi più grandi, quali uffici, supermercati, hotel, ospedali, centri commerciali, etc. Ci sono innumerevoli casi d'uso per il Commercial IoT, come il monitoraggio delle condizioni ambientali di un certo luogo, o la gestione degli accessi a luoghi privati, dunque queste soluzioni puntano al miglioramento dell'esperienza dei consumatori e delle condizioni di business.

- **Industrial IoT** (**IIoT**), che può essere considerata come la branca del mondo IoT più dinamica. Il suo focus è sul miglioramento dei sistemi industriali esistenti, per renderli più efficienti e produttivi. Le distribuzioni IIoT sono generalmente localizzate in grandi fabbriche e stabilimenti produttivi, e sono spesso associate a industrie di settori quali salute, agricoltura, automobili, e logistica.

- **IoT di Infrastruttura**, che può essere considerata una branca dell'Industrial IoT, e si concentra sullo sviluppo di infrastrutture smart che incorporano tecnologie IoT, con lo scopo di aumentare efficienza e risparmio. Questo tipo di sistemi hanno la capacità di monitorare e controllare le operazioni delle infrastrutture urbane e rurali, quali ponti, binari ferroviari, centrali eoliche, dighe, etc.

- **Internet of Military Things** (**IoMT**), che riguarda l'uso dell'IoT in contesti militari e situazioni di battaglia. Questo tipo di tecnologia si pone come obiettivo quello di

migliorare la consapevolezza dell'ambiente di guerra, la valutazione dei rischi, e i tempi di risposta in situazioni critiche. Alcune applicazioni dell'IoMT includono la connessione fra navi, aerei, blindati, droni, soldati, e persino basi operative, tramite un sistema interconnesso, o l'utilizzo di sistemi che producano dati che possono essere sfruttati per migliorare pratiche, strategie, ed equipaggiamenti militari.

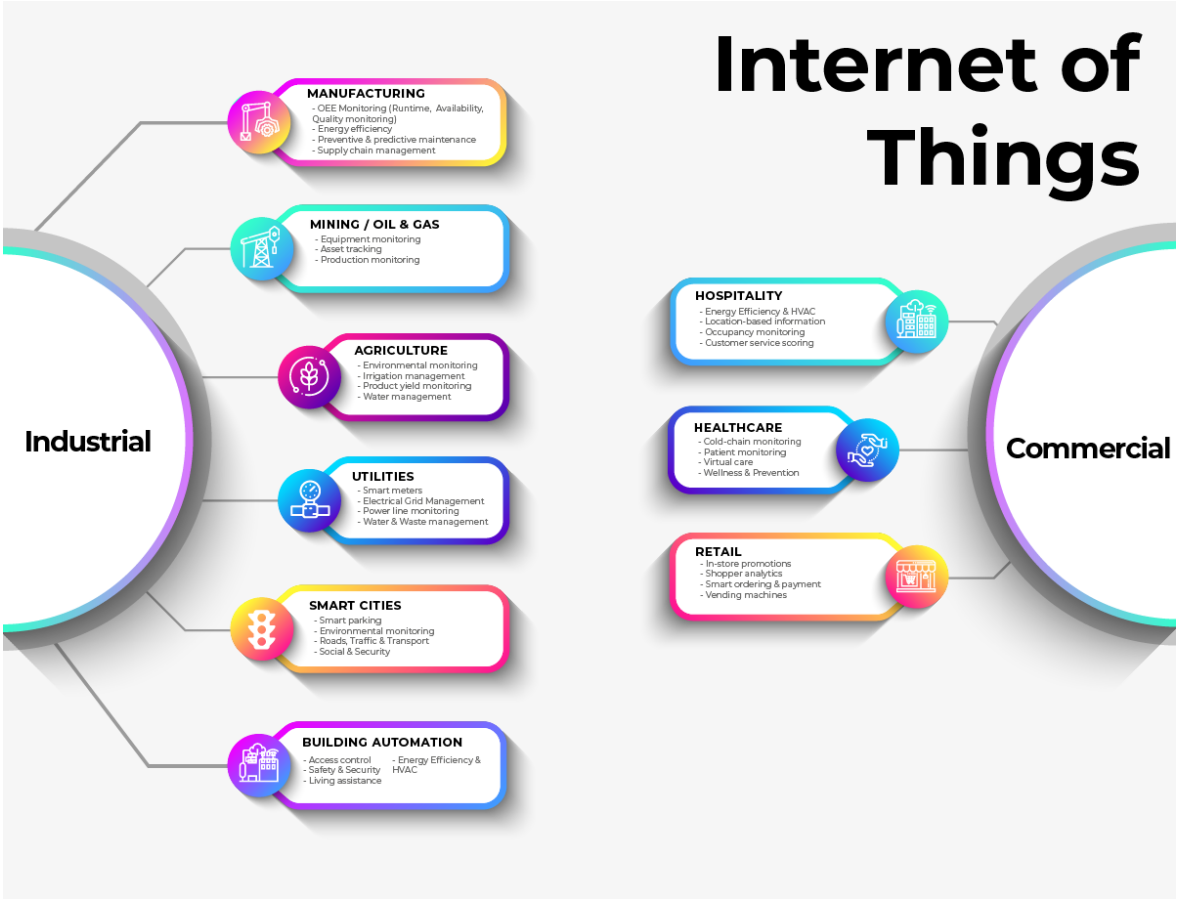

Figura 2.5: Alcuni dei settori di impiego dei sistemi IoT

2.1.4 Problematiche dei sistemi IoT

Una qualsiasi azienda, prima di investire in un upgrade adottando appunto un sistema IoT, deve innanzitutto considerare come il suo impiego e utilizzo impatterà le strategie aziendali. Nonostante abbiamo spiegato come il mondo dell'IoT possa portare notevoli vantaggi in tutte le attività della vita quotidiana, è fondamentale concentrarsi non sui vantaggi, ma piuttosto su eventuali svantaggi e criticità che l'utilizzo di un sistema IoT può portare.
A tal proposito, valuteremo nei seguenti punti quelli che sono i principali problemi. Nella seguente sezione, approfondiremo nel dettaglio i concetti di privacy e sicurezza [10].

- **Maturità**, in quanto la digitalizzazione delle aziende, nella maggior parte dei casi, risulta essere costosa a causa delle tecnologie da impiegare, e difficile a causa della scarsa disponibilità di esperti del settore. Essendo l'IoT un mondo in continua evoluzione, spesso le aziende sono restie ad adottare soluzioni in tale ambito, e quelle che lo fanno talvolta possono avere esperienze negative.

- **Mancanza di competenze**, in quanto l'IoT richiede una vasta gamma di competenze, software e sistemi che la maggior parte delle aziende non possiede, come ad esempio applicazioni in grado di processare i dati provenienti dai sensori, applicazioni per interfacciarsi con i sistemi, analisti ed esperti di dati per la gestione gli stessi, etc.

- **Vulnerabilità dei software**, infatti molti dispositivi IoT hanno software di bassa qualità, che sono causa di un elevatissimo numero di vulnerabilità, quali software exploit, cattivo uso della crittografia, autenticazioni fallite, etc.
  Queste debolezze possono portare ad attacchi a sistemi IoT, che espongono le reti su cui risiedono, che esse siano domestiche o aziendali. Questo punto sarà discusso in modo più approfondito nella sezione successiva.

- **Complessità**, in quanto l'elevato numero di opzioni di connettività nate col tempo risultano essere più un problema che un aiuto. Infatti, coloro che desiderano utilizzare un sistema di IoT devono scegliere se limitarsi ad utilizzare una cerchia ristretta di dispositivi o tecnologie, o se sforzarsi nell'impiego di diverse opzioni. Inoltre, la maggior parte dei dispositivi oggi connessi risultano avere una potenza computazionale troppo bassa per supportare i meccanismi di protezione richiesti al giorno d'oggi, il che richiede ai programmatori di adottare dei meccanismi di sicurezza alternativi e più complessi da progettare.

- **Interoperabilità**, in merito alla quale è sempre risultato essere presente una disparità rispetto all'elevato numero di dispositivi connessi da parte dei consumatori. La scarsa presenza di standard di interoperabilità per la creazione di sistemi intelligenti ha portato problemi specie nei contesti aziendali, in quanto se ci fossero presenti soluzioni aperte e versatili, si potrebbero creare dei sistemi molto più robusti e sicuri. Altrettanto critici sono i sistemi IT che si collegano ai sistemi operativi, che senza uno standard opportuno per lo scambio di dati e la sicurezza, non permetteranno mai all'IoT di raggiungere il massimo potenziale.

- **Privacy** che è sempre stata un problema che ha tormentato i sistemi IoT, sebbene i sistemi offrano ai consumatori un'esperienza inconfrontabile mediante l'accesso a questi dati sensibili. Il funzionamento di base dei sistemi IoT prevede che i dispositivi debbano inviare dati sull'internet, che è un luogo esposto a moltissimi rischi di attacchi e fughe di dati, le quali possono gravemente danneggiare aziende e mettere a rischio la privacy di ogni individuo.
  Sebbene l'impiego di una tale tecnologia richieda tempo e costi notevoli, i sistemi IoT sono tenuti a condividere informazioni utilizzando crittografie di prim'ordine per evitare questo tipo di problemi.

2.2 Questioni di privacy e sicurezza

L'Internet of Things é un mondo in costante evoluzione, al quale si aggiungono dispositivi giorno dopo giorno. Entro il 2025, ci si aspetta di avere circa 64 miliardi di dispositivi IoT in uso [7], e il loro impiego beneficia e supporta sempre più tutte le attività svolte ogni giorno dall'uomo.

Il mondo dell'IoT, che si pone come filosofia quella di fornire un alto livello di accessibilità, integrità, disponibilità, scalabilità e interoperabilità in termini di connettività [22], ha introdotto anche rischi notevoli proprio a causa di questo numero crescente di dispositivi, che in determinati ecosistemi può rappresentare un pericolo per la sicurezza e la privacy, in quanto ogni dispositivo di un sistema IoT costituisce per gli hacker un potenziale **entry point** da cui attuare violazioni e attacchi. Con entry point si intende il punto della rete o del flusso di dati a partire del quale un hacker è in grado di infiltrarsi nell'infrastruttura con scopi malevoli, il che può avvenire a livello delle applicazioni, nel cloud, nei dispositivi e nelle loro comunicazioni (Figura 2.6), con lo scopo di compromettere la sicurezza di una rete e di tutti gli utenti che ne usufruiscono.

Le preoccupazioni relative alla sicurezza e i problemi di privacy nel contesto dell'IoT presenta considerevoli implicazioni per le organizzazioni imprenditoriali e pubbliche, e a tal proposito c'è da lavorare per migliorare questi aspetti, specialmente per le soluzioni rivolte ai consumatori, con lo scopo di incoraggiarli a fidarsi di questi sistemi, anche alla luce della crescente consapevolezza che hanno gli utenti in relazione alla privacy dei propri dati.

In merito a tali questioni, nelle successive sottosezioni andremo ad analizzare nello specifico i problemi di sicurezza e privacy, insieme ad una spiegazione dei software responsabili della loro compromissione, con un particolare focus sulle reti botnet che affliggono l'IoT.

2.2.1 Problemi di sicurezza

Una delle chiavi principali per il successo dell'IoT è la fiducia che gli utenti avranno in questi sistemi. Se dal sistema e verso il sistema non sono presenti tecnologie, regolamentazioni e strumenti a garanzia dei requisiti di sicurezza, la maggior parte degli utenti non sarà disposta ad adottare il sistema stesso [8].
Molti dispositivi del panorama IoT (sensori, microfoni, telecamere, etc) sono concepiti per la distribuzione su larga scala, ed il fatto che presentano caratteristiche ed infrastrutture simili fra loro contribuisce a far emergere e persistere i problemi e le vulnerabilità di sicurezza. Inoltre, come detto precedentemente, a causa della interconnettività onnipresente nel panorama IoT, è sufficiente un solo dispositivo poco sicuro per compromettere la sicurezza di un intero sistema.
Essendo la componente di sicurezza fondamentale per far si che gli utenti abbiano fiducia nell'adottare soluzioni con l'IoT, di seguito sono elencati i problemi più rilevanti sui quali gli sviluppatori e le aziende dovrebbero lavorare [1]:

- **Protezione delle password inadeguata**, dovuta al fatto che le credenziali degli utenti vengono salvate in maniera hard-coded e vengono incorporate nei dispositivi IoT, che costituiscono un facile obiettivo per gli hacker, che possono compromettere in maniera diretta un dispositivo. Inoltre, l'utilizzo di password di default permettono agli hacker di entrare in un certo sistema senza alcun ostacolo.

- **Garanzie limitate** dai produttori di sistemi IoT, in quanto uno sviluppo software non curante degli aspetti di sicurezza porta inevitabilmente con sé lacune e problemi di sicurezza, mancanza di controlli nel trasferimento e lo storage di dati, tutti fattori che non fanno altro che aumentare i punti deboli di un sistema.

- **Gestione degli aggiornamenti dei dispositivi**, in quanto ci sono aggiornamenti che potrebbero portare con sé firmware e software che invece di migliorare, portano con sé potenziali rischi per vulnerabilità, andando dunque a compromettere la sicurezza. Inoltre, la gestione degli aggiornamenti può risultare un problema nel momento in cui vengono inviati dei dati di backup prima dello stesso sul cloud, operazione che se fatta senza un'opportuna connessione criptata e protezione per i file, potrebbe permette ad un eventuale agente malevole di accedere ad informazioni sensibili.

- **Mancanza di interfacce di sicurezza**, che dovrebbero essere presenti fra tutte le applicazioni, protocolli e servizi di comunicazione coinvolti nel processing e il trasferimento

di dati, ma che talvolta risultano essere assenti. I problemi più rilevanti in merito alla mancanza di interfacce si riscontrano con la mancanza di autenticazione e autorizzazione da parte dei dispositivi, e/o con meccanismi di crittografia deboli o totalmente assenti.

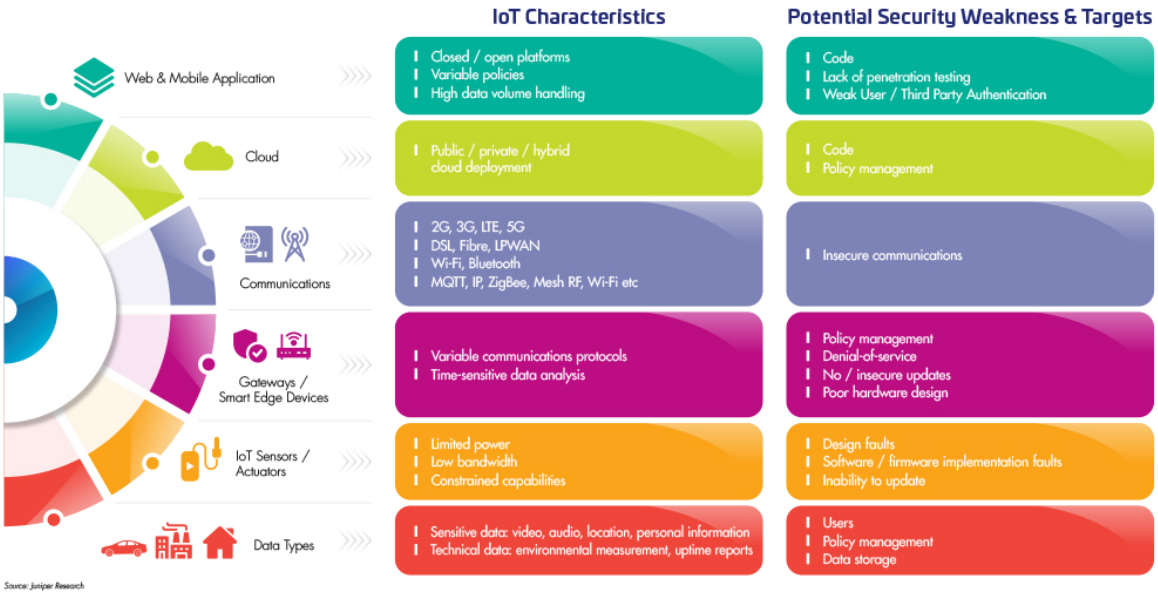

Figura 2.6: Caratteristiche del mondo IoT con i relativi problemi e mancanze [14]

2.2.2 Problemi di privacy

Molti consumatori considerano la validità dell'IoT in relazione alla capacità di salvaguardare e rispettare gli standard di privacy, e le assunzioni comuni relative a questo tipo di problemi dei sistemi IoT possono costituire un blocco nell'adozione di queste tecnologie.
Gli aspetti relativi alle garanzie di sicurezza e diritto alla privacy al giorno d'oggi sono di centrale importanza nei mondo dell'IoT, per gli utenti che tengono oggi più che mai ai propri dati, e per le aziende che per queste ragioni oramai si approcciano in maniera completamente diversa quando si tratta dello sviluppo di un sistema di questo tipo. In particolare, la connessione costante attraverso l'internet gioca un ruolo cruciale in questo contesto, in quanto senza un meccanismo univoco per la salvaguardia della privacy, l'accesso ad informazioni personali da qualsiasi angolo del mondo è più facile di quanto si immagini.
Le attuali sfide nel contesto della privacy sono relative alla definizione di una politica di collezione dei dati ed anonimizzazione degli stessi [51]. La prima si può ottenere tramite la definizione del tipo e della quantità di dati da collezionare, mentre la seconda é realizzabile tramite protezioni crittografiche e l'occultamento delle relazioni dei dati.
Malgrado si cerchi di applicare e migliorare le suddette tecniche, ci sono problemi di privacy che nel mondo dell'IoT sono comunque presenti, analizzati nei seguenti punti [1]:

- **Abbondanza di dati** generati dai sistemi di IoT. Per rendere l'idea, un numero di circa 10.000 abitazioni possono arrivare a produrre circa 150 milioni di entry point al giorno, pertanto è facile dedurre che la possibilità di essere vittima di una violazione è molto alta. Un potenziale hacker può scegliere gli entry point più vulnerabili, per poi accedere ad informazioni sensibili e/o per rendere vulnerabili i dispositivi IoT degli utenti.

- **Intercettazioni**, che sono più frequenti di quanto si possa immaginare, e sono considerati fra i pericoli più grandi per tutte le aziende che lavorano con dati sensibili [46]. Gli hacker possono penetrare all'interno di un sistema attraverso un entry point, ed utilizzare una determinata apparecchiatura smart per intercettare flussi di dati, conversazioni, stream video, etc. Addirittura gli stessi produttori e aziende, che hanno chiaramente un facile accesso a questi dati, possono invadere la vita privata di un individuo, e compromettere la sua privacy.

  Per citare un esempio, dei ricercatori, a scopo di apprendimento, sono riusciti ad intercettare dei dati non criptati da un contatore smart, e a partire dai quali sono riusciti ad identificare lo show televisivo che stava guardando la persona interessata, dati

relativi agli orari di entrata ed uscita dall'abitazione, e dei frammenti video di alcune telecamere indoor [1].

- **Esposizione pubblica indesiderata**, fenomeno causato essenzialmente dalle aziende produttrici con le documentazioni relative ai termini di servizio, che essendo inopportunamente lunghe e di difficile comprensione, di rado vengono lette per intero. Tali documenti hanno delle condizioni che (se accettate) permettono alle aziende e ai produttori di sfruttare i dati forniti "volontariamente" dai consumatori per effettuare analisi e previsioni.
  Ad esempio, una compagnia assicurativa potrebbe conservare informazioni relative agli spostamenti di un individuo nel tempo, e una tale informazione, se rubata e passata nelle mani sbagliate, potrebbe permettere a dei ladri di introdursi in un'abitazione quando non ci sono occupanti.

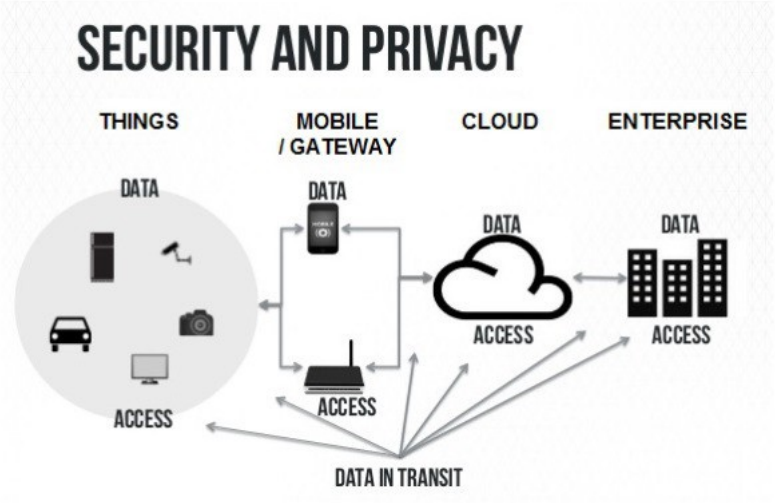

Figura 2.7: Il flusso dei dati nell'IoT e i relativi punti di transito

Le intercettazioni sono proprio l'oggetto di analisi di questa tesi, infatti il dataset utilizzato per addestrare il modello di intelligenza artificiale per il nostro sistema di Intrusion Detection contiene dati relativi a dispositivi IoT infettati da diversi malware, che andavano a creare una cosiddetta **botnet** [19], ovvero una rete di dispositivi "zombie" che possono essere controllati ed utilizzati per attacchi informatici, e che andavano appunto ad intercettare i flussi di dati che passavano fra questi dispositivi, esponendo gli utenti utilizzatori a gravissimi rischi per la loro sicurezza e privacy. Questo tipo di reti, insiemi ai malware sui quali sono basate, saranno approfondite nella sezione successiva.

2.2.3 Botnet e malware nell'IoT

Aldilà dei problemi di sicurezza ad alto livello discussi, i sistemi IoT sono costituiti da software, servizi e reti, e sono dunque vulnerabili ad attacchi software (tramite l'uso di malware), fisici (quando si agisce fisicamente per compromettere un sistema) e via rete. La maggior parte dei dispositivi IoT hanno dei fattori limitanti, quali potenza di calcolo, memoria, larghezza di banda e batteria, dunque risulta difficile l'impiego di soluzioni esistenti già consolidate in quanto richiedono un grande sforzo in termini computazione e comunicazione.
Proprio questo tradeoff costituisce il principale limite al grado di sicurezza presente nei sistemi di IoT moderni. L'utilizzo di soluzioni efficaci e all'avanguardia per la sicurezza è fondamentale per evitare e scoraggiare attacchi informatici atti a compromettere un sistema ed rubare dati sensibili, in quanto questi ultimi possono avvenire a qualsiasi livello di un certo ecosistema (vedi Figura 2.8).
La quasi totalità degli attacchi che interessano i sistemi IoT avvengono a causa della presenza di una rete **botnet** radicata. Una botnet (abbreviazione di robot network) [28] é una rete di computer e/o dispositivi infettata da un malware che é sotto il controllo di un attore malevolo, noto come bot-herder, ed ogni singola macchina sotto il suo controllo é chiamata bot. Da un qualsiasi nodo di questa rete, il bot-herder può, attraverso un software *Command and Control*, ordinare a tutte le macchine di mettere in atto un certo tipo di attacco verso un obiettivo, o catturare il flusso di dati che attraversa qualsiasi dispositivo sotto tale rete.
Le botnet sono considerate un problema molto stringente nel contesto della cybersecurity, in quanto hanno una potenzialità tale da costituire un pericolo anche per governi e aziende con tecniche di protezione avanzate. A tal proposito, analizzeremo nei seguenti punti [47] i vari tipi di attacchi informatici più diffusi e che al giorno d'oggi costituiscono i pericoli più grandi per la sicurezza nel contesto dell'IoT.

- **Attacchi DoS** (**Denial of Service**), che consiste nel "bombardare" un server obiettivo con richieste ridondanti per far sì che i dispositivi IoT vadano in tilt e non possano utilizzare o fornire alcun servizio. Il più noto fra questi é il **DDoS** (Distributed Denial of Service), dove si usano migliaia di protocolli Internet diversi per richiedere servizi, rendendo difficile al server distinguere dispositivi legittimi da dispositivi "attaccanti".

- **Jamming**, un tipo di attacco dove si inviano segnali falsi per interrompere le trasmissioni radio fra i dispositivi IoT, portando ad esaurire l'energia, la larghezza di banda, la potenza computazionale e le risorse di tali dispositivi, a causa dei continui tentativi falliti di connessione.

- **Spoofing**, che costituisce un attacco dove un certo nodo di "spoofing" impersona un dispositivo autorizzato, insieme al suo indirizzo MAC e tag RFID, per accedere illegalmente ad un intero sistema, per poi lanciare un attacco DoS o Man-in-the-middle.
- **Man-in-the-middle**, che consiste nell'invio di segnali di jamming e di spoofing con lo scopo di monitorare, spiare ed alterare segretamente le comunicazione private fra i dispositivi IoT.
- **Attacchi software**, attuati da malware, che sono identificati come qualsiasi codice aggiunto, cambiato o rimosso da un sistema software per causare intenzionalmente errori o sovvertire le normali funzioni del sistema [34]. Fra i vari tipi ci sono Trojan, worm e virus, ransomware, spyware, rootkit, etc., e generalmente hanno lo scopo di operare violazioni finanziarie, intrusioni mirate, leakare dati privati, sovraccaricare una rete, etc.

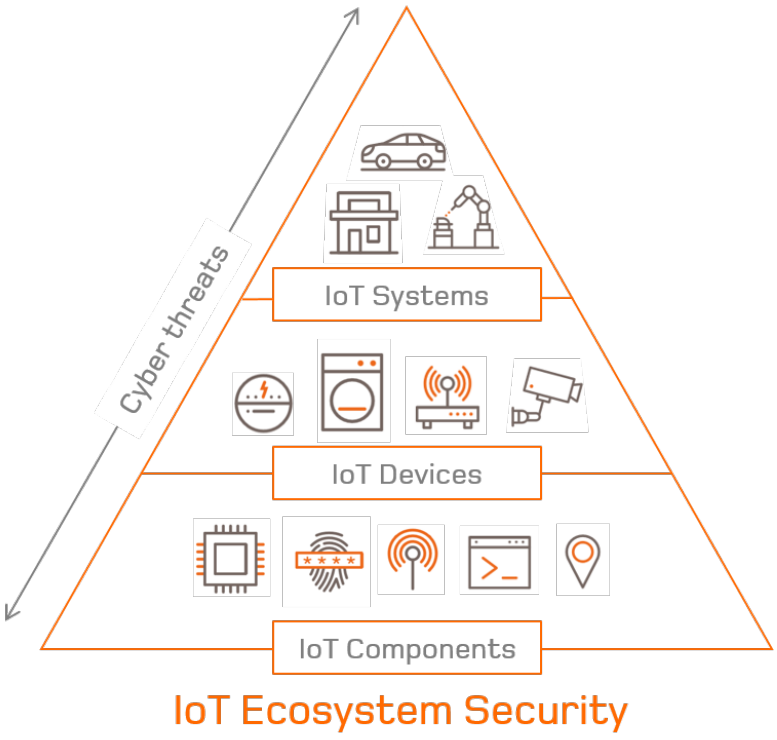

Figura 2.8: I pericoli per la sicurezza negli ecosistemi IoT

Purtroppo, le sopracitate reti botnet evolvono di pari passo con i protocolli di sicurezza adottati dalle aziende produttrici. Essendo questo problema solo parzialmente evitabile, sia produttori che consumatori, vivendo in ambiente dove il lavoro da remoto diventa sempre più una realtà stabile, hanno il dovere di adottare strategie con lo scopo di ridurre la possibilità di essere colpiti da attacchi. Le soluzioni da adottare in merito a questa problematica possono essere le seguenti [8]:

- Aggiornare i propri dispositivi appena possibile, in quanto le vulnerabilità dei dispositivi costituiscono i veri punti di partenza per gli attacchi, e con patch ed aggiornamenti continui e soprattutto qualitativi da parte dei produttori é possibile limitarle
- Utilizzare configurazioni sicure, specie per modem e router che dirigono il traffico, ma anche per i dispositivi più semplici
- Utilizzare password difficili da indovinare, in quanto le password deboli e/o di default possono essere superate in modo estremamente facile dagli hacker per penetrare nei sistemi
- Acquistare ed utilizzare solo dispositivi la cui sicurezza é universalmente riconosciuta, ed evitare di ripiegare su dispositivi più economici che possono costituire un rischio per la nostra privacy

2.3 Stato dell'arte

Per la natura dei sistemi IoT, che come abbiamo visto, sono caratterizzati da un alto grado di interoperabilità e comunicazione, é necessario tutelare la privacy e la sicurezza degli utenti, che come abbiamo visto nella sezione precedente possono essere interessati da diversi tipi di attacchi, come spoofing, DoS e DDoS, jamming, intercettazioni, etc. Nonostante si stia tentando in modo sempre più deciso di tenere il passo quanto a protocolli di sicurezza adottati, sistemi di prevenzione e rafforzamenti ad hardware e software, gli hacker [6], se forniti di abbastanza risorse e tempo, riescono quasi sempre a trovare una falle e debolezze in un sistema, specie se si parla si un sistema IoT, che per la sua natura variegata abbiamo visto celare diverse vulnerabilità. Spesso la tutela di questi sistemi é affidata in mano ad analisti della sicurezza, che talvolta risultano essere poco competenti e al passo coi tempi, e spesso si trovano a dover gestire una mole di dati fuori dalla loro portata. A tal proposito, per garantire una protezione completa e dinamica a questi sistemi ed avere una migliore gestione di questi attacchi, molti esperti di cybersecurity si stanno muovendo verso l'*Intelligenza Artificiale*.
L'Intelligenza Artificiale può essere considerata come un'entità che porta nella sua mente un modello in scala reale del mondo esterno e delle proprie possibili azioni, che sarà in grado di trovare alternative e decidere quale sia la migliore, e di utilizzare l'esperienza delle azioni passate per agire in modo più opportuno e affidabile nel presente e nel futuro. L'applicazione dell'IA é nota come *Machine Learning*, ed é una branca dell'informatica che esplora lo studio e la costruzione di algoritmi che sono in grado di imparare dai dati, e sulla base di essi effettuare decisioni e previsioni.
Sono tre le principali tecniche di ML, **apprendimento supervisionato**, **non supervisionato** e **per rinforzo** tutte ampiamente utilizzate nel contesto della cybersecurity. Vediamole nel dettaglio [13]:

- Le tecniche di **apprendimento supervisionato** sono quelle in cui un certo agente apprende usando dati etichettati, dove le etichette determinano il valore della variabile dipendente, ovvero ciò che si deve decidere/predire. Ai fini dell'addestramento, per ogni entry dei dati in input é noto anche il valore della variabile dipendente, che poi in fase operativa dovrà essere predetta. Gli algoritmi supervisionati comprendono (Figura 2.10) algoritmi di classificazione, per cui va predetta il valore di una variabile categorica, e algoritmi di regressione, per cui va predetto il valore di una variabile numerica.
  Possono essere utilizzate per etichettare il traffico sulla rete e le tracce delle applicazioni dei dispositivi IoT per costruire modelli di classificazione e di regressione. A livello

pratico, i dispositivi IoT possono usare un algoritmo SVM per rilevare intrusioni nella rete, applicare un algoritmo KNN per rilevare la presenza di malware, o persino reti neurali per rilevare intrusioni e attacchi DoS.

- Le tecniche di **apprendimento non supervisionato** rappresentano quelle tecniche in cui un agente apprende usando dati non etichettati, dunque l'agente dovrà essere in grado di imparare senza conoscere il valore della variabile dipendente, ed é dunque intuibile che questo tipo di agenti sono generalmente impiegati per problemi più complessi. A differenza dei precedenti, questo tipo di algoritmi si presta per il clustering (Figura 2.10), ovvero il raggruppamento di oggetti in gruppi che abbiano un certo grado di omogeneità ma che, al tempo stesso, abbiamo un certo grado di eterogeneità rispetto agli altri.
  Possono essere usate per valutare la similarità dei dati per associarli a diversi gruppi, in particolare i dispositivi IoT possono effettuare un analisi di correlazione per rilevare attacchi DoS, oppure applicare degli specifici modelli al livello fisico di una rete per la protezione della privacy.

- Le tecniche di **apprendimento per rinforzo** rappresentano le tecniche dove l'agente compierà una serie di azioni in modo sequenziale, al termine delle quali gli verrà assegnata una ricompensa, con lo scopo di incoraggiare comportamenti corretti e viceversa scoraggiare quelli sbagliati.
  L'impiego di queste tecniche permettono ad un dispositivo IoT di scegliere i protocolli di sicurezza e i parametri chiave per combattere diversi attacchi tramite continui tentativi ed errori. Alcuni esempi sono dati dall'algoritmo Q-learning che permette di migliorare le performance dell'autenticazione e dell'antijamming, e l'algoritmo Dyna-Q per rilevare malware.

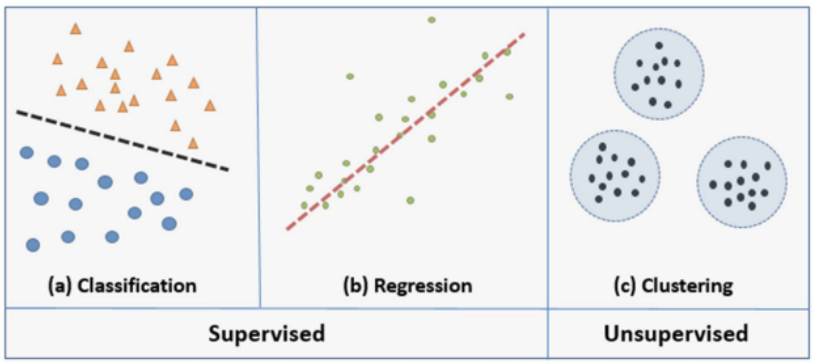

Figura 2.9: Categorizzazione delle tecniche di apprendimento

Nelle seguenti sottosezioni, analizzeremo più nel dettaglio i principi di funzionamento degli algoritmi più impiegati nel contesto della cybersecurity per l'IoT, insieme ai vari casi d'uso degli stessi.

2.3.1 Algoritmi di IA impiegati nel contesto dell'IoT

Ci sono numerosi algoritmi di Intelligenza Artificiale utilizzabili per alzare il grado di sicurezza di un sistema, ognuno dei quali può essere impiegato, plasmato, e utilizzato in combinazione con altri per risolvere uno o più problemi in quest'ambito. Come si mostra nella tabella 2.1, sono molteplici gli algoritmi di Machine Learning utilizzabili per la realizzazione di tecniche di sicurezza utili a contrastare attacchi come DoS, Jamming, Spoofing, intrusioni, malware e intercettazioni. Di seguito, analizzeremo ad alto livello alcuni di questi algoritmi intelligenti, di tipo supervisionato, non supervisionato e per rinforzo, insieme ad alcuni esempi di impiego:

- I **Decision Tree** [22] costituiscono una tecnica supervisionata che crea un insieme di regole sulla base dei dati di addestramento disponibili. In particolare, usa la divisione iterativa per trovare la migliore categorizzazione del traffico che sta analizzando, ed una volta trovato un insieme di regole efficaci, l'IA può analizzare il traffico in tempo reale e lanciare un'eventuale allarme in modo immediato in caso di attività anomala.
  Un caso di utilizzo di quest'algoritmo può essere per la rilevazione di attacchi DoS, tramite l'analisi del flusso dei pacchetti di dati e della loro tipologia, grandezza e permanenza nel traffico. Per dare un'idea, se abbiamo un flusso basso ma una alta durata di un determinato pacchetto nel traffico, allora molto probabilmente siamo sotto attacco, e l'algoritmo farà l'opportuna classificazione (figura 2.9).

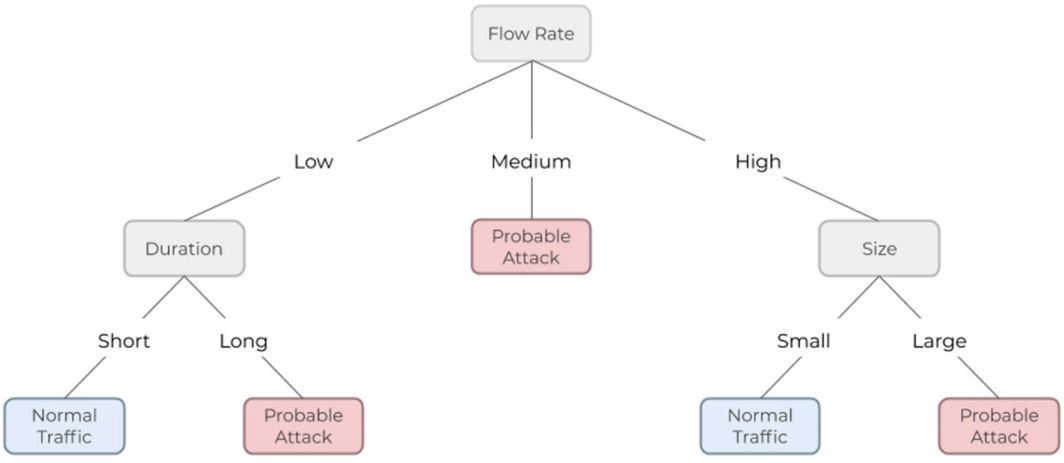

Figura 2.10: Struttura di un decision tree insieme con le scelte effettuate in base allo stato della rete

- La **K-Nearest Neighbours** (**KNN**) [22] é una tecnica supervisionata il cui funzionamento sta nello stimare la probabilità che una predizione sia corretta piuttosto di un'altra sulla base della distanza euclidea della classi prese in considerazione. É considerato un algoritmo 'pigro', in quanto non effettua alcun calcolo o operazione in fase di addestramento, ma si limita ad immagazzinare i dati. In fase di classificazione, effettuerà una valutazione sulla base della distanza euclidea delle k classi più vicine al campione considerato, e l'attribuzione sarà fatta in base alla classe che ha valutazione migliore.
  La KNN é molto utilizzata per i sistemi di intrusion detection e cyberattack detection, in quanto é in grado di imparare rapidamente da nuovi modelli di traffico per rilevare attacchi mai rilevati precedentemente.

- La **Support Vector Machine** (**SVM**) [13] é un modello di apprendimento supervisionato, basato sull'idea di costruire degli iperpiani in grado di separare un set di dati in due classi, e a questo scopo sono appunto utilizzati i vettori di supporto, che rappresentano le entry più vicine all'iperpiano. Sulla base della distanza tra i vettori di supporto delle classi più vicine si traccia l'iperpiano, e sulla base della posizione della nuova entry nel piano avverrà la classificazione. I vettori di supporto varieranno in base al set di dati che si sta analizzando, e nel caso in cui vengano rimossi o modificati, andranno ad alterare la posizione dell'iperpiano, e di conseguenza la classificazione dei dati.
  Nella cybersecurity, questa tecnica è utilizzata per analizzare i pattern del traffico sulla rete e separarli nelle loro classi componenti, come HTTP, FTP, SMTP. Inoltre, in applicazioni in cui gli attacchi possono essere simulati, si utilizza il traffico di rete generato dai test di penetrazione come dati di allenamento.

- Le **Reti Neurali** [22] sono reti di neuroni virtuali, che si ispirano alla modalità con cui i neuroni interagiscono fra di loro nel cervello umano per comunicare e interpretare informazioni. A livello pratico, in una rete neurale un neurone è un'equazione matematica che legge dati in input e restituisce un valore obiettivo in output, il quale é passato ai vari neuroni in relazione al valore di quest'ultimo. Gli algoritmi basati su reti neurali iterano finché il valore di output non é accettabilmente vicino al valore obiettivo, il che permette ai neuroni di imparare e correggere i propri pesi misurando l'errore fra il valore atteso e il valore dato in output precedentemente. Quando questo processo termina, l'algoritmo fornisce un'equazione matematica che é in grado di dare in output valori utilizzabili per classificare dati.
  Le reti neurali presentano il vantaggio di essere incredibilmente flessibili quando si

trovano di fronte a nuove informazioni, anche nei confronti di attacchi mai affrontati in precedenza, a differenza degli altri modelli di apprendimento che possono diventare obsoleti in presenza di nuovi tipi di traffico o tipi di attacchi. Proprio per questo, si prestano in maniera ottima per la creazione di sistemi di Intrusion Detection.

- L'algoritmo di clustering **k-means** (k-medie) [39] è un algoritmo di apprendimento non supervisionato, il cui scopo è di individuare k gruppi a partire dai dati di addestramento. In fase operativa, l'algoritmo lavorerà in modo iterativo per assegnare ogni data point ad uno dei k gruppi sulla base delle features fornite. Come risultato, il k-means troverà un centroide (un record 'rappresentativo') per ognuno dei k cluster, i quali potranno essere utilizzati per categorizzare i dati.
  L'algoritmo k-means [35] è fra gli algoritmi non supervisionati più utilizzati nelle analisi di sicurezza, in quanto è in grado di dividere i dati di sicurezza in gruppi di entità simili, il che aiuta ad ottenere importanti informazioni su pattern sia conosciuti che sconosciuti.

- Il **Q-Learning** [36] è un algoritmo di apprendimento per rinforzo model-free, off-policy e values-based, che sceglie un azione in base a quanto qualitativa sarà quell'azione in rapporto alla ricompensa che riceverà la stessa. In particolare, è off-policy e di conseguenza model-free, in quanto non utilizza alcuna funzione di transizione/di ricompensa per stimare la politica operativa ottimale (difatti non esiste), ed è values-based in quanto aggiorna la funzione di valutazione utilizzando un equazione, in particolare l'equazione di Bellman. A livello pratico, questo algoritmo effettua una decisione in relazione alla risposta che ci aspetta di ottenere dall'ambiente, ed invece di utilizzare un sistema di ricompense per il miglioramento, utilizza una politica di trial and error.
  Nel paper [3], si mostra come su un sistema CPS (Cyber-Physical System), che comprende anche sistemi IoT, venga impiegato un algoritmo Q-Learning per risolvere delle falle della sicurezza scoperte in seguito ad attacchi DDoS e Jamming.

Tipi di attacchi	Tecniche di sicurezza	Algoritmi di ML	Performance
DoS	IoT offloading, Controllo degli accessi	Reti neurali, Analisi di correlazione multivariata	Detection accuracy e errore quadratico medio
Jamming	IoT offloading	Q-learning, DQN	Consumo energetico SINR
Spoofing	Autenticazione	Q-learning, Dyna-Q, SVM, DNN, dFW	Tasso di errore medio, Detection accuracy, Classification accuracy, Tasso di falsi allarmi, Tasso di misdetection
Intrusion	Access control	SVM, Naive Bayes, KNN, NN	Classification accuracy, Tasso di falsi allarmi, Tasso di rilevamento, Errore quadrato medio
Malware	Malware detection, Controllo degli accessi	Q/Dyna-Q/PDS, Random forest, KNN	Classification accuracy, Tasso di falsi allarmi, Detection accuracy , Detection latency
Intercettazioni	Autenticazione	Q-Learning, Non-parametric Bayesian	Proximity data rate, Secrecy data rate

Tabella 2.1: Tecniche di sicurezza impiegate per l'IoT basate sul ML [47]

2.3.2 Sistemi di sicurezza intelligenti

Le tecniche di Machine Learning come apprendimento supervisionato, non supervisionato e per rinforzo sono dunque sempre più utilizzate per migliorare la sicurezza, in particolare per operazioni quali autenticazione, controllo degli accessi, antijamming, e rilevazione di malware, come abbiamo visto nella tabella precedente. Di seguito saranno descritti diversi sistemi di sicurezza normalmente impiegati per proteggere i dispositivi che utilizziamo tutti i giorni, insieme a degli esempi su come l'utilizzo di tecniche di Machine Learning possa supportare e potenziare gli stessi.

Malware detection

Il **malware detection** é il processo di analisi di un computer o dispositivo per rilevare la presenza di malware. Un **Malware Detection System** (**MDS**) é un sistema software che é in grado di analizzare i programmi e le applicazioni presenti su uno o più dispositivi, per determinare se quella applicazione é benigna o un malware. Questi sistemi generalmente hanno una componente detector, che é responsabile di determinare l'identità dell'applicazione in analisi.
Ci sono diversi tipi di MDS, che variano in base all'approccio di rilevamento [4]:

- **Signature-Based Malware Detection**, che si basa sull'incapsulamento della struttura di un programma per identificare il malware, ed é l'approccio generalmente utilizzato con gli antivirus commerciali. Costituisce una tecnica efficiente e veloce per il rilevamento di malware noti, ma ha gravi lacune per quanto riguarda malware nuovi al software, e anche per l'utilizzo da parte dei malware di tecniche di offuscamento che rendono obsoleto lo stesso.

- **Behavior-Based Malware Detection**, il cui funzionamento si basa sull'osservazione del comportamento dei programmi con appositi tool per determinare se il programma in esame é benigno o é un malware. Questi software si basano sul principio per cui anche se il codice di un programma cambia il suo comportamento rimane simile, ed in questo modo anche i nuovi malware possono essere rilevati efficacemente. Ciò nonostante, ci sono situazioni in cui i malware non funzionano in modo atteso sotto certe situazioni, portando il software di detection a classificazioni e segnalazioni errate.

- **Heuristic-Based Malware Detection**, che presentano approcci di rilevamento molto più complessi dei precedenti, in quanto utilizzano diverse tecniche di Machine Learning.

Questo tipo di software presenta tassi di rilevamento molto alti ed ha la capacita di rilevare malware sconosciuti, ma presenta lo svantaggio di non comportarsi bene nei confronti di malware molto complessi.

- **Deep Learning-Based Malware Detection**, che sfrutta tecniche di Deep Learning con le quali é possibile creare sistemi completamente automatici, con un alto tasso di rilevamento ed una capacità di rilevare malware sconosciuti difficilmente eguagliabile da altri sistemi. Di contro, l'addestramento di un tale modello risulta molto complesso e difficile, soprattutto in merito al basso numero di dati a disposizione.

Intrusion detection

Un **Intrusion Detection System** (**IDS**) è un sistema hardware o software, utilizzato per identificare accessi non autorizzati, violazioni ed attività malevoli a computer o reti, le quali sono tipicamente segnalate e schedate utilizzando un sistema di gestione centralizzato.
Ci sono varie declinazioni di Intrusion Detection System, che variano in base all'approccio utilizzato per il rilevamento delle intrusioni [23]:

- Un **Network Intrusion Detection System** é rilasciato in uno o più punti strategici sulla rete, in modo che possa monitorare il traffico in entrata e in uscita verso tutti i dispositivi della rete, ed effettuare eventuali segnalazioni.

- Un **Host Intrusion Detection System** é rilasciato su tutti i computer e dispositivi che abbiano un accesso diretto alla rete da proteggere. Rispetto al precedente, ha il vantaggio di essere in grado di rilevare pacchetti di rete anomali che provengano da un qualsiasi dispositivo della rete stessa, o addirittura dal dispositivo in questione.

- Un **Signature-based Intrusion Detection System** monitora tutti i pacchetti che attraversano la rete, e li confronta con un database di firme di attacchi e attributi di malware noti, in maniera molto simile ad un software antivirus, per poi effettuare l'eventuale segnalazione.

- Un **Anomaly-based Intrusion Detection System** monitora il traffico sulla rete e lo confronta con una baseline per determinare se quel tipo di traffico é normale rispetto a fattori quali larghezza di banda, protocolli, porte, e altri dispositivi.
  Questa particolare declinazione di IDS spesso sfrutta il Machine Learning per stabilire la baseline, che poi allerterà chi di dovere nel caso di attività sospette e violazioni della policy. Con questo approccio, si superano ampiamente i limiti degli IDS visti in

precedenza, specie per il rilevamento di attacchi e malware mai incontrati prima, anche se i costi di realizzazioni sono molto elevati.

Per rilevare quando un certo attacco è attuato da entità non autorizzate, un IDS si serve delle seguenti funzionalità:

- monitora le operazioni di router, firewall, server di gestione di chiavi, e file che sono necessari ad altri controlli di sicurezza
- fornisce agli amministratori un modo di gestire, organizzare e interpretare processi di verifica e log, che sarebbero altrimenti difficili da ottenere e analizzare
- fornisce un'interfaccia user-friendly, in modo che utenti non esperti possano essere di supporto per la gestione del sistema di sicurezza
- include un database completo delle firme dei vari attacchi, da utilizzare per il confronto con lo stato attuale del sistema
- riconosce e segnala quando un IDS rileva i file che sono stati alterati
- genera un allarme e notifica nel caso in cui ci sia stata una breccia nella sicurezza
- reagisce agli intrusi, bloccandoli o bloccando il server stesso

Dato che un IDS generalmente si limita a segnalare attività sospette e schedarle, vengono spesso affiancati da un *Intrusion Prevention System*, che è in grado di intercettare i pacchetti di rete che possono potenzialmente danneggiare la stessa, bloccare indirizzi IP, chiudere l'accesso a risorse critiche. In tal caso, si parla di Intrusion Detection System *attivi* (quelli visti finora sono passivi).

Network Monitoring e Traffic Filtering

Il **Network Monitoring** [2] si riferisce ad un insieme di tecniche utili a monitorare il traffico della rete ad un determinato livello di granularità (generalmente a livello di pacchetti), con lo scopo di ottenere informazioni relative alle operazioni e alle performance della rete, insieme ad informazioni sul comportamento degli utenti che ne usufruiscono. Per l'implementazione di sistemi di questo tipo, viene preferito l'uso del Deep Learning, specie per il tradeoff fra complessità e performance, che risulta molto meno critico rispetto ai sistemi precedentemente analizzati. Il Network Monitoring ovviamente è fine a stesso se operato senza attribuire alcuna reazione a situazioni anomale, e a tal proposito a queste tecniche viene solitamente affiancato

un sistema di Traffic Filtering

Il **Traffic Filtering** [50] è effettuato dalle componenti forwarder e middleboxers che fanno parte dell'infrastruttura di una rete, ed è realizzato facendo sì che il sistema di routing diriga il traffico verso un filtro, così che esso possa esaminarlo per rilevare e prevenire attacchi di flooding, jamming, etc. In questo caso, i filtri attuano un'azione difensiva, generalmente limitandosi ad interrompere il traffico. Anche per questa tecnica di sicurezza sono presenti diverse declinazioni:

- Il **Content-based Traffic Filtering** è caratterizzato da dei cosiddetti *criteri di filtering*, con i quali si identifica il traffico anomalo o sospetto, e che sono utilizzati per esaminare header e campi dei pacchetti che viaggiano sulla rete per rilevare anomalie e violazioni di privacy e attacchi.
- Il **Path-based Traffic Filtering** utilizza sempre i criteri di filtering, ma li migliora dinamicamente analizzando il percorso effettuato da ogni pacchetto sulla rete. Questo porta un miglioramento alla precisione con cui si rilevano pericoli e anomalie, insieme con un filtraggio più efficace, in quanto vengono bloccati pacchetti pericolosi che provengono dalla stessa fonte o che effettuano lo stesso percorso.

CAPITOLO 3

Progettazione ed implementazione dell'IA per l'Intrusion Detection

Nel seguente capitolo, sono descritte tutte le tecniche di validazione, normalizzazione, preprocessing e classificazione che sono state considerate e/o implementate per lo sviluppo ed il perfezionamento del modello di intelligenza artificiale, insieme con tutte le scelte implementative effettuate e le problematiche via via riscontrate e risolte. Per tutta la durata del lavoro di progettazione ed implementazione, é stato seguito l'approccio ingegneristico CRISP-DM con le sue varie fasi, con lo scopo di ottenere un software con un livello di qualità e affidabilità il più alto possibile.

3.1 Modello CRISP-DM

L'obiettivo principale di questo studio é quello di fornire un modello che sia affidabile, versatile, estendibile, e soprattutto che sia addestrato sulla base di dati veritieri e autentici. Con l'obiettivo di rispettare questi standard di qualità, é stata ritenuta opportuna l'adozione del modello **CRISP-DM** [9], un approccio ingegneristico dalla validità universalmente riconosciuta, molto utile per gestire il ciclo di vita di progetti basati su intelligenza artificiale. Le fasi previste da questo modello sono *Business Understanding*, *Data Understanding*, *Data Preparation*, *Data Modeling*, *Evaluation* e *Deployement*, le quali possono essere eseguite ciclicamente un numero illimitato di volte fino a raggiungere il risultato desiderato.
Come si può vedere dal diagramma sottostante, la sequenza delle fasi non deve essere necessariamente seguita, ed é permesso lo spostamento avanti e indietro fra una fase e l'altra con lo scopo di perfezionare il lavoro svolto.

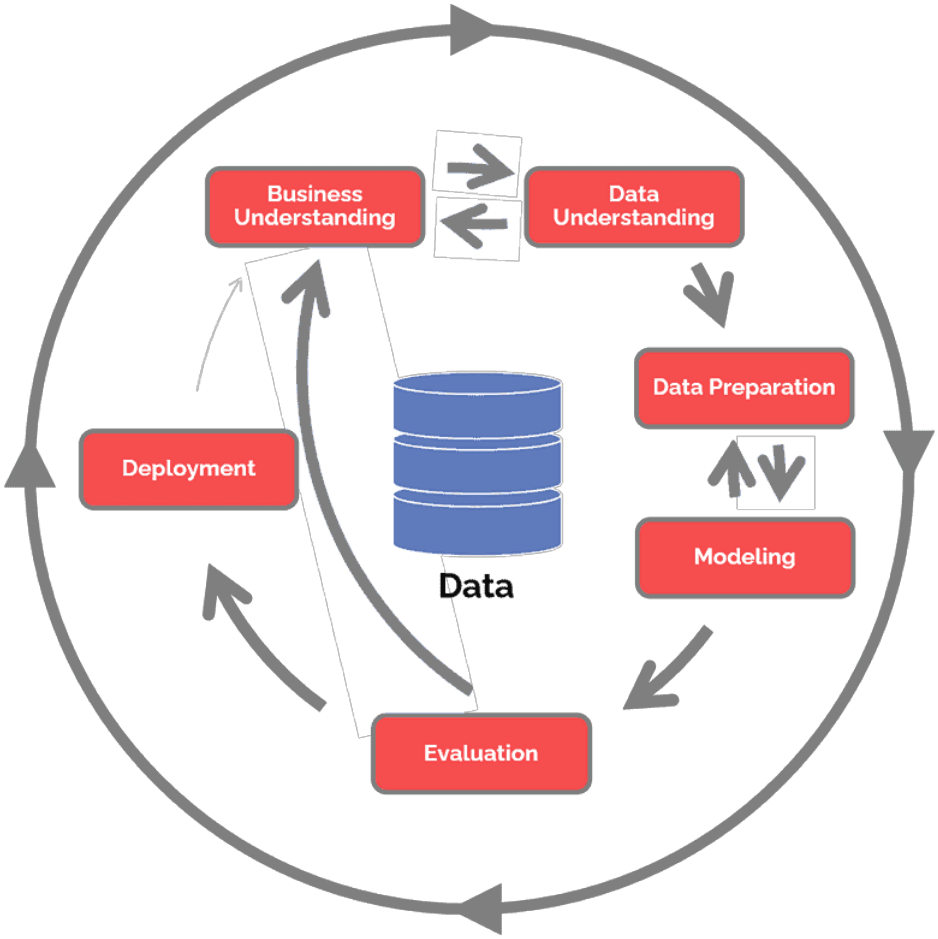

Figura 3.1: Diagramma che mostra l'ordine e le dipendenze fra le fasi del modello CRISP-DM

3.2 Business Understanding e Data Understanding

La prima fase del CRISP-DM, il **Business Understanding**, prevede la raccolta e la definizione degli obiettivi che si intende raggiungere con il proprio software, che prendono il nome di *obiettivi di business*, e che rappresentano i criteri in base ai quali possiamo accertarci che il sistema sia in linea con i risultati prefissati. Gli obiettivi di business stabiliti, a partire dai quali sarà avviata la progettazione e l'implementazione sono i seguenti:

- i dati su cui l'agente intelligente sarà addestrato dovranno essere dati reali e autentici, ottenuti da un'autentica presenza di malware, per ottenere un modello affidabile in casi d'uso reali
- inoltre, l'agente sarà addestrato con dati relativi ad almeno 3 diversi dispositivi, anch'essi reali, per avere un modello quanto più flessibile possibile

- considerando le metriche più diffuse, quali accuracy, precision, recall e F1 score, si deve ottenere una media dei valori non inferiore al 90%
- a prescindere dalla combinazione ottima di algoritmi di normalizzazione, preprocessing e classificazione, saranno implementati e messi a disposizione diversi algoritmi per tipologia, in modo da fornire diverse tecniche fruibili e testabili all'utilizzatore
- il sistema metterà a disposizione diversi tipi di grafici, come matrici di confusione e matrici di correlazione, per poter effettuare ulteriori valutazioni in modo rapido ed intuitivo

Oltre ai requisiti del modello, il dataset scelto avrebbe dovuto rispettare i seguenti requisiti per quanto riguarda i dati per l'addestramento:

- avere formato .csv ed essere di classificazione
- possedere informazioni relative a diversi dispositivi IoT
- possedere informazioni relative a diverse reti botnet e/o diversi malware
- possedere informazioni che siano facilmente ottenibili a partire dall'analisi di un qualsiasi porzione del traffico di rete di un sistema e/o di un dispositivo
- tutte le features devono avere formato numerico (intero, decimale, ecc.)

Prima di passare alla fase successiva, é stato necessario anche definire le tecnologie ed i tool da utilizzare. In tal senso, é stato scelto il linguaggio **Python**, il quale oggigiorno é in assoluto il più consistente ed immediato per l'analisi di dati a basso ed alto livello, la costruzione di modelli di intelligenza artificiale, l'analisi della bontà degli stessi, oltre ad avere la disponibilità di un gran numero di librerie orientate al machine learning e alla produzione di grafici.

La fase di **Data Understanding** prevede l'identificazione e l'analisi del dataset che, sulla base dei requisiti definiti, risulta essere il più indicato per il raggiungimento degli obiettivi, oltre all'effettuazione di un'esplorazione sui data atta ad identificare caratteristiche e relazioni fra i dati selezionati.
Va detto che buona parte dei dataset relativi a questo dominio di interesse sono ottenuti simulando la presenza di malware e reti botnet, ed essendo impensabile la realizzazione di un modello di Intrusion Detection solido a partire da dati non realistici e dunque poco attendibili, i dataset considerati di seguito sono stati frutto di test eseguiti a partire da una presenza di malware autentica, in accordo con uno dei requisiti prima definiti.
Una delle sfide maggiori nello sviluppo di questo modello è stata riscontrata dal principio, ovvero nell'individuazione dei dati di partenza per l'addestramento del modello, che in ambito IoT risultano essere davvero scarsi in letteratura. Dopo un'accurata ricerca su noti siti web, quali Kaggle, Google Dataset Search, e Harvard Dataverse, sono stati quattro i dataset pertinenti con i requisiti definiti, che sono descritti di seguito nei loro punti di forza e debolezza:

- Il primo [27] è costituito da oltre 7 milioni di data points e 115 feature in formato numerico, e contiene dati relativi a 9 diversi dispositivi di note aziende, colpiti da diverse tipologie di attacchi, dunque un dataset sicuramente completo ma generalmente complesso da gestire, manipolare e studiare

- Il secondo [12] ha circa 2 milioni di datapoints, ma presenta un elevato sbilanciamento ed overfitting, oltre a contenere dati relativi a dispositivi e tipologie di attacchi poco pertinenti con l'IoT

- Il terzo [33] contiene informazioni relative a circa 900 traffici di dati su diversi dispositivi ottenuti tramite l'API *pcap* e non processati, dunque i dati in formato grezzo avrebbero necessitato di un refactoring importante e inopportuno in base ai requisiti stabiliti

- Il quarto [49] ha una dimensione ancora maggiore del primo, dunque una complessità di manipolazione ancora a maggiore, oltre al fatto che tutti i dati presenti sono stati ottenuti sotto la stessa rete di dispositivi, e che dunque non avrebbero portato alla realizzazione di un modello eterogeneo ed universalmente attendibile

In conclusione, ho deciso di scegliere il primo dataset, che sarà identificato sotto il nome di '**N-BaIoT**'. Sebbene la complessità di gestione era l'unico che, in accordo coi requisiti definiti, avrebbe permesso la realizzazione di un modello efficace contro i più noti tipi di attacchi e che avrebbe garantito una protezione più completa possibile, data la presenza una grande mole di dati, di dispositivi con caratteristiche diverse fra loro, e di malware appartenenti a diverse categorie.

3.2.1 Analisi dei dati

Prima di proseguire con l'implementazione preliminare del modello, considerando la mole dei dati a disposizione, é stato necessario analizzare gli stessi, insieme alle relazioni fra loro presenti e il significato di ogni feature.
Il dataset, in formato .csv, contiene informazioni relative ad un traffico di dati reale, ottenuto da 9 diversi dispositivi IoT di natura commerciale, infettati in modo autentico da malware della famiglia Mirai e BASHLITE. Il dataset [27] conta 7062606 record, ad ognuno dei quali sono associate 115 features. É presente una suddivisione in 9 sottogruppi che corrispondono ai 9 dispositivi esaminati, e per ognuno di essi é presente un ulteriore partizionamento effettuato in relazione all'attacco a cui sono soggetti. Oltre ai dati sul traffico benigno, sono presenti dati relativi al traffico anomalo, che possono essere divisi in 10 sottogruppi relativi rispettivamente a 10 attacchi portati avanti da reti botnet della famiglia Mirai e BASHLITE, dunque si presta bene per una classificazione multi-classe, per un totale di 10 classi di attacchi più una classe di traffico normale (tali classi sono suddivise sottoforma di file .csv). Una descrizione più approfondita dei dispositivi oggetto di studio é offerta dalla seguente tabella:

Brand	Dispositivo	Dotazioni extra	Criticità
Danmini	Campanello	Telecamera, Microfono, Altoparlante, Apertura porte in remoto	Intrusioni indesiderate in proprietà private
Ecobee	Termostato	Altoparlanti e Microfono, integrazione con servizi Apple HomeKit e Amazon Alexa	Manomissioni di temperatura, Furto di dati sensibili a partire da app collegate
Ennio	Campanello	Monitor, Telecamera, Microfono, Altoparlante, Apertura porte in remoto	Intrusioni indesiderate in proprietà private
Philips	Baby Monitor	Altoparlante, Microfono, Telecamera infrarossi	Invasione della privacy
Provision	Telecamera di sicurezza (Modello PT-737E)	Altoparlante, Microfono, Telecamera infrarossi, Sensore di movimento	Invasione della privacy, Compromissione della sicurezza
Provision	Telecamera di sicurezza (Modello PT-838)	Altoparlante, Microfono, Telecamera infrarossi, Sensore di movimento	Invasione della privacy, Compromissione della sicurezza
Samsung	Webcam	Altoparlante, Microfono, LED microfono	Invasione della privacy, Ascolto di conversazioni private
SimpleHome	Telecamera di sicurezza (Modello XCS7-1002-WHT)	Microfono, LED di funzionamento	Invasione della privacy, Compromissione della sicurezza
SimpleHome	Telecamera di sicurezza (Modello XCS7-1003-WHT)	Microfono, LED di funzionamento	Invasione della privacy, Compromissione della sicurezza

Tabella 3.1: Dispositivi utilizzati nel traffico di dati oggetto di studio

Nelle due tabelle che seguono sono elencate le features dal dataset insieme ad attributi specifici del flusso di dati, ognuno dei quali è associato ad ognuna delle features presenti:

Feature	**Descrizione**
MI_dir	Statistica che riassume il traffico recente dalla sorgente MAC-IP
H	Statistica che riassume il traffico recente dall'host di questo pacchetto (IP)
HH	Statistica che riassume il traffico recente dall'host di questo pacchetto (IP) all'host di destinazione del pacchetto
HpHp	Statistica che riassume il traffico recente che va dall'indirizzo host+porta (IP) di questo pacchetto all'indirizzo host+porta di destinazione del pacchetto
HH_jit	Statistica che riassume il jitter del traffico che va dall'host di questo pacchetto (IP) all'host di destinazione del pacchetto

Tabella 3.2: Features presenti nel dataset

Attributo	**Descrizione**
weight	Peso del flusso di dati (considerando l'intervallo temporale)
mean	Media del flusso di dati
std	Deviazione standard del flusso di dati
radius	La somma del valore delle radici quadrate della varianza dei due flussi
magnitude	La somma del valore delle radici quadrate della media dei due flussi
covariance	Valore di covarianza approssimativo dei due flussi
pcc	Valore del coefficiente di correlazione di Pearson dei due flussi

Tabella 3.3: Metriche di valutazione associate al flusso di dati

In pratica, tutte le features presenti sono ottenute dalla combinazione di tutti gli attributi presenti con tutte le statistiche presenti nelle due tabelle sopra. Inoltre, per avere un idea delle varie finestre temporali all'interno delle quali è misurato il traffico di dati, per ognuna delle features combinate sono presenti 5 ulteriori campi a cui sono associati diversi valori di un coefficiente **lambda** (**L5**, **L3**, **L1**, **L0.1**, **L0.01**). Il valore associato a tali coefficienti rappresenta l'intervallo di tempo in secondi all'interno dei quali é stata analizzata la rete, e in questo modo é possibile confrontare le condizioni del traffico su diverse finestre temporali, garantendo quindi un'analisi più accurata ed estesa dei dati. Dunque, la combinazione di statistiche, attributi e intervalli temporali permette di usufruire di 115 features distinte.

3.2.2 Data Exploration

Per un migliore orientamento nelle scelte degli algoritmi e delle tecniche da utilizzare nella successiva fase di Data Preparation, é stato ritenuto necessario effettuare Data Exploration sul dataset. A tal proposito ho utilizzato *seaborn*, una libreria per la visualizzazione di dati ad alto livello, che permette la generazione di grafici informativi molto intuitivi. Ciò di cui si necessitava erano grafici che permettessero l'interpretazione del grado correlazione fra le features del dataset, sia ad alto livello, sia fra specifiche feature.
A tal proposito, tramite seaborn é stata generata una **matrice di correlazione**, ovvero una tabella che illustra i coefficienti di correlazione tra ogni possibile coppia di feature. Tali coefficienti variano da -1 a 1, e in particolare:

- le caselle con colori tendenti al rosso (coefficienti vicini allo 0) indicano una scarsa correlazione fra le feature
- le caselle con colori tendenti al nero (coefficienti vicini al -1) indicano una correlazione lineare negativa, ovvero al crescere del valore di una feature, decresce il valore dell'altra feature corrispondente
- le caselle con colori tendenti al bianco (coefficienti vicini all'1) indicano una correlazione lineare positiva, ovvero al crescere del valore di una feature cresce analogamente anche il valore dell'altra feature

Per avere un ulteriore base a partire dalla quale valutare le relazioni fra features, ho generato sempre tramite seaborn un **grafico a dispersione**, ovvero un grafico in cui viene riportato l'andamento di due features di un set di dati su un piano cartesiano. I dati risultanti sono riportati sottoforma di una collezione di punti che rappresenta i record del dataset, ciascuno con posizione sull'asse orizzontale determinato da una feature e sull'asse verticale determinato dall'altra. Per quanto riguarda l'interpretazione di questi grafici, più i punti tenderanno a creare una linea dritta, più sarà forte la relazione fra le features. Inoltre, se i punti tenderanno ad andare dall'origine verso l'alto, si avrà una correlazione positiva, viceversa si avrà una correlazione negativa.
A causa del limite delle risorse computazionali e dell'inefficienza della libreria a produrre questo grafico, non é stato possibile generare grafici a dispersione per tutte le coppie di variabili, dunque sono state generati grafici solo a partire dalle features a cui sono associati coefficiente lambda L5 e metrica 'weight', ritenute le più esplicative.
Sulla base dei criteri sopra definiti, la produzione delle tabelle ha dato i seguenti risultati:

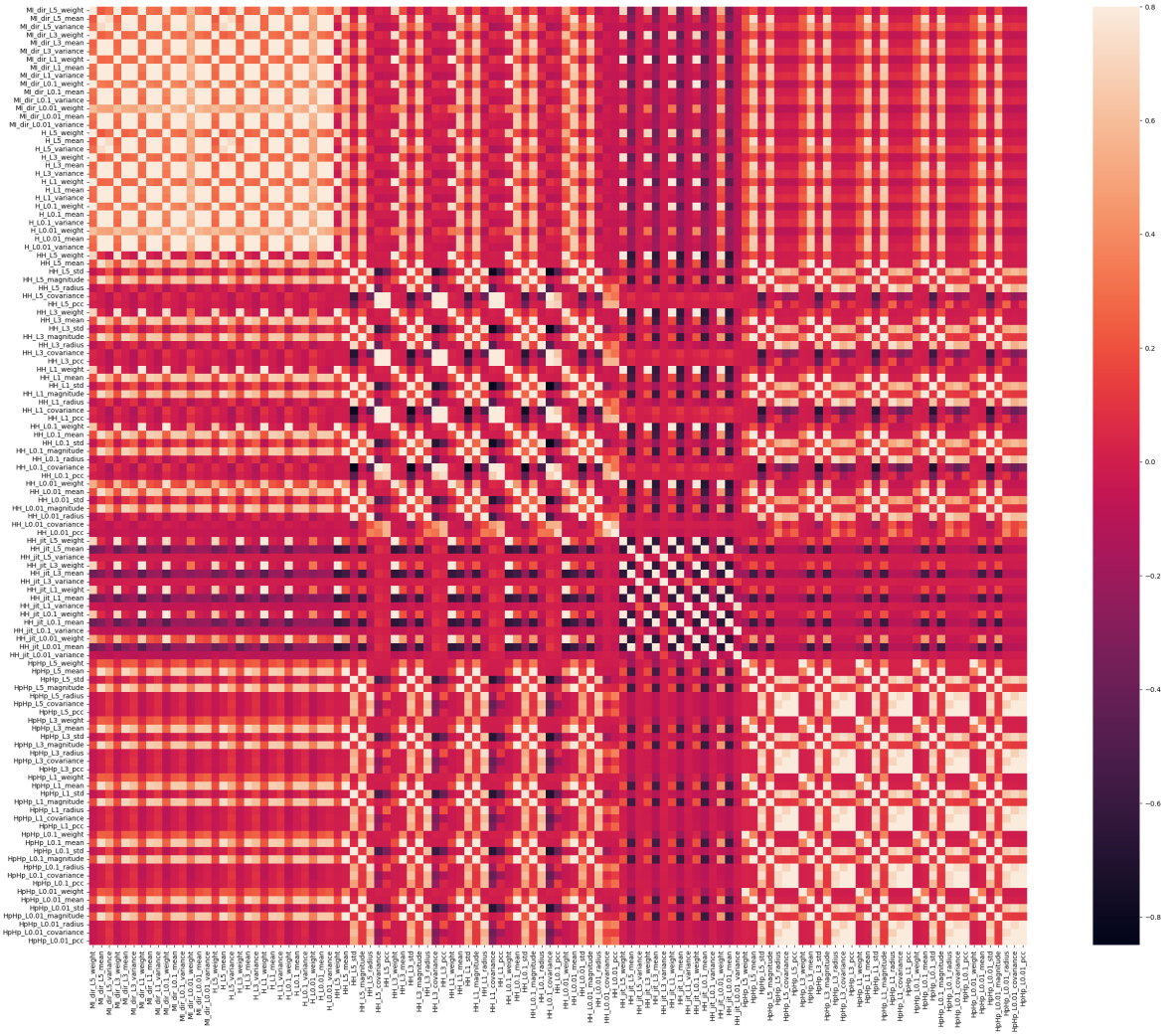

Figura 3.2: Matrice di correlazione generata a partire dai dati del dispositivo Philips

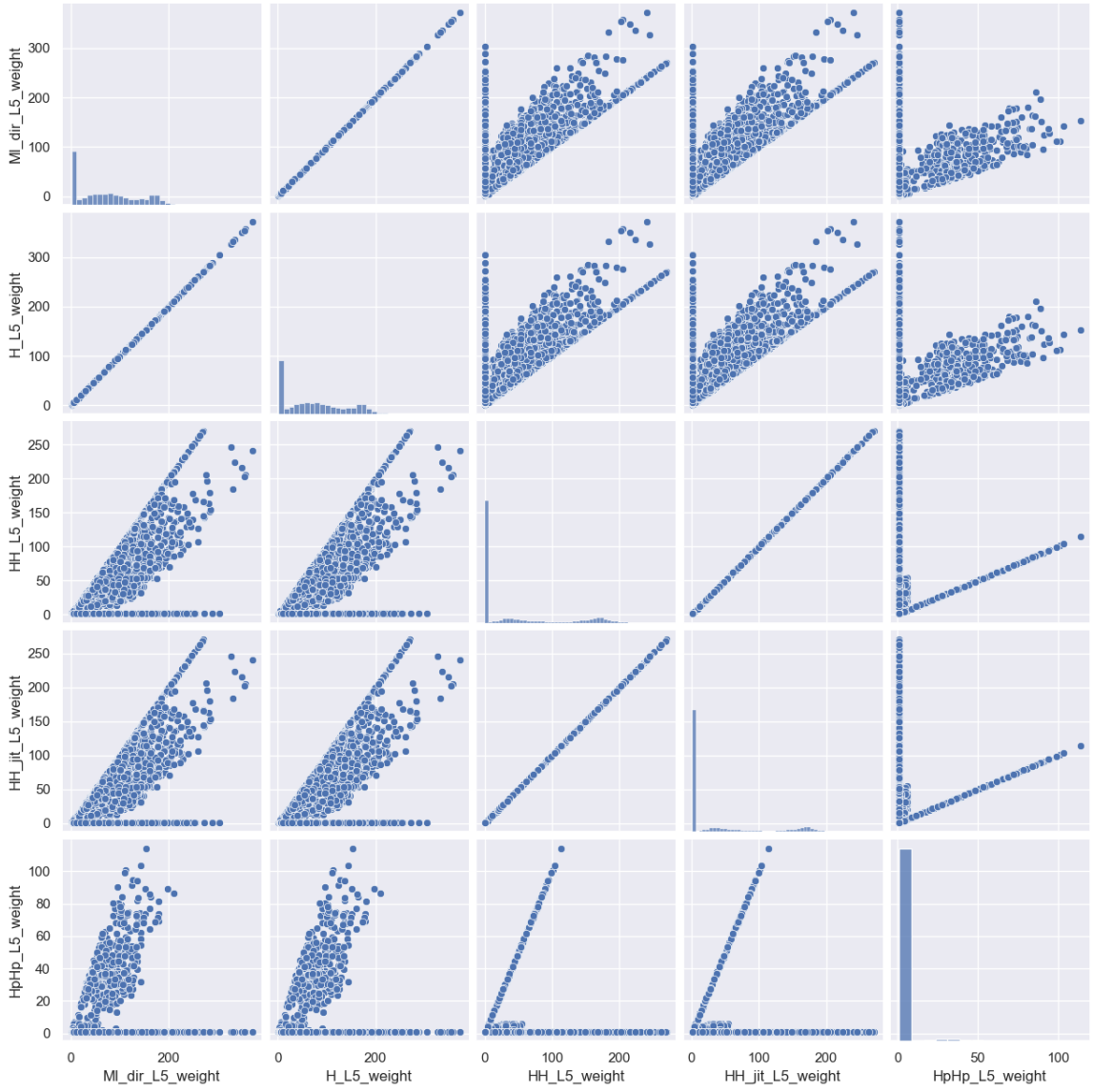

Figura 3.3: Grafico a dispersione generato a partire da un sottoinsieme di features dal dispositivo Philips

Come é possibile dedurre, l'andamento dei grafici suggerisce una relazione lineare positiva fra tutte le coppie di variabili considerate nel grafico a dispersione, dunque al crescere di una, cresce anche l'altra. Unendo questo dato con l'analisi della matrice di correlazione, é stato possibile fare le seguenti considerazioni sul dataset:

- le poche relazioni lineari negative presenti nella matrice di correlazione (le caselle nere) corrispondono ad osservazioni effettuate all'interno di finestre temporali molto diverse fra loro (coefficienti lambda L5 e L0.1), fattore che giustifica le tendenza negativa, quindi possono essere non considerate
- in accordo con i grafici a dispersione, la maggioranza delle osservazioni del traffico relative a finestre temporali uguali o simili fra loro hanno una correlazione lineare positiva (le caselle rosse)
- le variabili scarsamente correlate (le caselle bianche) si trovano in corrispondenza di metriche concettualmente slegate fra loro

In conclusione, é possibile affermare che il dataset ha un alto grado di *multicollinearitá*, che é un fenomeno che si verifica quando due o più variabili indipendenti (nel nostro caso, quasi la totalità di esse) sono correlate fra loro in modo forte, ma non assoluto (ovvero il coefficiente di correlazione é molto vicino all'1), dunque siamo in presenza di un dataset normalmente distribuito, informazione molto importante per la successiva fase di normalizzazione.
Il fenomeno della multicollinearitá costituisce generalmente un problema solo in casi di regressione, mentre nel nostro caso potrebbe solo portare ad un inefficienza a livello temporale nella fase di addestramento e la predizione del nostro modello. Nelle seguenti sottosezioni, saranno descritti gli approcci utilizzati e adottati in seguito a queste valutazioni.

3.3 Data Preparation

In questa sezione saranno discusse le operazioni e le scelte tecniche e progettuali fatte per la fase di **Data Preparation**, il cui scopo prevede appunto la preparazione dei dati affinché siano pronti per essere dati in input al modello per l'addestramento, tramite l'applicazione di processi di pulizia dei dati e algoritmi per la riduzione di dimensionalità, feature scaling e preprocessing.
In particolare, sarà prima esaminata l'implementazione già pronta di un utente sul dataset scelto, e a partire dai risultati osservati saranno mossi i primi passi della nostra implementazione, iniziando già ad avere un'idea su cosa può essere migliorato e/o esteso. Dunque verrà descritta a 360 gradi la pipeline del modello costruito, insieme a tutte le valutazioni via via effettuate e le decisioni implementative prese.

3.3.1 Implementazione iniziale

Una volta analizzati i dati ad alto livello, a partire dal dataset in esame é stata cercata una implementazione già pronta e di buona qualità per iniziare ad avere un'idea di come approcciare il problema, ottenere informazioni sui dati utilizzati, ed avere una base di partenza più solida da cui avviare l'implementazione. Molto utile in tal senso é stata l'implementazione dell'utente 'Stefanos T' [41], che a partire da una piccola partizione del dataset, in particolare da un campione del 20% dei dati relativo al solo dispositivo Provision PT-737E, ha costruito una semplice rete neurale tramite l'ausilio di Python e di apposite librerie, che saranno meglio descritte in seguito.
In particolare, questo utente ha effettuato una normalizzazione con approccio z-score, senza l'utilizzo di alcuna altra tecnica di preprocessing. La validazione é stata approcciata con uno split 75/25 (75% training, 25% testing), ottenendo un accuracy circa dell'85%. Purtroppo questo valore é in grado di dirci ben poco sulla bontà effettiva del dataset, in quanto sono stati considerati circa 50.000 record relativi ad un solo dispositivo su un totale di oltre 7 milioni di record relativi a 9 dispositivi, ma considerando la buona qualità dell'implementazione della rete neurale, ho ritenuto opportuno prendere la stessa come base di partenza per il prosieguo del lavoro.
Per tener traccia del lavoro svolto da qui a questa parte ho creato una repository Github, in modo da tener traccia delle varie fasi dello sviluppo del software nel tempo. Innanzitutto, per rendere possibili il confronto e rendere più atomici i risultati ottenuti dal modello, ogni run effettuerà la classificazione selettivamente per il dispositivo selezionato, dunque saranno

necessarie 9 run per ottenere informazioni su tutti i dispositivi e di conseguenza informazioni sulla bontà generale del modello. Per la lettura dei file .csv e le successive operazioni é stata utilizzata la libreria *pandas*, che fornisce un insieme di tool utili per l'analisi e la manipolazione dei dati. Con l'ausilio di questa libreria gli 11 file .csv (1 traffico benigno + 10 traffico anomalo) sono stati letti e trasformati in formato DataFrame, la struttura dati principale di questa libreria, in modo da poter trattare i dati sotto forma di tabelle bidimensionali.
Successivamente, per ognuna delle tabelle ottenute viene effettuato un campionamento randomico con lo scopo di ottenere un numero di campioni uguale per ognuna delle classi di malware da predire. In questo modo, si evita che in seguito siano presenti un numero di osservazioni diversi per ogni classe, e di conseguenza vengono evitati dal principio fenomeni di oversampling e undersampling.
Infine, ad ognuna delle tabelle viene aggiunta la colonna 'type', a cui viene associata una stringa che permette di identificare la classe di appartenenza della tabella in questione, per poi fonderle insieme in un'unica tabella, che costituirà in toto l'effettivo dataset a partire dal quale verranno effettuate tutte le operazioni di validazione, preprocessing e classificazione, che saranno descritte nelle seguenti sezioni.

3.3.2 Data Cleaning e Feature Scaling

Prima di applicare gli algoritmi di preprocessing, ho ritenuto necessario applicare delle tecniche di pulizia e normalizzazione. Fortunatamente, il dataset originale non presenta dati mancanti, quindi non é stato necessario effettuare alcun tipo di imputazione sui dati.
Sempre osservando il dataset, sono emerse due criticità:

- tramite l'ausilio di pandas, é stata effettuata una ricerca dei duplicati, la quale ha dato esito positivo, e dunque era necessario agire anche in tal senso
- con l'ordinamento originario, i data points consecutivi risultavano essere molto simili fra di loro, fattore che avrebbe potenzialmente influenzato in modo negativo il modello

Dopo aver confermato la presenza di duplicati, attraverso una ricerca più approfondita con i dati a disposizione di tutti i dispositivi, é emerso che era presente un duplicato ogni 100 data points, che su un totale di 7 milioni di osservazioni poteva creare qualche anomalia, e grazie ad un apposita funzione di pandas essi vengono eliminati dal dataset ad ogni run prima di andare incontro a qualsiasi operazione.
Per quanto riguarda la questione della somiglianza fra record consecutivi nel dataset, ho ritenuto eccessivo applicare una riduzione della dimensionalità in quanto si sarebbe perso il potenziale informativo del dataset, dunque ho optato per un semplice shuffling delle righe, in modo tale da non trarre in 'inganno' il modello nella successiva fase di classificazione.

Per la fase di feature scaling, sono stati utilizzate due tecniche, quali **Standard Scaling** e **Min-Max Normalization**.
Lo **Standard Scaling** [43] é una tecnica che si basa sull'assunzione che i dati siano normalmente distribuiti su ogni feature, e che la loro distribuzione sia centrata attorno allo zero, con una variazione di al più uno. Lo scaling avviene in modo indipendente a partire da un calcolo effettuato a partire da ciascuna feature a partire dai campioni nel dataset. Questo metodo ha il vantaggio di non essere troppo influenzato dalla presenza valori molto lontani dalla normale distribuzione, ma lo svantaggio di basarsi sull'assunzione che il dataset abbia sia normalmente distribuito.
La **Min-Max Normalization** [43] é una tecnica che trasforma il valore di ogni feature scalandolo all'interno di un range che può essere personalizzato fra qualsiasi valore (generalmente si usa 0 ed 1), ed i valori del minimo e del massimo di questo range corrispondono a minimo e massimo valore nel dataset considerato, mentre gli altri si troveranno all'interno di questo

range. Questa tecnica ha il vantaggio di non richiedere un dataset di partenza normalmente distribuito, ma lo svantaggio di essere pesantemente influenzato dai valori molto lontani dalla normale distribuzione. Tali valori prendono il nome di **outliers**, che costituiscono dei valore anomali in un insieme di osservazioni, ossia valori chiaramente distanti dalle altre osservazioni disponibili. Essi possono costituire sia delle entry molto utili ed informative per la caratterizzazione di una data classe, sia possono essere causa di gravi anomalie nella fase di classificazione. Nel nostro dataset gli outliers sono presenti, seppure in quantità molto limitata rispetto alla totalità dei record a disposizione, dunque non creano generalmente problemi. Ciò nonostante, in seguito saranno discussi approcci utilizzati per la rimozione degli stessi, con i relativi risultati.

Per applicare queste tecniche di scaling, é stata utilizzata la libreria *Scikit-Learn*, la quale presenta numerosi algoritmi e tecniche utili alla realizzazione di una pipeline di machine learning, in quanto comprende tool per l'analisi dei dati, l'apprendimento dei modelli, la valutazione dei risultati, etc.

3.3.3 Tecniche di preprocessing

L'applicazione di tecniche di preprocessing é stata necessaria affinché l'insieme dei dati forniti in input al modello fossero facilmente analizzabili ed interpretabili. Come anticipato nel primo capitolo, saranno previste diverse tecniche di preprocessing, quali rimozione di outliers, Variance Threshold, K-Best Selection e Principal Component Analysis.
Sebbene siano tutte sono orientate alla riduzione della complessità del modello, sono utilizzate sia tecniche di **Feature Selection** che di **Feature Extraction**. Con il processo di Feature Selection le features originali sono mantenute e vengono ridotte in base ad un certo criterio, mentre con la Feature Extraction le caratteristiche del dataset sono trasformate in uno spazio di features completamente nuovo, a partire dal quale viene applicata la riduzione di dimensionalità. Tutte le tecniche utilizzate, descritte nei seguenti punti, sono messe a disposizione dalla libreria Scikit-Learn:

- La **rimozione degli outliers** [40] é una tecnica che ha lo scopo di rimuovere i data points che si discostano di molto dalla normale distribuzione di un dataset e che sono causa di anomalie e risultati scadenti in fase di classificazione. Il sistema implementato prevede due tecniche di rimozione di outliers, tramite *Z-Score* e *Interquartile Range*.
  La prima tecnica prevede l'assegnazione di un punteggio ad ogni record in base a quanto esso sia vicino alla media di tutti i record, e se tale punteggio supera una determinata soglia, il record in esame é identificato come outlier e viene rimosso. L'approccio con Interquartile Range utilizza una tecnica statistica per misurare la dispersione dei record in un dataset, andando a dividere tutti i record in quattro parti, che contengono in ordine il più piccolo quarto ed il più grande quarto di dati. Andando a considerare i dati che risiedono fra il primo ed il terzo quarto, siamo in grado di determinare i record che appartengono alla media, e di conseguenza tutti gli altri valori possono essere considerati come outlier ed essere rimossi. L'unico problema che risiede nella rimozione degli outlier é che alcuni di essi risultano essere indispensabili per il valore informativo del dataset, e la loro eliminazione può costituire un problema piuttosto che un vantaggio, motivo per il quale quest'operazione dev'essere sempre effettuata sulla base di analisi specifiche e in presenza di problemi di sbilanciamento sui dati di addestramento.

- Il **Variance Threshold** [31] é una tecnica di Feature Selection, che consiste nel rimuovere tutte le feature dove la varianza non rispetta una soglia prestabilita. Il concetto su cui si basa é che le features che hanno valore uguale o simile sulla totalità dei campioni

non costituiscono informazioni significative per il modello, e dunque possono essere rimosse. Questo metodo ovviamente non tiene conto delle relazioni tra features, ed é per questo motivo che é sempre opportuno analizzare a priori l'importanza delle stesse.

- Il **K-Best Selection** [11] é una tecnica di Feature Selection, che consiste nell'assegnare un punteggio ad ogni feature, attraverso una funzione appositamente scelta che determina la bontà della stessa, e successivamente mantenere le k migliori features in relazione al loro punteggio. Il vantaggio di questo approccio é che permette di scegliere il numero di features da mantenere, in modo da utilizzare le features con maggiore potenza predittiva, ma senza ridurre eccessivamente le caratteristiche del dataset, ma come la precedente ha il problema di non tenere conto delle relazione tra features.

- Il **Principal Component Analysis** (**PCA**) [32] é una tecnica di Feature Extraction, che a differenza delle precedenti, é in grado di determinare le relazioni tra le features del dataset e l'importanza di ognuna di esse, e sulla base di tali informazioni applicare la riduzione di dimensionalità.
  In particolare, vengono identificate le suddette relazioni mediante una matrice di covarianza, e tramite un processo di decomposizione degli autovalori, vengono calcolati i relativi autovettori e autovalori. Tramite gli autovettori, trasformiamo i dati a disposizione in componenti principali, che costituiranno il nuovo spazio di features. Infine, tali componenti sono poi ordinati in ordine crescente di valore, e in questo modo é possibile ottenere una classifica della migliori features e rimuovere quelle con minor potere predittivo.

Ovviamente data la loro diversa natura di funzionamento, la scelta di quale utilizzare sarà effettuata sulla base degli algoritmi di classificazione utilizzati, che saranno discussi a breve.

3.4 Data Modeling

In questa sezione sarà discusso il core del modello, nonché la fase principale del CRISP-DM, quella di **Data Modeling**. Questa fase prevede la definizione degli algoritmi di classificazione da adottare in relazione al problema in esame e alle caratteristiche dei dati a disposizione, per poi stabilire quale tecnica massimizza il risultato.
Saranno innanzitutto descritte le tecniche di validazione che saranno utilizzate per valutare le prestazioni del modello, per poi discutere degli algoritmi da classificazioni testati ed impiegati, con un focus sui principi di funzionamento di ognuno insieme a pregi e difetti che li contraddistinguono, per una migliore interpretazione dei risultati nella fase successiva.

3.4.1 Tecniche di validazione

Nell'ambito del Machine Learning, con validazione ci si riferisce al processo mediante il quale un modello é valutato, utilizzando una partizione del dataset di partenza appunto per determinare la bontà delle predizioni effettuate. Affinché questa tecnica sia fattibile, é necessario che il partizionamento avvenga prima dell'addestramento del modello, motivo per il quale questa parte della pipeline é trattata prima della fase di addestramento. Per questa fase, sono state impiegate tre tecniche, quali Train-Test Split, K-Fold Cross Validation e impiego di matrici di confusione, che saranno descritte nei seguenti punti:

- Il **Train-Test Split** [38] costituisce una tecnica facile e veloce, che consiste nel dividere il dataset originale in due partizioni separate, una di training, che dunque sarà utilizzata per l'addestramento del modello, ed una di testing, che sarà utilizzata per determinare la bontà del modello una volta addestrato. Generalmente, tale partizionamento é effettuato con proporzione 80/20, dunque l'80% del dataset é utilizzato per l'addestramento, e il restante 20% per la fase di predizione e valutazione. Ha il vantaggio di essere semplice ed immediata da realizzare, ma non é molto efficiente per piccoli dataset, in quanto può accadere che i record più qualitativi e informativi possano andarsi a trovare solo in una delle due partizioni.

- La **K-Fold Cross Validation** [38] é una tecnica che consiste nel partizionare il dataset di partenza in k sottoinsiemi di uguale dimensione. La particolarità di questo processo é che la validazione é incrociata, dunque il processo di validazione é ripetuto k volte, per far sì che ognuna delle k partizioni venga utilizzata esattamente una volta per la validazione. Terminato il processo di validazione incrociata, a partire dai k risultati

ottenuti può essere fatta la media, oppure può essere semplicemente considerata la k-esima iterazione che ha ottenuto il miglior risultato.
Il vantaggio di questo metodo é che tutte le osservazioni ottenute saranno utilizzate sia per l'addestramento che per la validazione, dunque si ha una convalida a 360 gradi su tutti i record del dataset, ed é inoltre molto indicata per piccoli dataset.

Per ottenere delle metriche quanto più veritiere possibili a partire da qualsiasi tecnica di validazione, é opportuno utilizzare insiemi dei dati indipendenti per il testing, che si discostino da quelli che sono i dati di partenza. Mentre la K-Fold Cross Validation consente di utilizzare tutte le singole osservazioni sia per l'addestramento che per il testing, di contro richiede una finestra temporale di esecuzione molto maggiore, dato che la classificazione va eseguita per tutte le coppie di dati per training/prediction, e inoltre é particolarmente indicata per dataset di partenza molto piccoli. Il train/test split, invece, costituisce una tecnica semplice, con un tempo di esecuzione molto ridotto, e che in considerazione della grandezza del nostro dataset é in grado di fornire metriche in fase di valutazione affidabili, motivo per il quale sarà la tecnica di validazione preferita considerando il nostro caso d'utilizzo. Ciò nonostante, la K-Fold Cross validation è stata comunque implementata nel caso in cui, in futuro, si utilizzi il sistema con un dataset più piccoli ed in ogni caso si voglia prediligere il suo utilizzo.

3.4.2 Algoritmi di classificazione impiegati

Come anticipato, il modello di partenza prevedeva come classificatore una rete neurale, con la quale erano stati raggiunti risultati già accettabili, con valori fra accuracy, precision e recall circa dell'85%. Ciò nonostante, è stata ritenuta necessaria l'implementazione tutte le tipologie di classificatori più utilizzati per essere in grado di identificare il modello con le prestazioni più alte. A tal proposito, sono stati implementati e testati i seguenti algoritmi di classificazione:

- Una **rete neurale** [30] é costituita da una serie di algoritmi che hanno lo scopo di individuare le relazioni nascoste in un insieme di dati tramite un processo che simula il modo in cui opera il cervello umano. A livello pratico, una rete neurale é costituita da un certo numero di livelli, dove si distinguono il layer di input, che riceve appunto l'input originale, i layer di processing, attraverso i quali i dati scorrono più volte e il layer di output, che processa in modo definitivo i dati ricevuti.
  Ciascuno di questi livelli é costituito da nodi, chiamati *neuroni*, ognuno dei quali é connesso a quelli adiacenti mediante degli archi. Quando un neurone riceve un input,

esso viene processato e viene fornito l'output tramite gli archi ai nodi adiacenti, che a loro volta viene processato ulteriormente. Ogni neurone ed ogni arco hanno un peso, che viene modificato man mano che l'addestramento procede, e che serve a determinare se e in che modalità processare l'input ricevuto. Questo processo di addestramento viene effettuato per tutti i dati di addestramento disponibili, e ripetuto per un certo numero di iterazioni, dette epoche. Quando la differenza fra il valore dell'output atteso e il valore dell'output ottenuto si trova al di sotto di una certa soglia, e/o quando un certo numero di epoche é raggiunto, l'addestramento termina. In fase operativa, sulla base dei pesi stabiliti vengono processati gli input ricevuti dalla rete.

- **Naive Bayes** [37] é un algoritmo probabilistico, basato sul *teorema di Bayes*, che a partire dalle features dell'istanza da classificare, calcola la probabilità che questa faccia parte di una classe piuttosto che un altra, sotto l'assunzione che le features non siano in alcun modo correlate l'una con l'altra (da qui il nome Naive, ingenuo), dunque non verrà valutata in fase di classificazione la potenziale utilità data dalla combinazione di più caratteristiche.
  Operativamente, l'algoritmo calcola prima la probabilità di ogni classe, data dalla frequenza delle istanze di una certa classe rispetto a quelle totali, poi calcola la probabilità condizionata, ovvero la probabilità di appartenenza ad una certa classe rispetto all'appartenenza ad altre classi, per poi determinare effettivamente la classe di appartenenza sulla base del valore della probabilità più alto ottenuto.

- **Decision Tree** [21] é un algoritmo che mira a creare un albero i cui nodi rappresentano un sottoinsieme delle features del problema e i cui archi rappresentano le decisioni che porteranno man mano alla classificazione di un istanza. Questo algoritmo parte dalla definizione di un nodo radice, a cui é associata la feature con maggiore potenza predittiva, e divide il dataset in sottoinsiemi composti da valori simili per una certa caratteristica. Viene ripetuta questa operazione fino a quando non si ottengono tutti *sottoinsiemi puri*, ovvero sottoinsiemi di dati che contengono data points appartenenti ad una sola classe. La divisione del dataset nei vari sottoinsiemi é effettuato in base ad un fattore detto *entropia*, che misura il grado di difficoltà di interpretazione di una feature.

- **Random Forest** [16] é un algoritmo che sfrutta il funzionamento dei Decision Tree per la classificazione. Esso si basa sul concetto in base al quale ogni albero decisionale considera un sottoinsieme più o meno casuale di feature, e dunque possiede in qualsiasi

caso un margine d'errore. Costruendo una *foresta* di alberi decisionali, invece, si ha modo di creare alberi decisionali costruiti a partire da diversi sottoinsiemi di feature, e dunque andando ad analizzare le stime di tutti gli alberi decisionali costruiti si ha modo di effettuare una predizione in modo molto più preciso. Ovviamente, saranno necessarie più risorse ed un tempo di addestramento di gran lunga maggiore per effettuare quest'operazione.

- **K-Nearest Neighbors** [25] é un algoritmo il cui funzionamento sta nello stimare la probabilità che un record appartenga ad un gruppo piuttosto che un altro sulla base della distanza di tale data point ai gruppi in considerazione. É considerato un algoritmo 'pigro', in quanto non effettua alcun calcolo o operazione in fase di addestramento, ma si limita ad immagazzinare i dati. In fase di classificazione, effettuerà una valutazione sulla base della distanza euclidea delle k classi più vicine al record considerato, e l'attribuzione sarà fatta in base alla classe che ha valutazione migliore.
  Questo algoritmo é molto indicato per la sua facilità ed efficienza, ma in presenza di molte features può risultare essere inefficiente.

- **Support Vector Machine** [20] é un algoritmo che si basa sull'idea di trovare un iperpiano che divida al meglio il dataset in due classi. Un iperpiano può essere considerato come una linea che divide il dataset in due parti per classificarne i record, e più lontani sono i record dall'iperpiano, più é probabile che esso appartenga a quella classe. A tal proposito sono identificati i vettori di supporto, che costituiscono i record più vicini a quella classe, che se rimossi alterano la posizione dell'iperpiano e di conseguenza la successiva fase di classificazione. Quest'algoritmo é molto indicato per la sua accuratezza in particolare per piccoli dataset, ma può essere inefficiente in presenza di dataset rumorosi e di grandi dimensioni.

I risultati degli algoritmi sopra descritti, insieme alle tecniche di preprocessing scelte ed impiegate per ognuno di essi, saranno esposti nel seguente capitolo.

CAPITOLO 4

Valutazione della tecnica

In questo capitolo sarà trattata la fase di Evaluation, durante la quale é effettuata l'analisi e la valutazione dei risultati, e viene stabilito se essi sono in linea con gli obiettivi di business stabiliti durante la prima fase di progettazione. In particolare verranno descritte le metriche impiegate per la valutazione del modello, e saranno confrontati i risultati ottenuti dai vari algoritmi di classificazione. A partire da questi, sarà determinato quale algoritmo é il migliore e perché per il nostro sistema di Intrusion Detection. Infine, verranno descritti quali sono i punti di forza e le mancanze del sistema implementato.

4.1 Metriche di valutazione adottate

Come per ogni modello che si rispetti, sono state previste delle metriche di valutazione, necessarie per valutare la bontà dello stesso ed i risultati ottenuti da tutti i possibili punti di vista. La scelta delle metriche impiegate per un modello é dettata dal tipo di problema in analisi e specialmente da quali informazioni di qualità vogliamo ricavare dall'analisi del risultati.

Per il nostro modello, in accordo con i requisiti definiti, vogliamo tener conto non solo della percentuale di predizioni corrette (dunque veri positivi), ma si é puntato in particolare a fare un'analisi relativa a falsi positivi, falsi negativi e veri negativi, correlando il tutto con grafici addizionali, facendo in altre parole un'analisi prestazionale a 360 gradi con tutti i possibili valori di riferimento, in modo da capire al meglio come e perché il modello sbaglia, ed effettuare miglioramenti mirati. A tal proposito, sono state utilizzate le seguenti metriche [24]:

- L'**Accuracy** é una metrica ottenuta calcolando il rapporto fra il numero di predizioni corrette rispetto al numero di predizioni totali fatte per un dataset.

- La **Precision** é una metrica ottenuta calcolando il rapporto fra i veri positivi e la somma dei veri positivi e i falsi positivi. Questa metrica tiene conto sia dei campioni positivi che negativi classificati, dunque considera quando un campione é classificato come positivo, senza considerare la corretta classificazione di tutti i campioni positivi.

- La **Recall** é una metrica ottenuta calcolando il rapporto fra i veri positivi, e tutti i positivi restituiti dal modello. Questa metrica ci da informazioni sulla corretta classificazione dei soli campioni positivi, senza considerare però se un campione negativo sia classificato come positivo.

- L'**F1 Score** é una metrica che combina sia precision che recall, e in particolare é calcolata sulla base di una media pesata di queste due metriche, con un range che va da 0 a 1.

- La **matrice di confusione** é una tabella 2x2 che mostra il numero di predizioni corrette ed errate per ognuna delle classi, confrontando le predizioni del modello con gli effettivi valori del dataset di test, in modo da fornire informazioni sul tipo di errori commessi. I valori sulla diagonale principale rappresentano tutti i record per cui la predizione é stata effettuata correttamente, mentre quelli sulla diagonale obliqua rappresentano tutti i record per cui la predizione é stata effettuata in modo errato.
  Nel nostro caso, sono state impiegate sia matrici di confusioni *binarie*, che mostrano le suddette informazioni per ogni singola classe, per un'analisi più specifica, sia matrici di confusioni *multiclasse*, dove ogni riga denota il numero di previsioni totali sia corrette che errate per ognuna della classi da predire, per un'analisi più ad alto livello.

Per ottenere i suddetti valori é stata utilizzata ancora la libreria sklearn, mentre per i grafici delle matrici di confusioni é stato utilizzato matplotlib e seaborn, entrambe librerie utili alla produzione di grafici informativi.

4.2 Risultati del modello e osservazioni

Prima di passare all'analisi dei risultati per questa fase di **Evaluation**, é importante dire che la scelta delle tecniche di validazione, preprocessing e normalizzazione adottate per ognuno degli algoritmo di classificazione é stata effettuata sulla base di una serie di 'trial and error', con lo scopo di trovare le combinazioni che restituiscano le migliori performance da ognuno di essi, e successivamente poter determinare quella con il miglior risultato. Per semplicità tale processo non sarà documentato, ma si passerà direttamente all'elencare per ogni classificatore le combinazioni migliori di algoritmi di Data Preparation.
Per quanto riguarda i valori mostrati nei risultati, in considerazione del fatto che ogni combinazione di dataset di training/test risulta essere diversa ogni run, per ognuno dei classificatori saranno eseguite tre run per dispositivo, utilizzando i dati di tre dispositivi (Provision, Philips ed Ennio), che sono quelli con le caratteristiche più slegate fra loro rispetto agli altri dispositivi a disposizione. Dunque, saranno effettuate un totale di 9 run per classificatore, e il risultato mostrato sarà una media dei valori ottenuti. Inoltre, tutte le classificazioni effettuate sfruttano le medesime tecniche di Data Cleaning, descritte nel capitolo precedente, e la stessa tecnica di validazione, ovvero train/test split (80% train, 20% test).
I risultati sono mostrati nella seguente tabella:

Classificatore	Normalizzazione	Preprocessing	Metriche
Decision Tree	Min-Max Normalization	Nessuna tecnica	Accuracy: 93.78% Precision: 90.06% Recall: 93.78% F1: 91.45% Train time: 24.18s
Random Forest	Min-Max Normalization	Nessuna tecnica	Accuracy: 93.83% Precision: 96.28% Recall: 93.83% F1: 91.53% Train time: 216.94s
Support Vector Machine	Z Score	Variance Threshold	Accuracy: 87.38% Precision: 92.18% Recall: 92.18% F1: 84.33% Train time: 580.77s
K-Nearest Neighbours	Min-Max Normalization	K-Best Selection	Accuracy: 99.85% Precision: 99.85% Recall: 99.85% F1: 99.85% Train time: 0.10s
Naive Bayes	Z Score	Nessuna tecnica	Accuracy: 82.98% Precision: 80.49% Recall: 82.98% F1: 80.10% Train time: 0.41s
Neural Network 1	Z Score	PCA	Accuracy: 93.31% Precision: 89.62% Recall: 93.31% F1: 90.98% Train time: 118.69s
Neural Network 2	Z Score	PCA	Accuracy: 90.92% Precision: 86.51% Recall: 90.92% F1: 87.99% Train time: 96.03s

Tabella 4.1: Tecniche utilizzate e risultati ottenuti con i classificatori impiegati

Come é possibile notare, tecniche quali rimozioni di outliers e feature selection ed extraction non sono state mai o quasi impiegate, questo perché il dataset in esame si é rivelato essere molto ben bilanciato ed eterogeneo, ed esente da dati ridondanti o rumorosi.
Gli outlier presenti, così come praticamente quasi tutte le features, costituiscono tutti elementi caratterizzanti, e non fanno altro che aumentare la potenza predittiva totale, dunque in tal senso le tecniche di preprocessing sono state utilizzate in modo pressoché marginale, in quanto il loro utilizzo 'eccessivo' talvolta risultava diminuire le performance.
Da questi risultati è facile evincere che il miglior classificatore è il **K-Nearest Neighbors**, che con il suo risultato di 99.85% in tutte le metriche di riferimento adottate, ha performance che rasentano la perfezione. Non ci si è limitati solo ad affermare questo, in quanto si è cercato di individuare la motivazione per la quale ci sia questa netta differenza nei risultati, comunque ottimi per quasi tutti gli altri classificatori. Con l'ausilio delle matrici di confusione, si è potuto osservare che tutti gli altri classificatori compiono sempre lo stesso errore, ovvero sbagliano a causa della notevole somiglianza dei record associati alle classi 'gafgyt_tcp' e 'gafgyt_udp'. A fine sezione, è mostrato come con due classificatori diversi si manifesti lo stesso problema, mentre con KNN si abbia una classificazione impeccabile.
Il K-Nearest Neighbours, per la sua natura di funzionamento, ovvero per il fatto che banalmente in fase di training non venga effettuata nessuna operazione se non quella di immagazzinare i dati in modo grezzo, e che si deleghi alla fase di prediction l'analisi della classe più vicina al record in esame sulla base della distanza euclidea, fa sì che si abbia una situazione ottimale a partire dalla quale classificare le istanze nel nostro problema, e che dunque nel nostro caso di utilizzo risulta essere ideale.

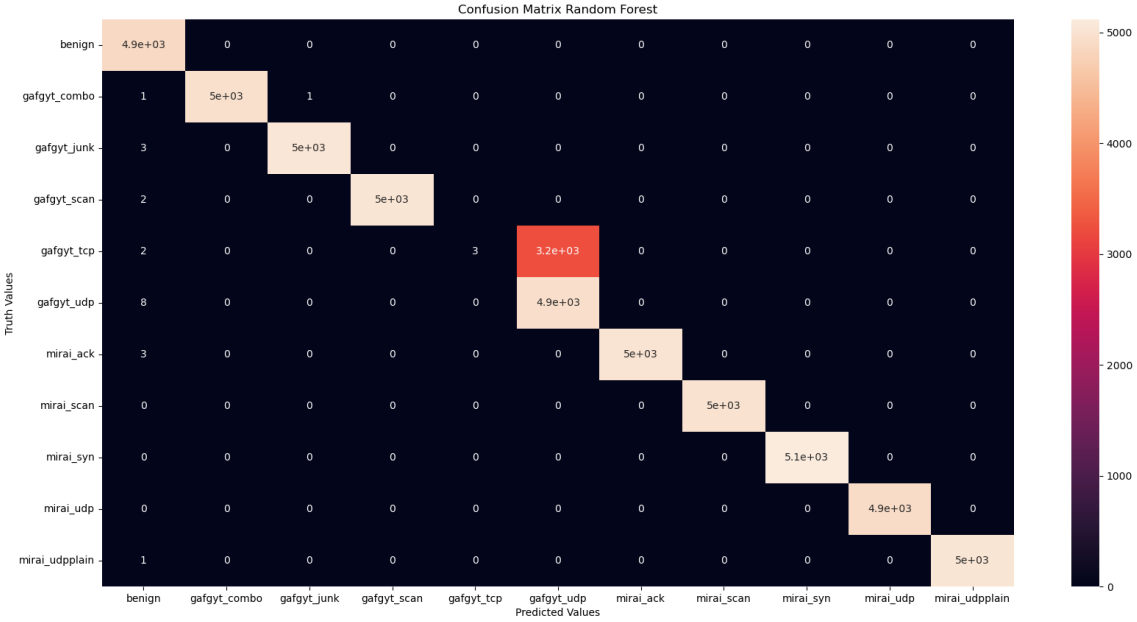

Figura 4.1: Matrice di confusione prodotta dal classificatore Random Forest

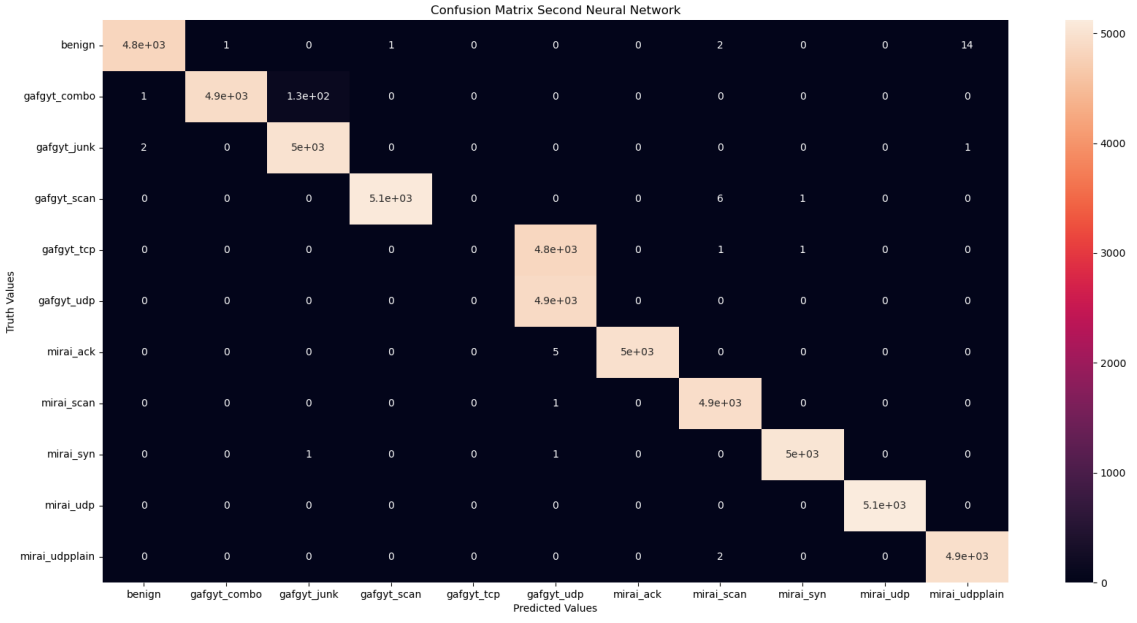

Figura 4.2: Matrice di confusione prodotta dalla seconda rete neurale

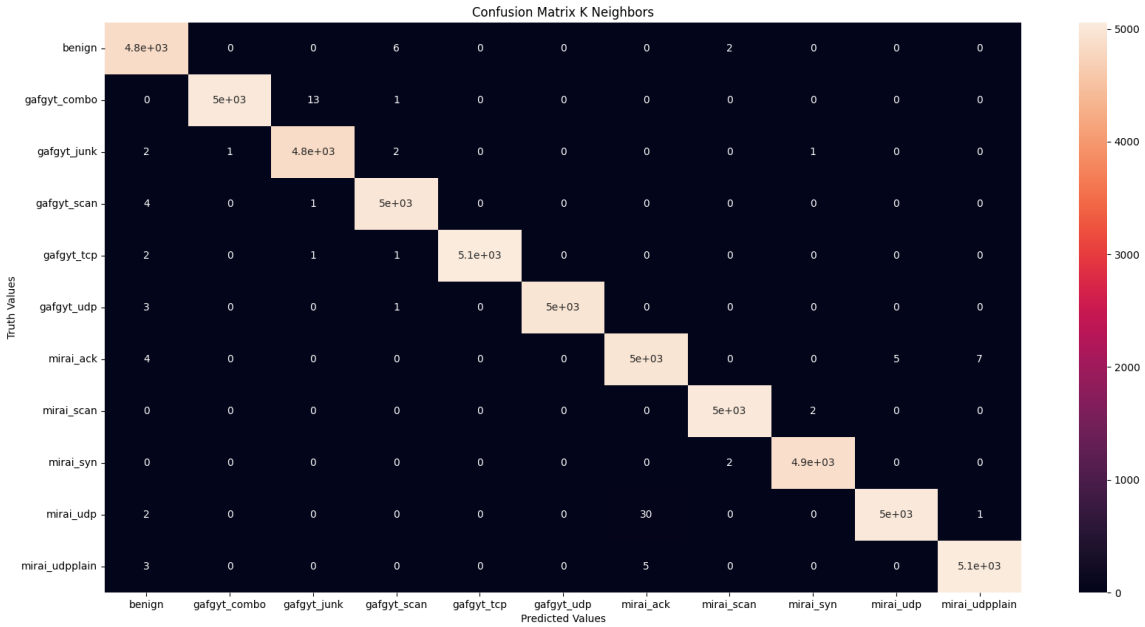

Figura 4.3: Matrice di confusione prodotta dal classificatore K-Nearest Neighbors

4.3 Punti di forza e debolezza

Tirando le somme, ci si può ritenere soddisfatti dei risultati ottenuti, in quanto sono stati rispettati tutti gli obiettivi di business definiti durante la prima fase di progettazione, superando anche le aspettative. Il sistema sviluppato, a partire da dati sul traffico di rete di vari dispositivi IoT, permette di determinare la presenza di malware e persino essere in grado di classificarli in 10 diverse varianti appartenenti alla famiglia Mirai e BASHLITE.
Per quanto riguarda le performance, tramite l'ausilio di un discreto calcolatore, é possibile effettuare l'analisi del traffico dati in tempi relativamente stretti, e soprattutto con dei risultati assolutamente affidabili.
Quanto a fruibilità, il sistema non ha assolutamente problemi, in quanto i dati a partire dai quali avviene l'Intrusion Detection sono generati usufruendo di un semplice tool per l'analisi del traffico di dati. In particolare, il dataset utilizzato é stato generato dall'osservazione del traffico dati tramite l'API **pcap**, che fornisce un'interfaccia ad alto livello per il tracciamento del traffico dei pacchetti, in particolare permettendo di catturare tutti i pacchetti in movimento sulla rete, compresi quelli in entrata e in uscita da altri host. Questo implica che l'utilizzo di tale API o software simili, permette di ottenere facilmente ulteriori dati a partire dai quali effettuare altri esami ed analisi su altre reti e/o dispositivi, utilizzando gli algoritmi e i classificatori a proprio piacimento fra quelli messi a disposizione del software.
Chiaramente questo sistema non é esente da limiti. Come già discusso nel capitolo 2, i malware classificabili dal sistema non sono gli unici a poter interessare i dispositivi IoT. Per l'addestramento del modello si é voluto rigorosamente scegliere un set di dati che fosse stato ottenuto a partire da un'autentica presenza di malware, ma la mancanza di infrastrutture e studi adeguati rende impossibile la fruizione di ulteriori dati relativi ad una famiglia molto più ampia di malware.
A ciò si aggiunge il fatto che le maggiori aziende nel campo IoT sono sempre più restie a rilasciare dati relativi ai traffici in quanto reclamano rischi per la sicurezza dei loro dispositivi, ma questo fattore sta rendendo impossibile la produzione di un sistema universalmente affidabile per l'identificazione di virus e malware.

Inoltre, andrebbe affrontata quella che é l'ultima fase dell'approccio CRISP-DM, ovvero il **Deployement**. Mettere in funzione il software sviluppato e renderlo utilizzabile e fruibile ad utenti di qualsiasi livello richiederebbe il possesso di dovute autorizzazioni, infrastrutture e controlli oltre ad un meccanismo di manutenzione opportuno, che ai fini dello studio in questione sono chiaramente superflui, ma che potrebbero tranquillamente essere ottenuti e realizzati da parte do grandi aziende e produttori nel momento in cui si decida di muoversi definitivamente verso una soluzione definitiva e totale ai problemi di sicurezza e privacy nell'IoT.

CAPITOLO 5

Conclusioni e sviluppi futuri

In quest'ultimo capitolo viene fatto un riassunto di tutto ciò che é stato fatto, a partire dalla progettazione del software tramite l'approccio ingegneristico CRISP-DM fino alla discussione dei risultati finali, e vengono fatte delle considerazioni sulla modalità con le quali il software può essere utile e in che senso ci si può muovere per estenderlo e migliorarlo, insieme ad alcune considerazioni scaturite dallo studio e dal lavoro svolto.

Data la crescente adozione di soluzioni orientate all'IoT nel campo della domotica, entertainment, salute, militare e così via, garantire una comunicazione ed un utilizzo sicuro ed esente da rischi per la privacy é diventato fondamentale [26]. Gli utenti sono spesso allo scuro di tutti i problemi di sicurezza relativi ai dispositivi che utilizzano tutti i giorni, e questo é anche dovuto alla politica di aziende che non sono aperte a condividere dati relative al traffico e rendono dunque infattibile la realizzazione di sistemi ad hoc di prevenzione ed individuazioni di malware.
A tal proposito, é stato deciso di agire in modo diretto. Basandosi sul modello CRISP-DM, é stato progettato e sviluppato un sistema di Intrusion Detection intelligente, che a partire da dati reali sul traffico di rete fra diversi dispositivi IoT, é in grado di determinare la presenza di una botnet nella propria rete, analizzando la regolarità del traffico, e in caso di anomalie individuare di quale malware la rete fosse vittima. Per fare ciò, é stata innanzitutto individuato un dataset che contenesse dati ottenuti in maniera autentica e non simulata, e sugli stessi è effettuata un'approfondita analisi dei dati per determinare le caratteristiche e le relazioni fra gli stessi. Successivamente é stata definita e creata la pipeline, che comprende una fase di data cleaning, ed è corredata di diverse tecniche di data normalization, data preprocessing,

feature selection ed extraction e rimozione di outlier. Per effettuare uno studio più esteso e per cercare di ottenere il miglior risultato possibile, si è scelto di utilizzare diverse tipologie di classificatori, quali Support Vector Machine, Naive Bayes, Decision Tree, Random Forest, due versioni di reti neurali e K-Nearest Neighbors. Dopo aver effettuato tre run per ognuno dei tre dispositivi selezionati per il testing, sfruttando il train/test split come tecnica di validazione, proprio il K-Nearest Neighbors si è rivelato essere il migliore, con un valore fra accuracy, precision, recall e F1 score in media del 99.85%.

Ovviamente il sistema di Intrusion Detection non è da considerarsi terminato, in quanto a causa della limitatezza dei dati a disposizione, riesce solamente a rilevare la presenza di malware appartenenti alla famiglia Mirai e BASHLITE, che sappiamo essere solo una piccola parte della totalità dei malware a cui possono essere soggetti i dispositivi e i sistemi IoT. Considerando, però, che i dati utilizzati per l'addestramento sono stati ottenuti a partire da diverse misurazioni sul traffico tramite l'API pcap, la quale é open source e la cui validità e affidabilità sono universalmente riconosciute, questo ci porta a riflettere sulle vere ragioni per le quali il mondo IoT si trova in una situazione di sicurezza precaria. Finché non ci saranno sistemi costruiti ad hoc, con apposite infrastrutture, che permettano di effettuare rilevazioni a partire da dispositivi IoT reali e infezioni di malware autentiche, non si potrà mai avere un sistema di prevenzione e rilevamento efficace ed affidabile, e questa colpa è da attribuire alle aziende del settore IoT che con la loro politica sempre più restrittiva e limitante rendono impossibile accedere ai loro dispositivi e reperire dati per la creazione di tali sistemi. Finché non ci muoverà in tal senso e non si avrà modo di ottenere questa tipologia di dati, purtroppo la problematica questione di sicurezza e privacy per l'IoT non farà alcun passo in avanti.

Bibliografia

[1] 101Blockchains. Security and privacy issues in the internet of things, 2021. (Citato alle pagine 20, 22 e 23)

[2] Mahmoud Abbasi, Amin Shahraki, and Amir Taherkordi. Deep learning for network traffic monitoring and analysis (ntma): A survey. *Computer Communications*, 170:19–41, 2021. (Citato a pagina 35)

[3] Montdher Alabadi and Zafer Albayrak. Q-learning for securing cyber-physical systems : A survey. pages 1–13, 06 2020. (Citato a pagina 31)

[4] Ömer Aslan Aslan and Refik Samet. A comprehensive review on malware detection approaches. *IEEE Access*, 8:6249–6271, 2020. (Citato a pagina 33)

[5] BehrTech. 6 leading types of iot wireless tech and their best use cases, 2021. (Citato a pagina 12)

[6] Kirk Bresniker, Ada Gavrilovska, James Holt, Dejan Milojicic, and Trung Tran. Grand challenge: Applying artificial intelligence and machine learning to cybersecurity. *Computer*, 52(12):45–52, 2019. (Citato a pagina 27)

[7] BusinessInsider. The security and privacy issues that come with the internet of things, 2022. (Citato a pagina 19)

[8] Mauro Cherchi. *La sicurezza nell'Internet of Things*. PhD thesis. (Citato alle pagine 20 e 26)

[9] Ambika Choudhury. Why is crisp-dm gaining grounds, 2020. (Citato a pagina 37)

[10] CMSWiRe. 7 big problems with the internet of things, 2021. (Citato a pagina 17)

[11] DataTechNotes. Selectkbest feature selection, 2019. (Citato a pagina 53)

[12] Wells David. Unsw$_n b$15, 2019. (*Citatoapagina* 40)

[13] Aur'elien G'eron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. " O'Reilly Media, Inc.", 2019. (Citato alle pagine 27 e 30)

[14] Thales Group. Understanding iot security risks, 2022. (Citato alle pagine iv e 21)

[15] The Guardian. Ddos attack that disrupted internet was largest of its kind in history, experts say, 2016. (Citato a pagina 2)

[16] IBM. Random forest, 2022. (Citato a pagina 56)

[17] iotforall. What is an iot platform?, 2019. (Citato a pagina 10)

[18] JTIoT. The 5 types of iot platforms, 2019. (Citato alle pagine 10 e 11)

[19] Kaspersky. What is a botnet?, 2020. (Citato a pagina 23)

[20] KDNuggets. Support vector machines: A simple explanation, 2016. (Citato a pagina 57)

[21] KDnuggets. Decision tree algorithm, explained, 2022. (Citato a pagina 56)

[22] Murat Kuzlu, Corinne Fair, and Ozgur Guler. Role of artificial intelligence in the internet of things (IoT) cybersecurity. *Discover Internet of Things*, 1(1):7, February 2021. (Citato alle pagine 3, 19, 29 e 30)

[23] Ben Lutkevich. Intrusion detection system (ids), 2021. (Citato a pagina 34)

[24] Medium. Evaluation metrics for classification models, 2020. (Citato a pagina 58)

[25] Medium. K-nearest neighbor, 2021. (Citato a pagina 57)

[26] Francesca Meneghello, Matteo Calore, Daniel Zucchetto, Michele Polese, and Andrea Zanella. Iot: Internet of threats? a survey of practical security vulnerabilities in real iot devices. *IEEE Internet of Things Journal*, 6(5):8182–8201, 2019. (Citato a pagina 67)

[27] Kashif Naveed. N-baiot dataset to detect iot botnet attacks, 2020. (Citato alle pagine 4, 40 e 41)

[28] paloalto. What is a botnet?, 2022. (Citato a pagina 24)

[29] Pareteum. Internet of things: The five types of iot, 2021. (Citato a pagina 15)

[30] pathmind. A beginner's guide to neural networks and deep learning, 2022. (Citato a pagina 55)

[31] ProjectPro. Analytics vidhya, 2020. (Citato a pagina 52)

[32] ProjectPro. Principal component analysis, 2022. (Citato a pagina 53)

[33] Sohel Rana. Malicious network traffic pcaps-2021, 2021. (Citato a pagina 40)

[34] Imtithal Saeed, Ali Selamat, and Ali Abuagoub. A survey on malware and malware detection systems. *International Journal of Computer Applications*, 67:25–31, 04 2013. (Citato a pagina 25)

[35] Towards Data Science. A beginners guide to q-learning, 2019. (Citato a pagina 31)

[36] Towards Data Science. A beginners guide to q-learning, 2019. (Citato a pagina 31)

[37] Towards Data Science. Naive bayes algorithm for classification, 2021. (Citato a pagina 56)

[38] Towards Data Science. Train/test split and cross validation in python, 2022. (Citato a pagina 54)

[39] Ambika Shrestha Chitrakar and Slobodan Petrović. Efficient k-means using triangle inequality on spark for cyber security analytics. In *Proceedings of the ACM International Workshop on Security and Privacy Analytics*, IWSPA '19, page 37–45, New York, NY, USA, 2019. Association for Computing Machinery. (Citato a pagina 31)

[40] Statology. When to remove outliers in data, 2021. (Citato a pagina 52)

[41] Stefanos T. Iot intrusion detection, 2020. (Citato a pagina 48)

[42] Tech Target. What is the internet of things (iot)?, 2022. (Citato alle pagine iv e 2)

[43] Codecademy Team. Normalization, 2022. (Citato a pagina 50)

[44] TechTarget. Ethernet in iot still serves a purpose in the wireless age, 2020. (Citato a pagina 12)

[45] B.K. Tripathy and J. Anuradha. *Internet of Things(IoT): Technologies, Applications, Challenges and Solutions*. 10 2017. (Citato alle pagine 1 e 7)

[46] Wikipedia. Network eavesdropping, 2022. (Citato a pagina 22)

[47] Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. Iot security techniques based on machine learning: How do iot devices use ai to enhance security? *IEEE Signal Processing Magazine*, 35(5):41–49, 2018. (Citato alle pagine v, 24 e 32)

[48] Bei-Ni Yan, Tian-Shyug Lee, and Tsung-Pei Lee. Mapping the intellectual structure of the internet of things (iot) field (2000–2014): a co-word analysis. *Scientometrics*, 105(2):1285–1300, Nov 2015. (Citato a pagina 8)

[49] YISROELMIRSKY. Kitsune network attack dataset, 2020. (Citato a pagina 40)

[50] Pamela Zave and Jennifer Rexford. Patterns and interactions in network security, 12 2019. (Citato a pagina 36)

[51] Zhi-Kai Zhang, Michael Cheng Yi Cho, Chia-Wei Wang, Chia-Wei Hsu, Chong-Kuan Chen, and Shiuhpyng Shieh. Iot security: Ongoing challenges and research opportunities. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 230–234, 2014. (Citato a pagina 22)

[52] Zistemo. Iot: What is it and why is it important?, 2022. (Citato a pagina 8)

Ringraziamenti

Ci tengo innanzitutto a ringraziare il mio relatore, il Prof. Fabio Palomba, una persona eccezionale e disponibile che con le sue conoscenze e i suoi consigli é stato fondamentale per quest'ultima parte del mio percorso di studi, e non solo.
Ringrazio anche il Dott. Giammaria Giordano, che con il suo supporto e i suoi feedback sempre tempestivi é stato di grandissimo aiuto durante tutto il lavoro svolto.

Ringrazio la mia famiglia, per la loro pazienza nei miei tanti momenti bui, per l'affetto che mai mi hanno fatto mancare, per i valori che mi hanno saputo trasmettere, per avermi fatto sempre sentire speciale. Vi sarò eternamente grato, questo traguardo é tutto vostro.

Ringrazio la mia fidanzata Ludovica, per aver sempre cercato di migliorarmi le giornate e rendere meno amare le difficoltà, per non avermi mai fatto sentire solo, per la pazienza che ha avuto, per tutti i sorrisi che mi ha regalato. Grazie, sei unica.

Ringrazio tutti i miei amici, quelli delle mille risate, dei momenti di spensieratezza, delle parole di conforto, delle esperienze condivise insieme, che sono stati parte fondamentale della mia crescita. E ringrazio anche i miei 'nemici', che mi hanno forgiato e dato lezioni che mi hanno reso fiero di ciò che sono sono oggi.

Infine voglio ringraziare te, Matteo. Hai sempre cercato di credere in te stesso, hai lavorato sempre al massimo, non hai mai mollato, hai dato sempre più di quanto ricevuto, hai sempre saputo trasformare errori e debolezze in punti di forza, nonostante tutto.
Verso traguardi più grandi, **buona fortuna**.