UNIVERSITÀ TELEMATICA "e-Campus"

Analisi di algoritmi di Machine Learning per la rilevazione di attacchi DoS

Relatore: Callegari Christian

Tesi di Laurea di
Bonetti Alberto

Anno Accademico: 2017/2020

UNIVERSITÀ DEGLI STUDI
eCAMPUS
TELEMATICA · DM 30 · 01 · 2006

AUTORIZZAZIONE DIFFUSIONE DATI

Il/La sottoscritto/a
(cognome) Bonetti (nome) Alberto

nato/a a Domodossola (VB) il 03/11/1995

Residente in via Località Morandone 1

Cap 28873 Citta' Calasca-Castiglione Prov VB

Telefono 3282241810 Mail Alphire.nothing@gmail.com

Facoltà di INGEGNERIA

Corso di Laurea in INFORMATICA E DELL'AUTOMAZIONE (D.M. 270/04) [INIA]

☑ AUTORIZZA
☐ NON AUTORIZZA
(barrare la casella interessata)

L'Università telematica E-Campus a trasmettere i propri dati personali ad Enti Pubblici e Privati che ne facciano richiesta per collaborazioni professionali a vario titolo, stage o assunzioni presso le loro strutture.

Inoltre autorizzo il trattamento dei dati personali contenuti nel mio Curriculum Vitae,in base l'Art.13 del D.Lgs.196/2003.
(allegare CV)

☑ AUTORIZZA
☐ NON AUTORIZZA
(barrare la casella interessata)

Data _____ Firma studente _____

Indice

Elenco delle tabelle

Elenco delle figure

Abstract

Nel contesto odierno, dove l'affidabilità, robustezza e prontezza, sono qualità desiderabili e fondamentali per attori di discreto e alto livello operanti in internet, gli attacchi informatici che provocano un interruzione (o degradazione) dell'erogazione di uno o più servizi offerti agli utenti, ricoprono un aspetto importante nel campo della sicurezza informatica. Attacchi di questo genere, vengono detti DoS e DDoS, e rappresentano tutt'oggi[1] un campo aperto alla ricerca. L'importanza, la rilevanza e la continua evoluzione di tali offensive, in aggiunta all'uso sempre più predominante del Machine Learning (ML) / Deep Learning (DL) da parte della comunità scientifica, hanno indirizzato i ricercatori del settore, all'investigazione di possibili soluzioni, basate su tali tecnologie.
Nonostante il grande numero di lavori presenti nella letteratura corrente, una conclusione unanime sulla metodologia da adottare per mitigare od eliminare la problematica, non è ancora stata concordata all'interno dell'ambiente informatico. Questo sentimento discordante, è inoltre accompagnato da vari problemi correlati all'obsolescenza dei dataset, alle metodologie sulla valutazione e validazione dei modelli proposti, riscontrabili in numerosi lavori.
Proponiamo dunque un analisi riguardante le varie problematiche riscontrate e gli algoritmi/-metodologie maggiormente utilizzati in questo campo, con l'auspicio di delineare e chiarire la direzione generale consigliata da seguire, offrendo gli strumenti necessari per la stesura di nuovi lavori di ricerca.
Alberto Bonetti matricola 001028822 "Analisi di algoritmi di Machine Learning per la rilevazione di attacchi DoS".

[1] Ad esempio, per citare due attori coinvolti recentemente: nel febbraio 2020 Amazon Web Services (AWS), nel febbraio 2018 Github.

Capitolo 1

Prefazione

1.1 Norme seguite

Per famigliarizzare il lettore alla lettura di questo documento, partiamo con la definizione delle varie norme che seguiremo.

1.1.1 Collegamenti ipertestuali

Ogni collegamento che rimanda a citazioni, figure, tabelle, immagini, note a piè di pagina, sono contraddistinti da un numero di identificazione. Tale numero, in caso di visualizzazione tramite un software per la lettura di documenti *.pdf*, è possibile cliccarlo per andare direttamente al punto interessato dal riferimento. Unica eccezione è rappresentata da quelle parti di testo che fanno riferimento a collegamenti esterni (link), in questo caso, prendendo spunto dalla codifica HTML, abbiamo scelto di identificarli tramite la colorazione blu del testo.

1.1.2 Lessico

Alcuni termini saranno esposti con il loro nome "originale" in inglese. Principalmente la lingua scelta per la divulgazione di ricerche a livello mondiale è appunto l'inglese, di conseguenza, per il confronto di un proprio lavoro con tali ricerche, è necessario adoperare tali termini senza modifiche. Nonostante il seguente testo è pensato per il pubblico a livello nazionale, risulta dunque utile, per i motivi citati, adoperare in parte, il lessico anglosassone.

1.1.3 Acronimi, note e precisazioni

Seguendo, per così dire, la tradizione del campo militare, anche questo settore, fa uso massivo di acronimi per identificare varie tecnologie/attori/metodologie. Il significato di ogni acronimo è possibile reperirlo nell'apposita sezione 9. Nel caso invece di precisazioni e note, utilizzeremo lo strumento delle note a piè di pagina[1], oppure, se di maggiore importanza, ricorreremo all'uso dei riquadri:

Nota: Esempio di una nota

Non essendoci altre indicazioni riguardo, vi auguriamo una buona lettura.

1.2 Struttura del testo

Il documento verrà suddiviso in capitoli e sottocapitoli, seguendo una struttura simile a quella di un libro. Tralasciando il primo capitolo , troviamo il **Capitolo 2**, l'Introduzione, dove vengono esposte le sfide che offrono la tematica rilegata alla individuazione degli attacchi DoS/DDoS, includendo

[1] questa è una nota.

nell'argomento, la descrizione delle diverse famiglie degli IDS e le loro posizioni, con la differenza fra gli attacchi di tipo distriubuted (distribuiti) e non. Proseguendo nel **Capitolo 3**, "Datasets", vengono analizzati i lavori pubblicati rilevanti, fra il 2015 ed il 2020, con un'attenzione rivolta nei confronti dei datasets utilizzati e le metodologie usate; critiche e note da parte dell'autore sono inoltre presenti in alcune di queste pubblicazioni analizzate. Chiarita l'importanza ricoperta dal ruolo dei dataset, nel **Capitolo 4**, viene presentato l'omonimo "CIC-DDoS2019 Dataset" (ritenuto da noi come il più recente e pertinente in questo campo di ricerca), dove offriamo una spiegazione sommaria degli attacchi presenti all'interno dello stesso. Inoltrandoci nel corpo della ricerca, nel **Capitolo 5**, sottolineiamo le differenze fra le tecniche basate sul ML e DL, offriamo al lettore poi, gli strumenti necessari per comprendere le metriche utilizzate, per giudicare la bontà di individuazione di un generico modello. Sempre all'interno dello stesso capitolo, troviamo gli applicativi software più popolari per l'addestramento, costruzione e rilascio di modelli ML/DL, in aggiunta alla sezione di presentazione sugli algoritmi usati con maggior successo, nell'individuazione di attacchi maligni. Concludendo, nello stesso capitolo, riorganizziamo tutte le problematiche e critiche fatte precedentemente, chiudendo il discorso.
Chiarito ora l'ambiente che circonda gli attacchi DoS/DDoS, vengono esposte le varie proposte (**Capitolo 6**), per tentare di risolvere le mancanze rilevate precedentemente. Conclusione del testo è fatta nel **Capitolo 7**, dove riassumiamo e chiudiamo la discussione, lasciando (si spera) il lettore con nuovi spunti su questo tema intricato e complesso.

1.3 Conflitto di interessi

Trattandosi di un lavoro per il compimento della Laurea triennale, e lavoratore presso un settore non inerente a questo discusso, l'autore dichiara l'assenza di qualunque conflitto di interessi.

Capitolo 2

Introduzione

Dal momento della sua nascita fino ad oggi, il numero di utenti che utilizzano internet non ha fatto altro che crescere, d'altra parte, la pazienza nei riguardi dei servizi offerti sembra calare. In un contesto del genere, la disponibilità e la prontezza diventano delle qualità fondamentali da ottenere e preservare per i servizi online, rendendo gli attacchi DoS[1] e DDoS una seria minaccia. Queste tipologie di attacchi informatici hanno infatti come obiettivo, rendere irraggiungibile o inutilizzabile il servizio e/o le risorse di uno o più server, azione accompagnata, fra gli effetti principali, da un danno alla reputazione ed un mancato guadagno per l'attore target[1].

2.1 Motivazioni

Lo spazio di ricerca delle soluzioni alla mitigazione/eliminazione di questi attacchi, è caratterizzato da un uso importante di algoritmi di machine learning (ML) e/o deep learning (DL). Le ragioni di questa scelta[2] è dettata da plurime motivazioni, che andiamo ora ad esporre.
Assegnando una cardinalità, come prima ragione, abbiamo la caratterizzazione del problema a possedere una molteplicità di modi, o vettori di attacco, disponibili agli attori malintenzionati, con cui assaltare il sistema da proteggere. Basta osservare, ad esempio, la suddivisione rappresentata nella figura 2.1, proposta dall'istituto Canadese per la Sicurezza Informatica (CIC), su una tassonomia degli attacchi DDoS.[3] (parliamo di una poiché vi sono diverse filosofie per la classificazione), per renderci conto delle ramificazioni di questi attacchi.

Evitando tuttavia di scendere per il momento a questo livello di dettaglio, è possibile fare a monte una divisione più semplice e generalizzata rispetto a quella di Figura 2.1. Ricordando gli obbiettivi del DoS e DDoS, la suddivisione dell'intero ventaglio offensivo, può essere scissa in sole quattro categorie:[2]

- Consumo delle risorse del Server
- Consumo della banda della rete
- Crash del server sfruttando vulnerabilità presenti sullo stesso
- Spoofing dei pacchetti

Seconda ragione: essendo una (se non la) delle più comuni tipologie di attacco informatico [4], è affetta da continue evoluzioni, rendendo l'attività di individuazione, un compito arduo, pieno di sfide, e al contempo dinamico. Tenendo inoltre in conto il fatto che gli attori maggiormente affetti da questi attacchi, possiedono generalmente medie-grandi dimensioni come rete locale, può risultare essere necessario, analizzare una grande quantità dati di traffico rete, mantenendo al contempo, in limiti accettabili, la qualità del servizio per l'utente finale[3]. Il rapporto che si instaura, come in molti aspetti della sicurezza cibernetica, fra attaccanti e difensori, è quello di una corsa

[1] DoS è appunto acronimo di Denial of Service, in italiano negazione del servizio.
[2] Ovvero di fare utilizzo di tali algoritmi.
[3] Non possiamo ad esempio nel processo di analisi, rallentare eccessivamente la connessione dati dell'utente.

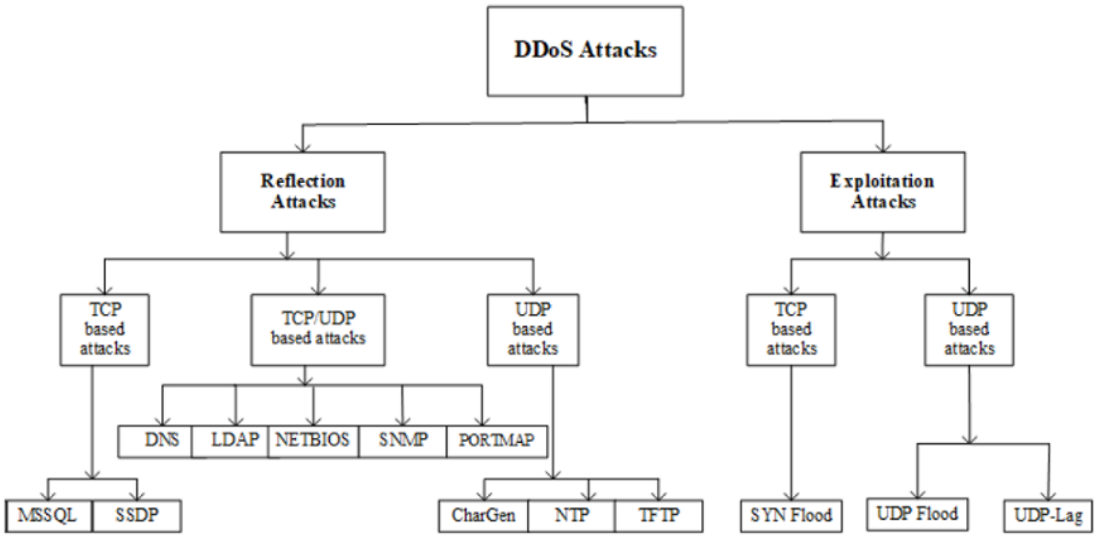


Figura 2.1: Tassonomia degli attacchi DDoS secondo il Canadian Institute for Cybersecurity

d'armi[5], dove il costante aggiornamento, e lavoro di ricerca, costituiscono dei requisiti di vitale importanza per la mitigazione del problema.[4] Per queste ragioni, in aggiunta all'utilizzo sempre più predominante in ogni campo dell'apprendimento automatico ed intelligenza artificiale, hanno portato a molti esperti nel campo alla ricerca di soluzioni facenti uso di ML e DL. Tali metodologie sono infatti adatte all'analisi di grandi mole di dati, all'individuazione di patterns difficilmente riconoscibili da un utente umano esperto[6] e capaci di definire ed impiegare nuove regole per il contrasto di sconosciute tecniche novelle.[5]

2.2 IDS

Conoscendo ora le ragioni che hanno portato alla scelta di metodologie basate sul ML/DL, andiamo ad individuare e discutere l'elemento, la figura, che ha il compito di proteggere il sistema da eventuali attacchi, l'IDS.

2.2.1 Introduzione

L'attore che si occupa di difendere la rete, viene solitamente chiamato Sistema per l'Individuazione di Intrusioni, o meglio, in inglese, Intrusion Detection System (IDS)[6]. La domanda ora da porsi è: ma cosa è un IDS? Con questo termine viene identificata quell'entità il cui compito è di individuare intrusioni in un determinato sistema. Dove per intrusioni si intende qualunque accesso non autorizzato al nostro sistema da proteggere, che essa sia una rete LAN, un determinato dispositivo, etc. A titolo informativo, consultando la letteratura (e come accennato nelle note a piè di pagina), è possibile anche riscontrare, per descrivere tale attore, il termine IPS (Intrusion Prevention System), la differenza sostanziale che è presente fra i due sistemi è suggerita dal nome: gli IDS agiscono una volta che individuano un intrusione, mentre gli IPS cercano di prevenirla. Per fare un'analogia, se il lettore ha esperienze lavorative, gli IDS sono similari ai DPI, svolgono un'attività di protezione nei confronti dei dipendenti (il nostro sistema da proteggere), partendo dal presupposto di avere sempre una presenza di un rischio. Gli IDS invece, sono analoghi a tutte

[4] Un buon riferimento per la fase preliminare di documentazione su nuove procedure individuate, è offerto dalla lettura della sezione *Alerts* e/o *Bulletins* della CISA.

[5] questa caratteristica è interessante per contrastare il problema dello zero-day detection

[6] Se oltre all'individuazione vi è anche un lavoro di prevenzione, prende il nome di Intrusion Detection and Prevention System (IDPS)

le attività di prevenzione adottate dal datore di lavoro per ridurre talle rischio.

Bisogna notare che i DoS e DDoS rappresentano solo una categoria di intrusione, mentre gli IDS, dalla definizione data precedentemente, devono essere capaci di ricoprirle tutte. Più precisamente, tutte le intrusioni possono essere categorizzate in quattro gruppi[97]:

- DoS e DDoS
- U2R: negli attacchi User to Root[7] l'attaccante sfrutta qualche debolezza, mancanza, caratteristica del sistema, per passare da privilegi amministrativi a quelli di root [17].
- R2L: Remote to Local o anche Remote to User attack, in questo caso, l'attaccante sfrutta qualche vulnerabilità della macchina della vittima che è connessa alla rete interessata, per poter acquisire un accesso autorizzato a quest'ultima. Se poi la vittima ha privilegi amministrativi è possibile seguire R2L con un attacco U2R.
- Probe: Sonda, di conseguenza un attacco mirato al sondaggio della rete interessata. Più precisamente, qualunque metodologia il cui scopo finale è l'acquisizione di informazioni pertinenti alla rete target. Questi tipi d'attacco, come indicato dal nome, sono di natura ricognitiva, di preparazione, antecedenti ad altri tipi di attacco.

Nel nostro caso, fortunatamente, l'interesse è centrato solamente ad una sola tipologia d'attacco: DoS e DDoS.

2.2.2 Famiglie

Ad oggi, è possibile suddividere gli IDS in quattro metodologie (o famiglie), una basata sulla cosiddetta firma (**signature-based**), una sull'analisi del traffico (**anomaly-based, behavior-based**), una su profili predeterminati (**Stateful Protocol Analysis**), ed infine una che impiega signature-based e behavior-based in parallelo fra di loro (**hybrid**)[138], ognuna di queste, come in ogni aspetto della vita, ha i propri pro e contro.
Possiamo riscontrare, in lavori svolti da altri ricercatori, un numero diverso da noi riportato. In ogni caso, le metodologie riconosciute sono sempre 2, 3, o infine 4. Se la divisione è binaria, le metodologie sono la **signature-based** e la **anomaly-based, behavior-based**, un esempio è rappresentato dall'articolo di revisione curato da Mohammed falih badran et al[8]. Nel caso di una divisione ternaria, in aggiunta alle due citate precedentemente si aggiunge l'**hybrid**, per citare un esempio anche in questo caso, è possibile documentarsi con il lavoro svolto da Mohammad M. Shurman, Rami M. Khrais, Abdulrahman A. Yateem[9].

2.2.3 Signature-based

Sistemi che controllano la firma sono precisi, veloci e con un numero di falsi positivi basso, tuttavia non possono sopperire ad una caratteristica fondamentale dei DoS descritta precedentemente, ovvero la continua innovazione e mutazione; se non si conosce la firma il traffico è considerato come benigno. Per un sistema del genere è di conseguenza di fondamentale importanza la continua attività di aggiornamento e ricerca sulle ultime e nuove modalità adottate dagli attori malintenzionati.

2.2.4 Anomaly-based, chiamata anche behavior-based

L'analisi del traffico, di controparte, invece di osservare le firme, esegue una sorta di profilazione degli utenti/traffico. Nella fase di training usa come campioni, traffico definito dall'utente come benigno, standard, normale. Nella fase di detection il traffico viene messo a confronto con i profili che si sono generati dallo studio svolto nella fase di testing; l'utilizzo di queste pratiche fa si che sistemi basati sull'analisi del traffico siano veloci e dinamici nell'individuazione di nuovi attacchi sconosciuti [14]. La dinamicità del sistema rappresenta però un'arma a doppio taglio; lo svantaggio

[7] in inglese, foneticamente, la parola *to* e *two*, sono molto simili, non è quindi raro, nella creazione di acronimi l'utilizzo del 2 al posto del *to*.

più grande ora è rappresentato dal grande numero di falsi positivi che vengono creati[15]. L'implicazione di questo svantaggio è simile alla morale della favola *"Al lupo! Al lupo"*: quando vi sono tanti falsi positivi, l'attenzione posta su eventuali veri positivi cala drasticamente.

2.2.5 Hybrid

Effettuando un'unione delle due precedenti metodologie, è possibile ottenerne una terza, quest'ultima prende il nome di hybrid. L'idea alla base è combinare le ottime prestazioni ottenibili tramite un database contenente le firme conosciute, con la dinamicità e prontezza al contrasto di nuovi attacchi, caratteristiche principali delle metodologie **anomaly-based**. Anche qui, come in molti aspetti ingegneristici, vi sono varie scelte disponibili sul come implementare tale modello, per fornire un esempio, un architettura possibile è rappresentata nella figura 2.2, adottata in simile maniera da [16], [2] e [18].

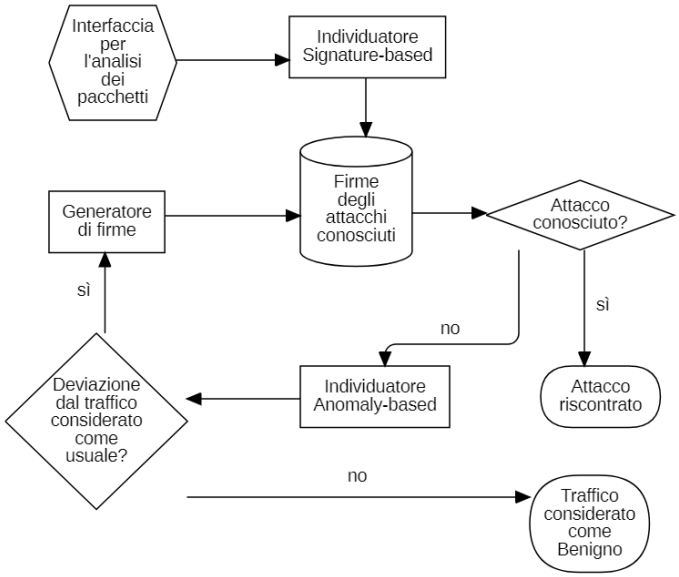


Figura 2.2: Schema a "pipeline" per la metodologia hybrid

2.2.6 Protocol Analysis

Fra le quattro, questa risulta essere la meno utilizzata, il funzionamento si basa sul confronto tra profili predeterminati e traffico attuale. I profili predeterminati vengono costruiti seguendo le definizioni di attività dei protocolli, considerate come benigne. La politica con cui viene deciso se un'attività del protocollo è di tipo benigno o meno, è dettata dalla sua conformità alle norme/guide/direzioni date dagli stessi sviluppatori dei protocolli. Questa politica deriva dall'osservazione che una buona parte di attacchi, sfrutta la creazione di pacchetti che violano norme/indicazioni descritte nelle RFC.
Una volta creati i profili, avviene un confronto con il traffico attuale, ricercando deviazioni [20]. Attraverso l'analisi dei protocolli utilizzati, queste metodologie riescono a restringere il campo di ricerca dell'attacco in corso, il quale porta come conseguenza benefica, alla riduzione del numero di falsi positivi[19]. Come svantaggio principale, troviamo l'uso intensivo di risorse richiesto dall'analisi e tracciamento degli stati per sessioni multiple, e l'incapacità di individuare attacchi che non violano le pratiche da seguire e non, dei protocolli[20].
Un sunto della caratteristiche appena descritte si può trovare alla tabella 2.1.

Metodologie usate dagli IDS				
	Signature-based	**Behavior-based**	**Stateful Protocol Analysis**	**Hybrid**
Performance della individuazione	Basso numero di falsi positivi	Alto numero di falsi positivi	Per definizione, genera pochi falsi positivi, si basa su standard dettati dagli sviluppatori dei protocolli di comunicazione	Sostenuto numero di falsi positivi
Capacità di Individuazione di Attacchi nuovi	Nessuna	Sì	Nessuna	Sì
Vantaggi principali	Pochi falsi positivi e velocità del sistema	Capacità di individuare possibili nuove tipologie di attacco	Regole definite e chiare, in aggiunta ad un numero ridotto di falsi positivi rispetto alle signature-based.	Combina i vantaggi degli anomaly e signature-based IDS
Svantaggi principali	Nessuna capacità di individuazione di nuovi tipi di attacco, forte dipendenza dal dataset su cui è stato eseguito il training	Alto numero di falsi positivi, difficoltà nel definire con precisione l'uso "abituale"	Uso intensivo di risorse, incapacità di individuare nuovi tipi di attacco	Velocità e manutenzione alta per controllare che i falsi positivi derivanti dalla parte anomaly-based, non crei firme fittizie

Tabella 2.1: Tabella riassuntiva sulle metodologie usate dagli IDS

2.3 Confini poco definiti con le Tassonomie

Sembra necessario ricordare che per la classificazione degli attacchi al momento conosciuti, è possibile utilizzare varie metodologie e criteri. Questa libertà in aggiunta a delle definizioni più o meno ampie, porta a dei confini molto variabili, conseguenza figlia di ciò, è la possibile presenza di confini per così dire 'sfumati' (fuzzy), non ben definiti, fra categorie. Proponiamo un esempio: nella tassonomia esposta precedentemente dal CIC fig 2.1, vi è una divisione in attacchi riflessivi e attacchi di sfruttamento (Exploitation), in questa suddivisione l'UDP Flood appartiene alla seconda categoria, come del resto il SYN flood. Lo stesso attacco, nel lavoro svolto da Dharmadhikari, Chinmay, et al.[83], è invece categorizzato come attacco volumetrico, il quale non comprende il SYN flood. Questo perché, come possibile osservare nella figura 2.3, la distinzione è stata effettuata fra il piano dei dati ed il piano del controllo.

L'importante in questo caso, è rendersi conto che non ci sono scelte sbagliate, o per meglio dire, è compito dei ricercatori usare o creare delle tassonomie ottimali, in base alla domanda di ricerca.

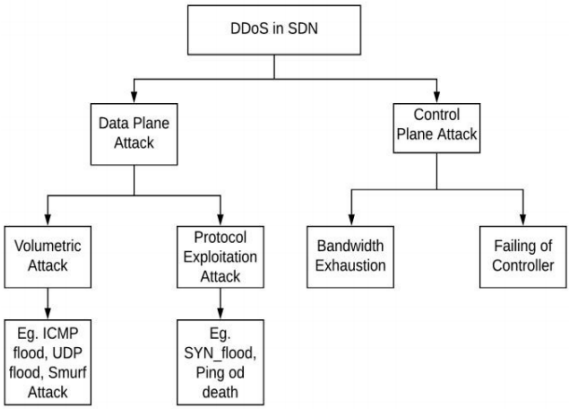


Figura 2.3: Tassonomia degli attacchi DDoS secondo Dharmadhikari, Chinmay[83].

2.4 I malware, e il pericolo dei device IoT

Un importante lezione è possibile trarre dall'attacco del 2016 effettuato dalla botnet costruita grazie al malware Mirai, per poterne parlare però è necessario sottolineare la differenza che intercorre fra un attacco DoS e uno di tipo DDoS. Si parla di attacchi DoS quando la macchina attaccante è singola [21](o comunque riconducibile ad un singola rete), mentre si utilizza il termine DDoS se sono multiple. In DDoS, la prima "D" rappresenta la caratteristica "Distributed", distribuito perché appunto il traffico di attacco è generato da molteplici dispositivi, dislocati sia geograficamente, sia a livello di rete (indirizzi IP diversi). Il grande numero di dispositivi IoT in aggiunta a varie problematiche legate alla sicurezza, hanno reso le botnets delle armi molto efficaci disponibili agli attori malintenzionati [22][8].

2.4.1 Botnets

Una, o più, botnet è una rete composta da macchine infettate da malware e a loro volta, collegate alla rete internet pubblica. I principali attori coinvolti in una botnet sono tre: le macchine infettate dal malware (i cosiddetti bot[9]), la macchina (o macchine) responsabile per il controllo e comando dei bot (chiamata Command and Control Infrasrtucture), ed infine l'utente malintenzionato che decide il tipo di attività illegale da conseguire mediante la botnet (chiamato botmaster) Figura 2.4. Possiamo ora discutere del malware Linux chimato *Mirai*[23], che come detto poco fa, ha contribuito all'esecuzione dell'attacco massivo avvenuto nel 2016. Scritto in *C*, l'obiettivo principale di questo, può essere scomposto in tre fasi:

- Ricerca: vengono cercati routers e telecamere IP disponibili in rete, attraverso una continua scansione su varie porte (come la 22,23,5747,etc.)
- Accesso/Login: una volta trovato un dispositivo, tenta di accedere ad esso, eseguendo attacchi di forza bruta attraverso l'utilizzo di un dizionario. Se il lettore non ha idea di cosa si tratta, la pratica consiste nel provare tutte le combinazioni di password e nome utente (la parte di forza bruta dell'attacco), conosciute, di default (il dizionario). Ad esempio, ipotizziamo banalmente di voler entrare in un router domestico, se l'utente non ha effettuato alcuna modifica alle credenziali d'accesso, un utente malintenzionato potrebbe provare ad accedervi inserendo coppie di credenziali utilizzate frequentemente di default, contenute appunto in un dizionario. All'interno dello stesso, in questo caso, vi sono sicuramente le seguenti coppie di credenziali: ***Nome utente:*** *Admin*, ***password:*** *admin*, come anche: ***Nome utente:*** *Admin*, ***password:*** *password* e molte altre.

[8] La maggior parte dei dispositivi che possono diventare bot/zombie, sono vulnerabili o per una errata configurazione da parte dell'utente finale, o per aggiornamenti software non eseguiti. Con configurazione errata, intendiamo anche il mancato cambiamento delle credenziali di default dei vari dispositivi, un aspetto molte volte tralasciato dall'utente medio, per una svalutazione sul rischio associato.

[9] alcune volte, come ad esempio dalla cisco, vengono anche chiamati *zombie*

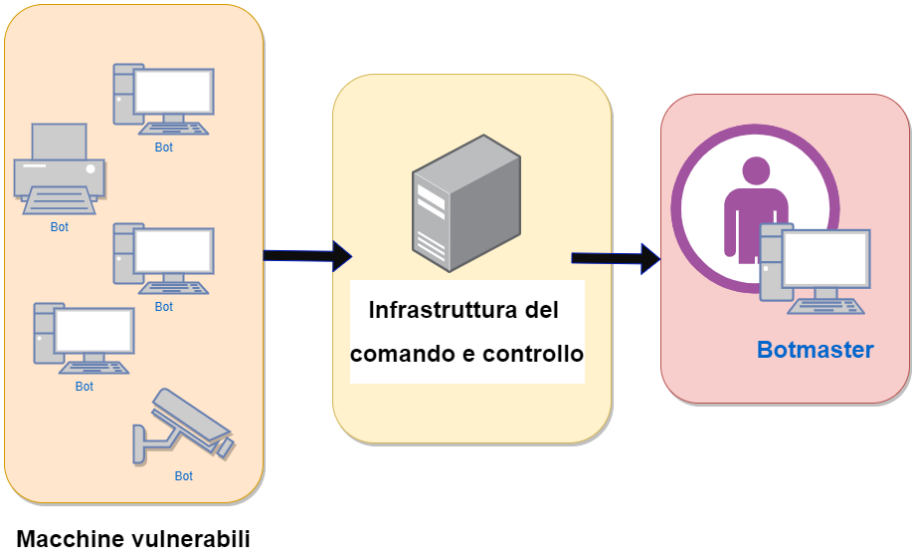


Figura 2.4: Componenti principali di una botnet

- Ricerca da parte del device: il device infettato cercherà a sua volta altri dispositivi IoT presenti nella rete.

Importante sottolineare che l'attacco eseguito nel 2016 tramite Mirai risulta essere il più grande attacco DDoS mai visto (1.2 Terabits al secondo).

Nota: Forza bruta, ma non proprio:
Solitamente quando si parla di metodologie basate sulla forza bruta della potenza e velocità di calcolo dei computers, si immagina numeri elevati, tuttavia, in questo caso, il dizionario usato per entrare nei device era composto solo da 50 termini[24].

2.5 Residenza del detettore

Una domanda che può essere sorta spontaneamente al lettore, arrivati a questo punto nella discussione, potrebbe essere: "In che luogo però è meglio porre l'IDS, a pari livello con gli ISP? o ai router estremi della nostra rete?". Cerchiamo di rispondere a questo quesito, analizzando le diverse soluzioni presenti nella letteratura corrente. Presentiamo a questo scopo alcune architetture di rete riscontrabili tutt'oggi.

2.5.1 Architetture delle reti

Sostanzialmente, osservando la Figura 2.5 e la Figura 3.1, le scelte dove implementare l'IDS si riducono a 4 componenti: i dispositivi finali (PC, smarthphones, apparecchi IoT), gli switch, i routers, o le macchine responsabili del controllo della rete (controllori SDN). Più precisamente, 3 componenti, poiché non sono presenti in letteratura (a nostra conoscenza) soluzioni basate sul ML adottate ai dispositivi finali, per il nostro scopo. Ogni punto presenta delle caratteristiche uniche con relativi pro e contro, incominciamo ponendoci al livello del gateway, e prendiamo come esempio la ricerca condotta da Rohan Doshi, et al.[22]. Qui viene suggerita la possibilità di utilizzare algoritmi di ML a basso consumo computazionale, eseguibili perfino su routers domestici (e altre middleboxes[10]). Fra le caratteristiche salienti abbiamo l'abilità all'individuazione, fin dal principio, di attacchi generati da device IoT facenti parti di una botnet, con il grande vantaggio di richiedere poche risorse. Sfortunatamente, questo approccio presuppone l'aggiornamento di una

[10] RFC 3234, vedi glossario

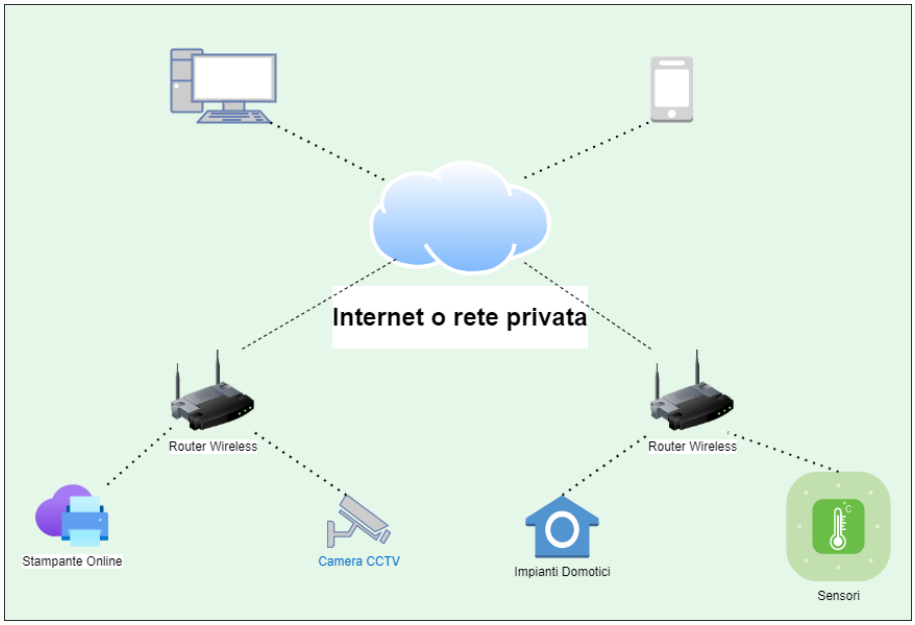


Figura 2.5: Un esempio di rete classica Wireless con dispositivi IoT

quantità elevata di middleboxes per l'installazione degli algoritmi ML, in aggiunta alla necessità dell'adozione di questa metodologia da tutte le aziende produttrici. Richiedendo dunque un'azione al di fuori dell'utente che si vuole proteggere, il grado di sicurezza sarà di conseguenza legato a quanti devices adotterebbero questa policy. Inoltre, anche ipotizzando la risoluzione di questi problemi logistici, e verificata l'effettiva applicabilità e funzionalità in campo di tali algoritmi, la copertura risulterebbe garantita solamente alla fetta degli attacchi DDoS, che utilizzano come bot, dei device IoT.
Escluse due delle quattro categorie, possiamo affermare che le strategie più diffuse nella letteratura odierna, rivolgono sulle ultime due possibili scelte: i controllori delle SDN, ed sul gateway/firewall (chiamato "router principale" nella Figura 3.1).

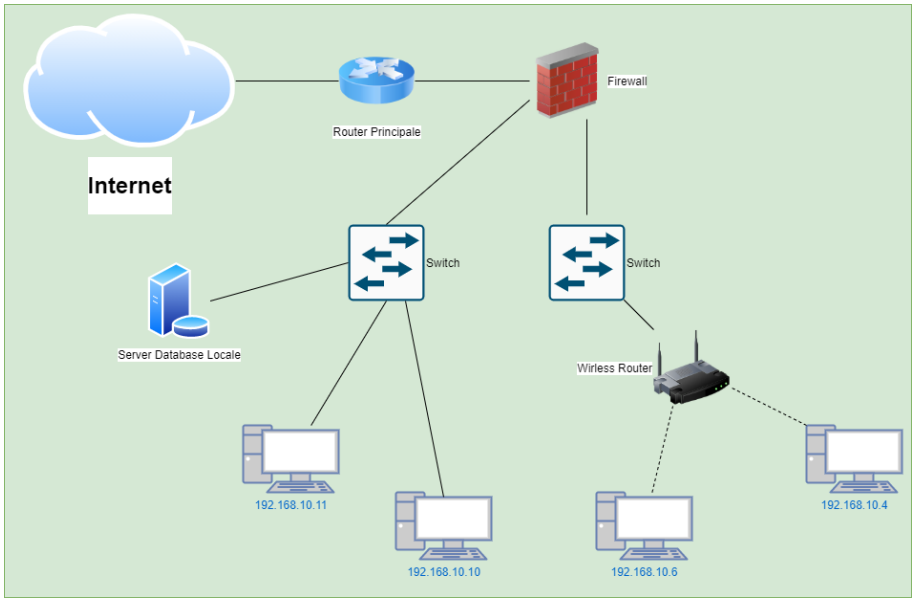


Figura 2.6: Un esempio di rete classica Aziendale

Capitolo 3

Datasets

Punto di partenza per gli IDPS e IDS che si basano sulla firma, è l'utilizzo di un dataset aggiornato, completo e rappresentativo della realtà. La qualità del database è infatti di fondamentale importanza per questa categoria di individuazione, essendo la base del training. I lavori consultati ricadono fra il 2015 ed il 2020, la scelta di questa finestra temporale è dettata da una caratteristica, già antecedentemente citata[1], degli attacchi DoS e DDoS, ovvero la continua evoluzione. In questi termini, ricerche al di fuori da questa finestra temporale, anche se potrebbero risultare ancora in qualche maniera validi e/o pertinenti, non verranno citati.

3.1 Lavori correlati

Discutiamo alcuni dei lavori svolti da altri ricercatori, studenti e professionisti nel campo, per inquadrare meglio l'ambiente circostante alla ricerca e soluzioni proposte ad oggi. Per ogni lavoro citato esponiamo ed argomentiamo delle critiche e/o osservazioni, con lo scopo finale di definire delle regole generali per la scelta del dataset adatto.

3.1.1 Ids using machine learning current state of art and future directions (2015)[97]

Hamid, Yasir, M. Sugumaran, e V. R. Balasaraswathi nel documento redatto per la fine del 2015[97], raccolgono 68 lavori svolti da vari ricercatori nello sviluppo di *IDS* con machine leraning, offrendo un ottimo sunto esposto in forma tabellare. Gli autori descrivono tre principali dataset per la valutazione dell'efficacia di nuovi IDS. Questi sono:

- KDDCup99 dataset
- Corrected KDDCup99 dataset
- 10% KDDCup99 dataset

Secondo le ricerche condotte dagli autori, e da molti altri[96][101][98], *KDDCup99 dataset*[2] costituisce il dataset più utilizzato per la individuazione di intrusioni. Premesso che il lavoro svolto dalla Lincon Laboratory sia notevole, non è certo assente da forti critiche[99][100]. Ad esempio, J. McHugh.[98], sottolinea il fatto che il dataset sintetico è irrealistico, affetto da notevoli deviazioni rispetto a tracce di traffico riscontrabili realmente sul campo[3], mentre Amjad M. Al Tobi e Ishbel Duncan[99], dopo un'attenta analisi del dataset e delle critiche al riguardo, concludono con la dichiarazione dell'inutilizzabilità del dataset in qualunque nuovo modello. Altra problematica sottolineata, è l'assenza di trasparenza su alcuni punti al quanto fondamentali. Ad esempio,

[1] sezione 2.1

[2] Per il loro set di dati target, utilizzato il dataset 1998 DARPA/Lincoln Labs[101]

[3] Come descrive l'autore, un flusso di dati realistico, è composto anche da traffico "strano" ma pur sempre legittimo (ad esempio tempeste di pacchetti FIN e RST, pacchetti frammentati ma che hanno il flag don't fragment settato, ecc.), traffico che risulta assente nel dataset.

il processo usato per generare il flusso dati è solo superficialmente descritto, nessuna validazione analitica o sperimentale sembra essere stata eseguita sul traffico di background (benigno), sulle caratteristiche dei falsi allarmi. Altre osservazioni importanti da considerare: essendo un lavoro svolto nel 1999 è da considerarsi obsoleto e non più adatto ad assumere il ruolo di benchmark che aveva assunto negli anni passati.
Nel caso del *Corrected KDDCup99 dataset*, possiamo notare che viene utilizzato solo da due ricerche:

- Intrusion Detection Using Fuzzy Association Rules [131]
- Using particle swarm optimization in fuzzy association rules-based feature selection and fuzzy ARTMAP-based attack recognition[132]

Entrambi i lavori utilizzano per la fase di training, *10%KDD dataset*, mentre per la fase di testing, *Corrected KDDCup99 dataset*. Il problema maggiore che ci sentiamo di sottolineare è l'assenza di documentazione del dataset usato per il testing. Per quanto ci è stato possibile verificare, non abbiamo trovato una soddisfacente descrizione nè nei lavori in cui viene utilizzato, nè in altri lavori svolti. L'unica descrizione offerta è la presunta assenza degli attacchi ridondanti presenti nel *KDDCup99 dataset*. Anche se non mettiamo in discussione la veridicità dell'affermazione, non è distante dalla realtà pensare che possa presentare le stesse problematiche individuate da J. McHugh.[98](dato che è figlio del *KDDCup99 dataset*. Un simile discorso può essere effettuato per il *10% KDDCup99 dataset*, il quale anche se più utilizzato rispetto al precedente, risulta essere (come indicato dal nome), una riduzione del dataset padre. Conseguentemente a ciò, le mancanze principali, esposte da J. McHugh, sono da ritenersi ancora valide.

3.1.2 A DDoS Attack Detection Method Based on Hybrid Heterogeneous Multiclassifier Ensemble Learning (2016) [27]

Invece di utilizzare un singolo classificatore, perché non utilizzarne una molteplicità? Una domanda dallo stesso significato se lo sono posti i ricercatori di questo lavoro, dove propongono un modello basato sulla tecnica dell'Heterogeneous Ensemble Learning. Sinteticamente, per cercare di aumentare la capacità di generalizzazione al modello finale, si uniscono[4] vari classificatori diversi in natura fra di loro, o meglio eterogenei (Heterogeneous appunto). Nonostante anche questo documento sia relativamente recente (del 2016), gli autori scelgono di utilizzare come dataset per la validazione del loro modello, l'obsoleto *Corrected KDDCup99 dataset*, o più precisamente un subset contenente il 10% di esso[5]. Difficile non notare inoltre, la mancanza di metriche sul tempo di esecuzione del modello proposto; suddivisibile in tre componenti, questo è formato da una componente correlata alla preparazione dei pacchetti del traffico di rete, una contenente i vari classificatori, ed infine una che effettua la loro unione. Ipotizzando che la procedura di preparazione non richieda un tempo eccessivo, quella sulla classificazione mediante i molteplici classificatori potrebbe comportare all'introduzione di ritardi non più ammissibili per il contesto reale, preoccupazione non priva di fondamento.

3.1.3 Comparison Deep Learning Method to Traditional Methods Using for Network Intrusion Detection (2016) [25]

Un altro spunto di ricerca da considerare, è offerto dagli algoritmi di Deep Learning (DL). In questo documento si pongono come obiettivo, il confronto fra i metodi cosidetti "tradizionali", e gli algoritmi di Deep Learning. I metodi tradizionali sottoposti al confronto sono: *Decision Tree (chiamato C4.5 nel documento in questione)*, *Naïve Bayes*, e *Support Vector Machine (SVM)*, mentre per la metodologia DL: *SVM-RBMS*.
Nonostante le buone intenzioni degli autori, non possiamo far altro che notare una serie problematiche; la prima delle quali, l'utilizzo dell'obsoleto e criticato dataset *KDDCup99* (capitolo 3.1.1), e più precisamente, il subset *10% KDDCup99 dataset*. Altra problematica risiede nelle precisioni riportate sui metodi "tradizionali", dato che ha noi interessa solamente il campo degli attacchi DoS,

[4] Ensemble, significa unione, insieme, in questa istanza l'unione è effettuata tramite un voto di maggioranza.
[5] Le plurime problematiche correlate al datset in questione, sono state discusse nel capitolo 3.1.1

ci concentreremo sulla *figura 4 del documento in questione*, dove vengono riportate le precisioni dei vari algoritmi più quella del modello proposto nel documento (SVM-RBMS). Gli autori citano, nel caso dell'algoritmo *Decision Tree*, la "variante" proposta nel 1993 da J. Ross Quinlan chiamata *C4.5*, ed una precisione (nel migliore dei casi) nell'individuare attacchi DoS del circa 60%. Invece, se prendiamo in esame la ricerca [10] del 2004 (ovvero 12 anni antecedenti rispetto al documento in esame), le percentuali ottenute in precisione, utilizzando sempre *C4.5*, e il subset composto dal 10% di *KDDCup99*, si attestano all'incirca al 95%[6], superando perfino la precisione ottenuta dal modello più performante di [25]. Per questi motivi, i risultati ottenuti, non verranno considerati come attendibili nel resto del testo.

3.1.4 Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey (2019)[31]

In questo documento i ricercatori raccolgono e discutono i vari vantaggi e svantaggi di diverse ricerche pubblicate fra il 2015 e il 2019, proponendo inoltre una nuova tassonomia, sottolineando l'importanza degli algoritmi di Deep Learning. Oltre ai classici *DARPA1998* e *KDD99*, una nuova aggiunta nella discussione riguardante ai dataset di benchmark, è esposta da Liu, Hongyu, e Bo Lang: *UNSW-NB15*. Nella stessa discussione i ricercatori espongono chiaramente i problemi di *DARPA1998* e *KDD99*, ribadendo in più occasioni, la necessità di costruire un nuovo dataset pubblico da utilizzare come benchmark per IDS basati su tecniche di machine learning. Oltre a ciò, viene osservata una chiara nuova tendenza della ricerca, su metodologie basate sul deep learning, dei 26 documenti consultati, 14 infatti ne fanno uso.

3.1.5 A Survey of Network-based Intrusion Detection Data Sets (2019) [129]

Un ottimo lavoro sulle caratteristiche dei dataset disponibili[7], è stato condotto da Ring, Markus, et al. [129]. In questa ricerca gli autori corredano 34 dataset datati fra il 1998 e il 2017, con proprietà rilevanti[8], come anno di pubblicazione, tipi di attacco, tipo di ambiente usato, dimensione, etc. offrendo ai lettori una guida efficace alla scelta dei datasets più appropriati[9] per la propria ricerca. La spiegazione dettagliata delle caratteristiche utilizzate per la classificazione, l'attenzione posta sulla necessità del continuo aggiornamento dei datasets e la necessità di subsets già definiti per confrontare correttamente lavori di diversi ricercatori[10], rendono questo documento, un'eccellente punto di partenza per le future ricerche nel campo.
Per completezza, ci sentiamo di aggiungere alla tabella III, seguendo le stesse metodologie degli autori, i seguenti recenti dataset: CIC-DDoS2019 e CSE-CIC-IDS2018 on AWS.

	Informazioni Generali				Natura dei Dati			Volume dei Dati		Ambiente per la registrazione			Valutazione		
Data Set	*Anno della crazione del traffico*	*Disponibile Pubblicamente*	*Traffico Normale*	*Traffico Maligno*	*Metadata*	*Formato*	*Anonimità*	*Numero*	*Durata*	*Tipo di Traffico*	*Tipo di Rete*	*Rete Completa*	*Sets predefiniti*	*Bilanciato*	*Classificato*
CIC-DdoS2019	2019	si	si	si	si	pacchetti, csv	no	965 pacchetti 193 GB – 18 file csv 31 GB	2 giorni	simulato	piccola rete	si	si	no	si
CSE-CIC-IDS2018 on AWS	2018	si	si	si	si	pacchetti, csv, altri	no	444 GB pacchetti	2 giorni	simulato	rete aziendale	si	no	no	si

Figura 3.1: Datasets Aggiunti seguendo il format di [129]

[6] *Tabella 3* di [10]
[7] per i NIDS
[8] sezione *IV. DATA SET PROPERTIES* di [129]
[9] Importante infatti sottolineare, che l'utilizzo di un singolo dataset è fortemente sconsigliato
[10] sezione *VII. OBSERVATIONS AND RECOMMENDATIONS*, sottosezione *Predefined Subsets*

3.1.6 DDoS datasets Use of machine learning to analyse intrusion detection performance (2020) [50]

L'obiettivo primario di questo documento redatto da Stefanos Kiourkoulis è l'analisi e la documentazione sulle performance dei vari dataset disponibili ai ricercatori. Infatti la disponibilità di dataset rappresentativi di traffico e attacchi reali, che risultano non obsoleti e corredata da una documentazione soddisfacente sulla topologia della rete, è riconosciuta dalla letteratura odierna, come una delle sfide principali nella costruzione di un IDS. L'autore sceglie in particolare quattro dataset, tutti sintetici (anche detti simulati), e di recente data (fra il 2017 ed il 2019) per soccombere al problema dell'obsolescenza. I dataset in questione sono:

- CICDDoS2019
- CSE-CIC-IDS2018 on AWS
- NDSec-1
- CICIDS2017

Per valutare l'efficacia dei vari dataset l'autore raccoglie tutte le informazioni presenti, effettua un lavoro di pre-processing dei dati e li divide in modo da avere tre sottogruppi: uno usato per il lavoro di training, uno per il testing ed infine uno per la validazione (validating). Nella fase di training vengono comparati sei diversi algoritmi: [11]

- k-nearest neighbour
- SVM
- naïve Bayes
- decision tree
- random forest
- logistic regression

Mentre per la validazione viene adoperato k-fold cross. L'autore conclude affermando che *CSE-CIC-IDS2018 on AWS* risulta essere il più performante fra gli altri dataset, con il 99% in accuracy-rate, F-mesaure e recall (i significati di questi termini saranno discussi nel capitolo 5.2.1).
Analizzando i dataset in questione e le dichiarazioni dell'autore al riguardo, sorgono dei dubbi su vari punti. Uno di questi è correlato al capitolo *"6.2.2 Volume and Class Distribution"*, vorremmo porre l'attenzione su due dataset in particolare, *CICDDoS2019* e *CSE-CIC-IDS2018*. Questo perché l'autore sostiene che il secondo sia più performante del primo, nonostante *CSE-CIC-IDS2018* risulta formulato dallo stesso istituto[12], antecedente all'altro, e comprendente un numero minore di tipi di attacchi Dos e DDoS.
Prendiamo in esame *CICDDoS2019*, in questo caso, l'autore individua 121,980 (41.4%) elementi classificati come traffico normale (benigno), e 172,647 (58.6%) elementi come attacchi, tuttavia, secondo un'altra ricerca[51], i numeri riscontrati sono 56,863 (benigno) e ben 50,006,249 (attacchi). Stesso discorso può essere effettuato per i numeri nel datset *CSE-CIC-IDS2018*. Qui l'autore individua 360,833 (34.5%) elementi classificati come traffico normale e 686,012 (65.5%) come attacchi, mentre, dalle nostre ricerche risulta: 9,176,239 di traffico benigno e 1,918,234 di traffico di attacco. La metodologia utilizzata per ricavare gli ultimi dati è descritta in dettaglio nella sezione 7.1, mentre le tabelle 3.1 e 3.2, presentano un sunto di quanto detto.

A fronte di queste discordanze non reputiamo attendibili le conclusioni a cui arriva l'autore. Per essere il più trasparenti possibili, un ulteriore lavoro di verifica, più dettagliato, dovrebbe essere effettuato per controllare se, anche con le mancanze sottolineate, le conclusioni risultino ancora valide.

[11] i modelli sono stati implementati con la libreria Python scikit-learn
[12] più precisamente, è una collaborazione con il Canadian Institute for Cybersecurity (CIC) e il Communications Security Establishment (CSE)

CICDDoS2019		CSE-CIC-IDS2018	
Benigno	**Attacco**	**Benigno**	**Attacco**
121,980	172,647	360,833	686,012

Tabella 3.1: Volume dei dataset dati descritti in [50]

CICDDoS2019		CSE-CIC-IDS2018	
Benigno	**Attacco**	**Benigno**	**Attacco**
56,863	50,006,249	9,176,239	1,918,234

Tabella 3.2: Volume dei dataset descritti in [51] e dalla nostra ricerca.

3.1.7 Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning (2019) [28]

Se si guarda negli anni passati fino ad oggi, il traffico di rete non ha fatto nient'altro che aumentare in dimensione ed essere sempre più variegato (nei tipi di protocolli usati)[30]. Non è dunque errato concludere che con l'aumento di dispositivi *IoT*, e lo sviluppo di nuove applicazioni, piattaforme online, questo trend continuerà a persistere. In un ambiente così definito, un *IDS* non può, realisticamente parlando, analizzare tutto il traffico per verificare la presenza di attacchi; anche se si dovesse solo mantenere gli headers di ciascun pacchetto, il carico di lavoro richiesto dalla macchina sarebbe eccessivo. Una possibile soluzione, che tiene conto di quanto detto, viene proposta nel lavoro di ricerca di origine brasiliana: *Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning*[28].
In questo lavoro gli autori adottano il protocollo *sFlow*[13] come campionatore per analizzare il traffico di rete. *sFlow* è uno standard industriale nel campo del monitoraggio del traffico di rete (nota a piè pagina 13), di conseguenza è molto utilizzato nei dispositivi attuali[28]. Tramite questo protocollo è possibile analizzare solo una parte del traffico, di fatto effettuando un campionamento, sgravando il carico di lavoro richiesto dalla macchina responsabile della individuazione. In un esperimento svolto dagli autori, si può osservare che con un traffico di rete di 81.3Mbps, il sistema di individuazione è stato soggetto a solamente 1.74Mbps (circa l'1.5% del traffico effettivo)[14], rendendolo dunque, secondo il loro parere, un approccio scalabile.
Dopo il campionamento del traffico, il quale include 5 variabili per ogni pacchetto, ne vengono derivate altre 28, la cui maggior parte di natura statistica. Le caratteristiche che definiscono se un determinato pacchetto è benigno o maligno, vengono scelte da *Recursive Feature Elimination with Cross-Validation (RFECV)* assieme ad un algoritmo di *ML*. Fra gli algoritmi di *ML* testati dagli autori possiamo trovare:

- Random Forest (RF)
- Decison Tree (DTree)
- Perceptron
- Stochastic Gradient Descent (SGD)
- Logistic Regression (LR)
- AdaBoost

Il più performante nel campo della precisione risulta essere *Random Forest*.
Delineiamo alcune note positive pertinenti a questo lavoro di ricerca. In primis gli autori, nel capitolo *4.1. Description of the Benchmark Datasets.* fanno presente dell'obsolescenza e l'inutilizzabilità dei seguenti dataset: *DARPA (Lincoln Laboratory 1998-99)*, *KDD'99 (University of California, Irvine 1998-99)*, e *LBNL (Lawrence Berkeley National Laboratory and ICSI2004-2005)*,

[13] Per maggiori informazioni al riguardo: sito per *sFlow*
[14] il parametro che controlla la frequenza di campionamento è comunque modificabile dall'utente, offrendo un maggior grado di controllo

a nostro parere, questa semplice azione rappresenta un buon indicatore di trasparenza da parte degli autori. Non è raro infatti riscontrare nella letteratura recente[15], lavori di ricerca che utilizzano dataset obsoleti, per la valutazione della metodologia proposta.
L'assenza di un dataset di benchmark realistico e attuale per attacchi *DoS* e *DDoS*, rappresenta una sfida ancora odierna per il testing di nuove metodologie di individuazione, sviluppate in campo accademico ed industriale. In questo caso gli autori, per sopperire a questa mancanza, utilizzano per il testing 3 dataset in aggiunta ad uno custom (fatto su misura): *CIC-DoS*, *CICIDS2017*, *CSE-CIC-IDS2018*. Tutti sviluppati totalmente o in parte dal *CIC* e di recente data (rispettivamente del 2017, 2017 e 2018).
Altra nota positiva, e sempre indice di trasparenza, scaturisce dalla sezione *Data Availability*. In questa piccola parte dell'articolo, gli autori rendono disponibile ai lettori il loro dataset custom e l'algoritmo utilizzato per la selezione delle caratteristiche, in aggiunta ai 3 dataset della CIC. Sfortunatamente, non è possibile vedere l'utilizzo di *CIC-DDoS2019 Dataset* poichè gli autori hanno completato e pubblicato il lavoro nei primi di giugno del 2019, mentre il dataset pubblicato dal *CIC*: è stato reso pubblico nei primi giorni di ottobre 2019.

[15] negli ultimi cinque anni (2015 - 2020)

Capitolo 4

CIC-DDoS2019 Dataset

Essendo la metodologia dell'individuazione degli attacchi tramite firma (signature-based), usata in modo elevato (sia totalmente, che in parte), descriviamo e presentiamo il dataset che risulta essere il più aggiornato al momento della stesura di questo documento: il CIC-DDoS2019 Dataset[1].

4.1 Caratteristiche

Per inquadrare meglio le varie categorie elencate nella figura 2.1, proponiamo qui di seguito un semplice sommario, dove indichiamo i protocolli, debolezze e/o caratteristiche che conferiscono i vari nomi DoS/DDoS.
Reflection Attacks: Pratica che consiste nel mandare richieste usando IP falsificati a diversi server, i quali risponderanno al vero indirizzo invece che al mittente.[52] Funzionano poiché alcuni servizi internet offrono amplificazione, ovvero la risposta è maggiore della richiesta mandata dall'attaccante/i [53]. Come misura di amplificazione vengono spesse definite due entità: *BAF* (Bandwidth Amplificaton Factor) e *PAF* (Packet Amplification Factor), calcolate nella seguente maniera:[54]

$$\text{BAF} = \frac{\text{lunghezza(dati UDP) da amplificatore a vittima}}{\text{lunghezza(dati UDP) da attaccante ad amplificatore}} \tag{4.1}$$

$$\text{PAF} = \frac{\text{numero di pacchetti da amplificatore a vittima}}{\text{numero di pacchetti da attaccante ad amplificatore}} \tag{4.2}$$

Gli attacchi facenti parte di questa categoria presentano 3 attori, l'attaccante/i che utilizza IP forgiati, il mezzo vettore per la riflessione ed amplificazione (che varia da tecnica a tecnica) ed infine il target.[2][55]

4.1.1 MSSQL

Questo attacco si basa sull'abuso del protocollo SQL Server Resolution Protocol (SSRP) [MC-SQLR], che rimane in ascolto sulla porta UDP 1434 [56][58]. SSRP è utilizzato per comunicare le informazioni del punto terminale di una determinata istanza del database[59]; ogni qual volta un client ha bisogno delle informazioni su server configurati con MS SQL, il SSRP viene utilizzato. L'attaccante, utilizzando scripts e mascherando il proprio indirizzo IP, esegue continue richieste al server SQL, il quale risponderà ogni qual volta con una lista di istanze; tutte queste verranno indirizzate al vero IP. L'amplificazione viene effettuata dal server MS SQL, che secondo l'analisi condotta da akamai, risulta essere circa 25 volte in dimensione rispetto alla richiesta del client.[57]

[1] una discussione sui vari dataset è presente nella omonima sezione 3
[2] La descrizione offerta dall' Istituto Canadese per la sicurezza Informatica (CIC), non ci sembrava appropriata poiché fornisce la pressoché medesima definizione per i Reflection Attacks e per gli Exploitation Attacks

4.1.2 SSDP

Il protocollo sfruttato è il Simple Service Discovery Protocol usato dai dispositivi Universal Plug and Play (UPnP), in condizioni normali permette a questi di comunicare la loro esistenza ad altri dispositivi presenti sulla rete; utilizza la porta 1900 UDP. Un attacco tipico di questo tipo segue due fasi:
una di scansione, mirata alla scoperta di più possibili dispositivi UPnP che rispondono alla ricerca dell'attaccante, ed una di attacco, che consiste nell'invio di pacchetti UDP M-SEARCH con IP forgiato. L'amplificazione in questo caso è offerta dai dispositivi UPnP, i quali possono produrre un'amplificazione di circa 30 volte.[60][61]

4.1.3 DNS

Per comprendere questa tipologia, è necessario prima introdurre cosa siano i DNS Open Resolver. Un DNS Open Resolver è un server DNS che permette a clients DNS che non fanno parte del suo dominio amministrativo, di usufruire dei sui servizi per la risoluzione del nome ricorsiva (recursive name resolution)[62]. La caratteristica qui sfruttata risiede nel fatto che la risoluzione standard DNS non è autenticata, ovvero è possibile per un utente malintenzionato, generare query DNS con un indirizzo IP forgiato, e la risposta DNS sarà mandata all'IP in questione. Un utente che non ha effettuato alcuna query potrebbe ricevere un numero eccessivo di risposte DNS.[63]

4.1.4 LDAP

Poniamoci in un contesto di rete aziendale, con vari utenti, applicazioni, documenti, stampanti e altre risorse accessibili, come possiamo descrivere e rendere disponibili le informazioni della suddetta rete? Tramite l'utilizzo di un apposito database, chiamato certe volte, directory. Un protocollo molto comune, usato in questo contesto, per servizi di directory, è il LDAP[64], il quale definisce una metodologia standard per accedere e aggiornare informazioni in una directory. Solitamente LDAP utilizza TCP per la comunicazione, tuttavia può far uso del protocollo di trasporto senza connessione UDP[65]. Sfruttando porte UDP aperte e pubbliche di server LDAP (solitamente la 389) è possibile effettuare queries con IP forgiato[66], ottenendo dal 46 al 55, come *BAF*.[67]

4.1.5 NetBIOS

Principalmente NetBIOS viene utilizzato da applicazioni in computer diversi, per accedere a risorse condivise e per "trovarsi" su una rete di area locale.[68]. Il servizio sfruttato per eseguire il DoS è il NetBIOS Name Service (NBNS), il quale su protocolli TCP/IP non offre metodi per l'autenticazione della comunicazione [69]. Inviando opportuni messaggi manipolati, l'attaccante può rendere la macchina target impossibilitata a comunicare con altri NetBIOS host. [70]

4.1.6 SNMP

Protocollo Internet per gestire dispositivi su reti IP, molti tipi di dispositivi supportano SNMP, ad esempio: routers, switches, servers, stampanti, UPSs e altri[71]. Nel lavoro svolto da V.Paxson[72] viene categorizzato come "probabilmente non un pericolo", conclusione che deriva in parte dalla facile identificazione dell'attacco, poiché in arrivo sulla porta SNMP 161. Tuttavia nel lavoro svolto dalla Tottori University, in un rapporto tecnico[73], hanno trovato un caso in qui lo switch centrale è vulnerabile ad SNMP. La causa di ciò, è stata attribuita ad una vulnerabilità architetturale dello switch che potrebbe causare un intenso uso di CPU, quando lo switch in questione riceve pacchetti SNMP.

4.1.7 Portmap

Chiamato a volte anche Portmapper, è un servizio che mappa programmi RPC a numero di versione e specifiche porte per il trasporto; questo servizio rende la corrispondenza dinamica (dynamic binding) di programmi remoti possibile[74]. Il client attraverso query alla porta 111 (PortMapper),

ottiene l'appropriata porta RPC; il problema risiede nell'utilizzo di UDP e dal fatto che quando interrogato PortMapper, può rispondere con una dimensione maggiore di quella della richiesta (amplificazione). Sempre secondo CISA[67], tramite questo servizio, è possibile ottenere fra 46 a 55 come *BAF*.

4.1.8 CharGen

Strumento di debug: Un server rimane in ascolto per datagrammi UDP sulla porta 19 UDP, quando un datagramma è ricevuto, un datagramma di risposta viene inviato dal server, contenente un numero random (fra 0 e 512) di caratteri [75]. Questo attacco è solitamente indirizzato a dispositivi con OS della famiglia UNIX, e oltre a CharGen usa in parallelo, il servizio ECHO[3] (porta 7). L'attacante manda pacchetti UDP ECHO alla porta che supporta CharGen con l'indirizzo di ritorno della porta ECHO della vittima, creando così un loop infinito.[76]

4.1.9 NTP

Protocollo per la sincronizzazione di un set di orologi di rete, sfruttando una serie di clients e servers distribuiti[78]. La caratteristica qui sfruttata risiede nella possibilità dell'uso del comando MONLIST, e della presenza di server NTP pubblici che ne permettono l'uso. L'amplificazione è offerta dal comando MONLIST, il quale ritorna gli ultimi 600 indirizzi IP che si sono connessi al server, per quanto riguarda il *BAF* il CISA[67] propone il valore 556.9, mentre il lavoro svolto da Czyz, Jakub, et al.[79] suggerisce un valore di 4 per la risposta tipica data dal comando MONLIST di un server NTP, ed almeno 15, per circa un quarto di server disponibili. Sempre gli stessi autori[79], individuano il problema dei cosiddetti "Mega Amplificatori", ovvero di server NTP che come risposta eccedono i 50K, il limite massimo di dimensione previsto per la risposta alla query MONLIST. Questi "Mega Amplificatori" come appunto implica il nome, offrono un *BAF* nettamente maggiore rispetto a quello definito precedentemente.[4]

4.1.10 TFTP

Protocollo molto semplice per il trasferimento di files. Molto semplice perché può solo leggere e scrivere files (o mail) da, o a, server remoti [80] ed è limitato alla trasmissione di binario o ASCII. Il servizio è disponibile sulla porta 69 e anche se concede l'utilizzo di molti protocolli di trasporto, solitamente fa uso di UDP. All'interno del lavoro svolto da Sieklik, Boris, Richard Macfarlane, and William J. Buchanan [81], viene descritto un nuovo attacco TFTP, nel quale l'attaccante invia la più piccola richiesta per un file sul server TFTP (con ovviamente l'indirizzo IP forgiato per essere identico a quello del target), stimolando una risposta maggiore diretta al target. Il *BAF* calcolato dagli autori per questo attacco è 60[5].

Exploitation Attacks: In questo caso gli attori malintenzionati sfruttano certe proprietà o bugs di qualche protocollo per indurre un consumo eccessivo delle risorse del target.[82] [2]. In alcune ricerche, vengono anche chiamati con *Protocol Esploitation Attack*[83]

- SYN Flood: Sfruttando la procedura di apertura di una connessione del TCP (Three-way handshake), l'attacante crea molte connessioni semi-aperte (half-open), consumando le risorse del server dedicate alla memorizzazione delle connessioni[84]. La connessione semi-aperta è ottenuta in due modi: nel primo il client che invia la richiesta SYN non risponde al SYN-ACK dell'host, nel secondo il client invia la richiesta SYN con un indirizzo IP forgiato[85]. L'idea di fondo è di non rispondere all' SYN-ACK dell'host, costringendolo a mantenere occupate delle risorse per ogni sessione richiesta tramite il messaggio SYN.

- UDP Flood: In questo caso l'attività di flooding è ottenuta sfruttando la capacità dell'UDP di produrre numeri elevati di pacchetti [84]. L'attacante in questo caso invia numerosi pacchetti UDP a porte scelte a caso del target. Il sistema target risponde con gli appropriati pacchetti

[3] il dispositivio che riceve dati su questa porta ne rimanda una copia identica al mittente (appunto un eco)

[4] Mi sentirei di consigliare anche la semplice ma effettiva esposizione esposta da Tom Scott nel video "The Attack That Could Disrupt The Whole Internet - Computerphile" del 14 mar 2014.

[5] Gli autori invece che utilizzare il termine *BAF* utilizzano *Amplification factor (A)*

ICMP se la porta risulta chiusa. Il grande numero di risposte ICMP fa rallentare il sistema, o addirittura provoca un crash.[86]

- UDP Lag: Solitamente usato per interrompere una connessione in giochi online, con l'intento di degradare le performance degli altri giocatori [138].

4.2 Tempistiche e tipi di attacco

Il dataset è suddiviso in due giorni, uno di training (12 Gennaio 2018) e uno di test (11 Marzo 2018). Nei due giorni sono stati registrati i vari attacchi simulati descritti precedentemente,con le tempistiche indicate nella tabella 4.1 e nelle timeline 4.1 e 4.2.

Attacchi e tempistiche		
Giorno	Tipo di Attacco	Tempistica
Primo giorno / Training Day	NTP	10:35 - 10:45
	DNS	10:52 - 11:05
	LDAP	11:22 - 11:32
	MSSQL	11:36 - 11:45
	NetBIOS	11:50 - 12:00
	SNMP	12:12 - 12:23
	SSDP	12:27 - 12:37
	UDP	12:45 - 13:09
	UDP-Lag	13:11 - 13:15
	WebDDoS	13:18 - 13:29
	SYN	13:29 - 13:34
	TFTP	13:35 - 17:15
Secondo giorno / Testing Day	PortMap	9:43 - 9:51
	NetBIOS	10:00 - 10:09
	LDAP	10:21- 10:30
	MSSQL	10:33 - 10:42
	UDP	10:53 - 11:03
	UDP-Lag	11:14 - 11:24
	SYN	11:28 - 17:35

Tabella 4.1: Attacchi e tempistiche del dataset CIC-DDoS2019

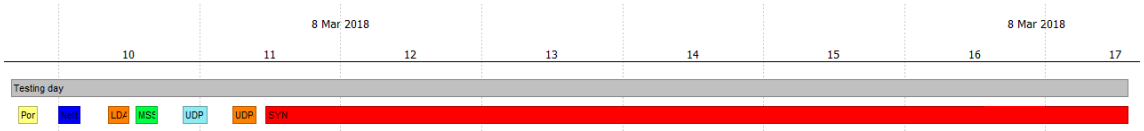


Figura 4.1: Timeline del giorno di Test (Testing-day)

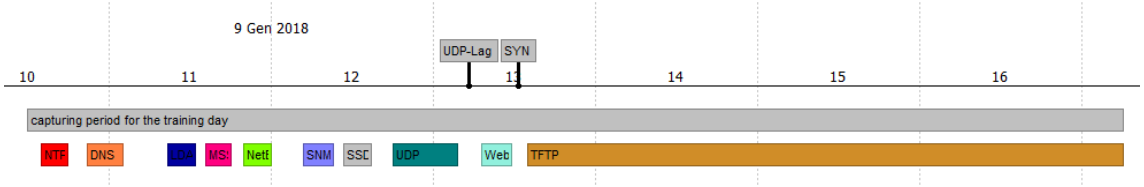


Figura 4.2: Timeline del giorno di Training (Training-day)

Capitolo 5

Corpo della ricerca

5.1 Machine Learning

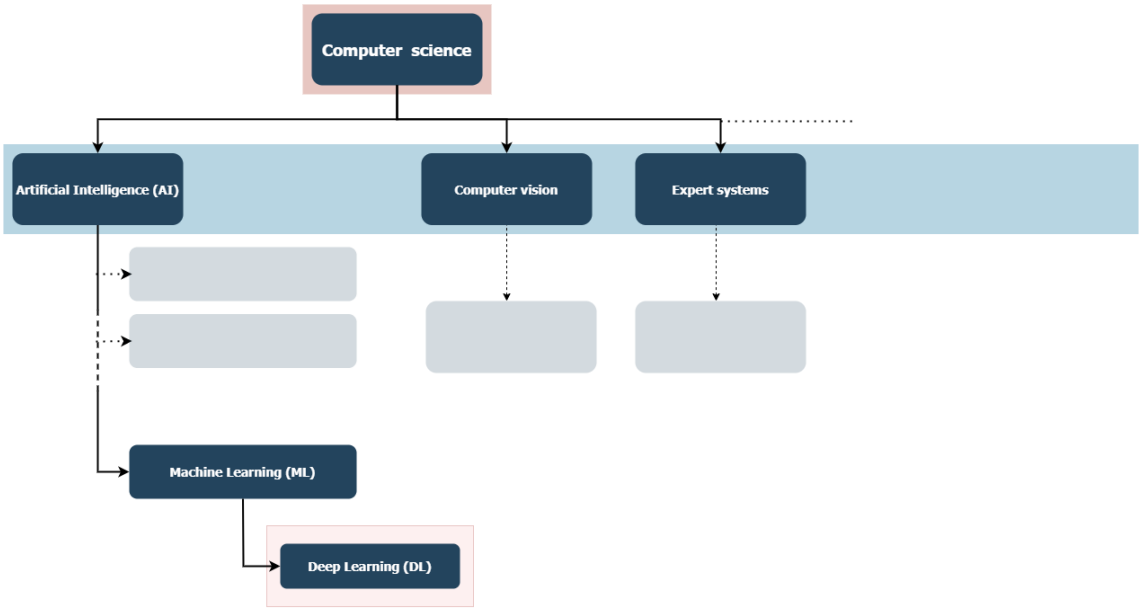


Figura 5.1: Relazione fra ML e DL. Per rendere leggibile il grafico indichiamo con le frecce e righe tratteggiate la presenza di altri argomenti.

5.1.1 Cosa è il ML, un infarinatura

Fino ad ora abbiamo parlato di ML e DL, ma cosa vanno ad indicare specificamente questi termini? Premettiamo innanzitutto che lo scopo di questa sezione è un infarinatura generale sull'argomento, con lo scopo finale di fornire al lettore un'idea generale, e una migliore consapevolezza al riguardo. Il Machine Learning (*ML*), o in italiano Apprendimento Automatico, è una branca dell'intelligenza artificiale. Un processo, algoritmo può essere classificato come Machine Learning se soddisfa entrambe le condizioni delineate da Bishop, Christopher M.[6]:

- **1:** Il processo si può dire complicato, e i dati risultati da questo, sono differenti in maniera sostanziale da quelli iniziali.
- **2:** Inoltre come risultato finale di questo processo, l'uscita futura del sistema sarà cambiata.

Cosa si intende con la seconda condizione? Il sistema deve derivare nuove informazioni autonomamente partendo da quelle offerte come input. Gli autori propongono un ottimo esempio per fare

chiarezza: un sistema in grado di risolvere funzioni matematiche complesse, anche se soddisfacesse la prima condizione, non sarebbe in grado di "imparare" in alcun modo nuove informazioni. Ovvero, non vi è alcun processo atto alla modifica del sistema, con lo scopo di risolvere altri tipi di equazioni, andando dunque a negare la seconda condizione.
All'interno del ML sono presenti diverse tecniche, quella che è a noi necessaria per poter discriminare il traffico benigno da quello maligno, ricade nella problematica chiamata: *Classification* e *Data mining*[25]. Attraverso le metodologie ad esso pertinenti, è infatti possibile scovare relazioni fra gli oggetti (attacchi) presenti in un pattern (traffico di rete), nonché scegliere le qualità significative di ogni attributo (feature selection) utili alla classificazione.

5.1.2 Deep Learning (DL)

Deep Learning è una branca del Machine Learning (Figura 5.1) cui recentemente[1] è stata impiegata anche nel campo dell'individuazione degli attacchi Dos/DDoS. Ma cosa contraddistingue questa branca dalle altre? Un indizio è dato dai termini, *Deep* significa *profondo*, mentre i modelli, da noi identificati come "tradizionali" (rilegati al ML), vengono chiamati *Shallow*, ovvero *poco profondi/superficiali*. Questa profondità richiama la natura esclusiva[2] posseduta dai modelli DL: la presenza di molte strutture annidate non lineari (layers)[3]. Proprietà che rende il comportamento del DL, simile a quello delle balck-boxes[26].

> **Nota: Il problema della scatola nera (Black-box problem)**
> Un componente definito con il termine "scatola nera" (Black-box appunto), è uno che non rileva alcuna informazione riguardo al suo funzionamento, struttura od/ed implementazione interna[32]. Ovvero è come cercare di capire il contenuto di una scatola nera sigillata osservandola dall'esterno.

Chiariamo il concetto: prendiamo ad esempio le reti artificiali neurali (ANNs) della branca DL, le quali si basano sulla simulazione di un cervello umano nel riguardo del processo di apprendimento, queste sono composte da uno o più livelli nascosti (Figura 5.2), non lineari, e con un numero elevato di iperparametri. Ora, facendo sempre riferimento alla Figura 5.2 definiamo con X il vettore degli ingressi x_1, x_2, x_3, con Y il vettore delle uscite binarie y_1, y_2 e con H il vettore delle funzioni appartenenti ai livelli nascosti $h_1, h_2, \cdots, h_N$; il modello cercherà dunque una funzione matematica $f(X) = Y$ che potrebbe essere arbitrariamente complessa e cambiare in base all'evoluzione degli ingressi X. Dunque, invece che definire una funzione determinata e precisa, ANN offre un approssimazione $f(X) = Y + \epsilon$, che con il grande numero di parametri, la non linearità del modello e i livelli multipli, fa sì che da quest'ultima, non sia possibile dedurre $f(X)$.
Ai fini del nostro problema, questa caratteristica non dovrebbe porre enormi svantaggi (a differenza di altri campi dove l'interpretabilità del modello è di fondamentale importanza, e.g. campo medico, veicoli autonomi, ecc.).
Altre caratteristiche che permettono la distinzione fra le due categorie sono elencate nella Tabella 5.1; in linea generale possiamo comunque affermare che i modelli DL sono più complessi e propensi all'apprendimento, rispetto ai "semplici" modelli shallow. La scelta fra modelli ML o DL non è comunque di ovvia risoluzione, la superiorità appena definita, dei modelli DL, è infatti contrapposta con un grande svantaggio. Nella definizione del nostro ambiente, è infatti risultata di chiara importanza l'abilità del sistema ad individuare gli attacchi in tempo reale; abilità nella maggior parte dei casi, non soddisfatta dai modelli DL[31].

5.1.3 L'apprendimento

La fase in cui il modello esegue tutte quelle operazioni correlate al processo di apprendimento ("imparare", "training"), può essere categorizzata in quattro gruppi, basandosi sul grado di intervento da parte di un operatore umano e al tipo di natura del dataset:

[1] da circa il 2015 [31]
[2] esclusiva nei confronti dei modelli di tipo Shallow.
[3] Il termine deep si riferisce dunque al numero di livelli, layers, del modello.

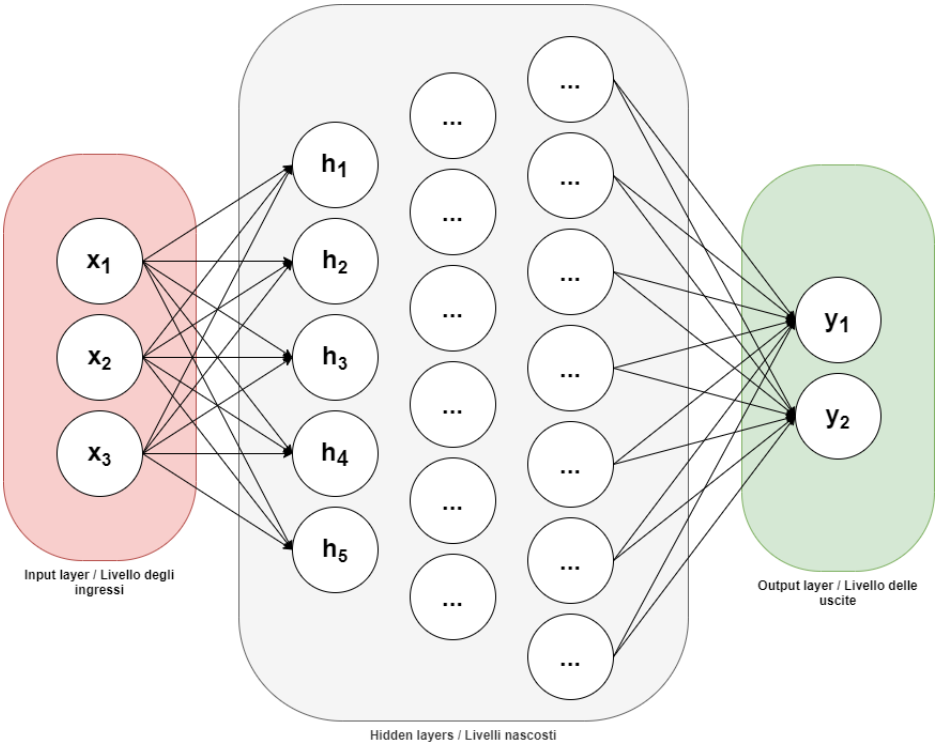


Figura 5.2: Un esempio di rete neurale, più precisamente: un esempio di rete neurale di tipo feedfoward.

Differenze salienti fra algoritmi "classici" ML e algoritmi di DL		
	ML	**DL**
Dipendenza dalla dimensione del Dataset	decresce generalmente con la dimensione del dataset	più preformanti con dataset di dimensioni elevate
Hardware necessario	nessuna caratteristica saliente da riportare	solitamente richiede macchine più performanti, specialmente per la GPU
Scelte delle caratteristiche / Feature processing	affidata, nella maggior parte dei casi, ad esperti del campo	processo automatico
Tempi	relativamente rapidi nel training, medio-lunghi del testing (rispetto agli algoritmi di DL, e non sempre verificato)	lunghi nel training, rapidi nel testing
Interpretabilità dei risultati	in base alla complessità del modello sono di facile, media e difficile interpretazione	blackBoxes, Scatole nere, il perché dell'ottenimento del risultato è di difficile se non impossibile interpretazione.
Numero di parametri	in numero contenuto	nettamente maggiori
Capacità d'apprendimento	minore, rispetto sempre ai modelli DL	grazie al grande numero di parametri, hanno una maggiore abilità nell'apprendimento[4]

Tabella 5.1: Alcune delle differenze esposte in [88] e in [31].

- Supervised, Supervisionato
- Unsupervised, Senza supervisione
- Semi-supervised, Semi supervisionato[5]
- Reinforced, Rafforzato

Nei problemi di classificazione, l'intervento umano si traduce classicamente, nell'uso o meno di dati etichettati (labeled data). Le etichette in questo caso, dato essere un problema di classificazione binaria, sono: "attacco DoS / DDoS" e "benigno"[6].

5.1.4 Supervised learning

Se il modello usufruisce di un dataset composto da dati etichettati, o meglio, sfrutta nel processo di apprendimento l'etichetta stessa, ricade in questa categoria. Tutti i dataset fino ad ora citati, ad esempio, sono già implicitamente od esplicitamente etichettati, lasciando la libertà ai ricercatori su quale tipo di metodologia utilizzare. Dataset etichettati comportano, in linea generale, ore umane e/o lavoro da parte degli autori, maggiori rispetto agli altri metodi[7].

5.1.5 Unsupervised learning

In questo caso il modello cerca connessioni e relazioni che accomunano i dati presenti nel dataset, attraverso delle ipotesi fatte a priori[89]. Viene chiamato senza supervisione, perché appunto, non è richiesto alcun intervento umano nella classificazione dei dati del dataset. Il lettore potrebbe essere a questo punto indotto a dedurre la superiorità di una metodologia rispetto ad un altra; d'altronde perché sprecare ore umane per eseguire un compito, quando esistono modelli che possono compierlo autonomamente? La risposta è molto articolata e al di fuori dallo scopo di questo documento, comunque basti sapere al lettore, che fra gli svantaggi di questa metodologia, vi sono: possibili degradazioni alla qualità del dataset finale ed impossibilità sull'applicabilità di determinati algoritmi di ML/DL. Di conseguenza, ambo le tipologie hanno senso d'esistenza, e l'una non è più performante all'altra a priori.

5.1.6 Semi-supervised learning

Facilmente deducibile dal nome, questa categoria racchiude tutti quei metodi capaci di trarre conclusioni e/o predizioni, tramite la combinazione di dati già classificati e non. Un'ottima introduzione e fonte di risposte per possibili domande da parte del lettore, è offerta dalla sezione *1 FAQ* di Zhu, Xiaojin Jerry[90].

5.1.7 Reinforcement learning

Categoria il cui scopo è: il controllo di un sistema, tramite la ricerca di massimizzazione di un determinato parametro numerico, il quale è rappresentativo di un obiettivo perseguibile nel lungo termine. In parole più concrete, il sistema osserva continuamente l'ambiente in cui sta operando α, esegue un'azione x e riceve in base ad una funzione $f(\alpha)$ un premio $r(f(\alpha))$. L'obiettivo del sistema è trovare l'azione che massimizza i futuri premi:

$$\arg\max_{\alpha} r(f(\alpha)) \tag{5.1}$$

Sostanzialmente, rappresenta una forma di retroazione, applicata all'apprendimento.

[5] Alcune volte, nella letteratura, questa non viene inclusa nella categorizzazione. Lasciando il supervised, l'unsupervised e il reinforced come uniche categorie.

[6] Nulla vieta una raffinazione della classificazione, mediante l'individuazione dello specifico attacco.

[7] alcune volte questo lavoro può essere affidato a molteplici operatori umani inconsapevoli, come stato fatto in passato dal sistema *reCAPTCHA* nella digitalizzazione dei libri (l'etichettatura qua consisteva nel riconoscimento di non comprensibili per la macchina/modello). Un ottimo video su *reCAPTCHA* "I'm Not A Robot" di Tom Scott

	Tipologia Dataset			Note
	Assente	Parziale	Completa	
Supervised			✓	Dispendioso da parte dell'autore dell'etichette
Unuspervised	✓			Non è sempre possibile l'utilizzo di questa tecnica
Semi-supervised		✓		
Reincorcement	✓			Deve essere possibile valutare qualitativamente la "bontà" del risultato

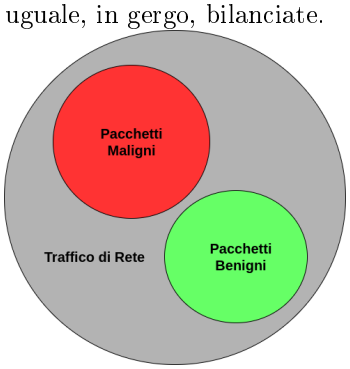
Tabella 5.2: Le varie tecniche di apprendimento con le tipologie di dataset necessari

5.2 Metriche per la valutazione

5.2.1 Introduzione

Prima di esporre le varie metriche classicamente utilizzate per la valutazione di bontà e qualità di un determinato modello, è necessario prima parlare delle matrici di confusione ed i concetti ad esse correlate. Una matrice di confusione, od utilizzando il corrispettivo termine inglese, **Confusion Matrix**, è una matrice che riassume come il modello in analisi ha classificato i vari elementi nelle determinate classi. Grazie a questa matrice, si possono immediatamente individuare gli elementi classificati correttamente e quelli erroneamente, fornendo una veloce e chiara misura sulla qualità del modello. Facciamo un semplice esempio per chiarire le idee, ipotizziamo di voler sviluppare un modello che sia in grado di distinguere e categorizzare correttamente tre tipi di frutta: mele, banane ed arance. Il dataset artefatto usato sarà composto da 50 elementi per ciascun frutto: quindi 50 mele, 50 banane e 50 arance. Una volta sviluppato ed implementato il modello, una possibile confusion matrix che ora ci si potrebbe presentare è quella proposta in Tabella 5.3. Leggendo orizzontalmente la matrice, possiamo verificare che il numero di elementi realmente presenti nel dataset per ogni categoria sia effettivamente uguale a 50. Verticalmente invece leggiamo che il modello ha classificato correttamente 40 mele, mentre 5 banane e 7 mele sono state soggette ad una errata classificazione. Estendendo il discorso alla seconda colonna, 47 sono state le banane individuate, di cui 45 classificate correttamente e 2 arance scambiate erroneamente per banane (evitiamo di parlare della terza colonna, dato che dovrebbe essere ora chiaro il metodo di lettura di questo strumento). Sulla diagonale sarà dunque sempre possibile leggere tutte le istanze classificate correttamente.[8]

Figura 5.3: Una possibile classificazione binaria effettuabile sul traffico di rete, in questo caso le due classi sono di dimensione pressoché uguale, in gergo, bilanciate.



		Predizione		
		Mela	Banana	Arancia
Reali	Mela	40	0	10
	Banana	5	45	0
	Arancia	7	2	41

Tabella 5.3: Confusion matrix dell'esempio coinvolgente la frutta.

Un'altra forma di confusion matrix molto utilizzata, fa uso di una distinzione binaria. Invece dunque di impiegare le varie categorie usate nella classificazione, che potrebbero risultare numerose rendendo la matrice di difficile e/o scomoda interpretazione, la matrice viene ridotta ad una $2 * 2$. Questa distinzione è composta da due classi, una positiva ed una negativa, racchiudendo tutti i possibili casi a 4 possibili categorie(Tabella 5.4). In inglese, queste vengono indicate con i nomi di: True Positive **TP** / True Negative **TN** e False Negative **FN** / False Positive **FP**.

[8] A volte le colonne e le righe sono scambiate fra di loro.

		Valore reale		
		Positivo	Negativo	
Predizione	Positivo	**TP**	**FP**	TP + FP
	Negativo	**FN**	**TN**	FN + TN
	Totale	TP + FN	TN + FP	

Tabella 5.4: Esempio di una binary confusion matrix.

Nota: Da uno a molti, la matrice confusionale
Dalla matrice confusionale è possibile costruire un grande numero di metriche (nell'ordine della ventina), per motivi di sinteticità, elencheremo quelle maggiormente riscontrate nella letteratura.

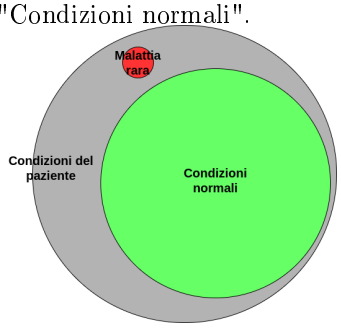
Utile, sempre in questo contesto, è il concetto delle classi:

5.2.2 Classi

Nel caso in cui si è in presenza di algoritmi che eseguono una classificazione (chiamati appunto classificatori), sorge il termine classe, il quale va ad indicare una determinata categoria, un insieme, un raggruppamento di elementi che sono accomunati da una o più caratteristiche. Calandoci nel nostro ambito di ricerca, una diffusa classificazione effettuata sui pacchetti del traffico di rete è:

- **Classe A: Pacchetti maligni**
- **Classe B: Pacchetti benigni**

Figura 5.4: Esempio di classi sbilanciate, in questo caso l'insieme (classe) "Malattia rara" è di dimensione nettamente inferiore rispetto a "Condizioni normali".



Il problema del bilanciamento

Un tema ricorrente e sorgente di varie problematiche sull'utilizzo di vari algoritmi di ML e DL, è il problema del bilanciamento delle classi. Con bilanciate, intendiamo la situazione rappresentata in figura 5.3, dove le due classi presentate possiedono dimensioni dello stesso ordine di grandezza (sono comparabili fra di loro). Uno scenario possibile in cui un algoritmo di ML potrebbe riscontrare delle complicazioni, è il caso in cui le classi definite all'interno del nostro dataset sono di dimensione non comparabile, o meglio, sbilanciate. Prendiamo in considerazione il seguente esempio: abbiamo un dataset che è composto dall'analisi mediche di un campione di persone; fra di queste vi è presente una piccola percentuale affetta da una particolare malattia rara. Il nostro obiettivo è creare un modello capace di classificare i pazienti sani, nella classe "Condizioni normali", mentre quelli affetti dalla malattia, nella classe "Malattia rara" (Figura 5.4). Se dovessimo addestrare il modello con il dataset fornito, potrebbe raggiungere un'alta accuratezza (Equazione 5.10) seguendo la regola banale:

$$f(x) = c_{\text{condizioni normali}} \forall x \in X \tag{5.2}$$

Dove X è l'intero dataset, x la generica istanza/vettore/persona facente parte del dataset, $f(x)$ la funzione del classificatore e $C = [c_{\text{malattia rara}}, c_{\text{condizioni normali}}]$ le due classi (Figura 5.5).
Più precisamente, l'accuratezza raggiunta risulterebbe del 96%, tramite l'utilizzo di un modello inutilizzabile.

5.2.3 Nomenclatura

Chiariti questi concetti, ci è possibile ora definire le metriche maggiormente utilizzate per la valutazione dei modelli:

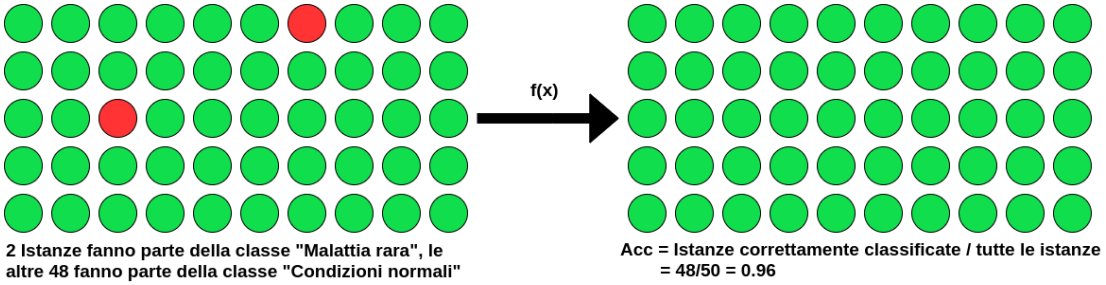


Figura 5.5: In questo caso il classificatore applica una funzione banale: indica come salutari sempre ed ogni paziente. Nonostante ciò, a causa del dataset pesantemente sbilanciato, riesce a raggiunge un *Acc* di 0.96.

- **Precision o anche Sensitivity**

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{istanze classificate correttamente come positive}}{\text{tutte le istanze classificate come positive}} \tag{5.3}$$

  Ovvero, fra tutte le istanze classificate come positive dal modello, quante di queste sono effettivamente realmente positive?

- **False Alarm Rate (FAR) o anche False Positive Rate (FPR)**

$$\text{FAR} = \frac{FP}{TP + FP} = \frac{\text{istanze negative erroneamente classificate come positive}}{\text{tutte le istanze classificate come positive}} \tag{5.4}$$

  Il numero di falsi allarmi misura dunque quella porzione di previsioni di eventi positivi non verificati[91].

- **Detection Rate (DR) o anche Negative Predictive Value (NPV) [93]**

$$\text{DR} = \frac{TN}{TN + FN} \tag{5.5}$$

- **True Negative Rate (TNR) o anche specificity:**

$$\text{TNR} = \frac{TN}{FP + TN} \tag{5.6}$$

  La proporzione di negativi effettivi (TN), correttamente identificati (il corrispettivo "negativo" di TPR, sensitivity, recall)[94].

- **False Negative Rate (FNR)**

$$\text{FNR} = \frac{FN}{TP + FN} \tag{5.7}$$

- **Total Error Rate (ER)**

$$\text{ER} = \frac{FN + FP}{TP + FP + TN + FN} \tag{5.8}$$

- **Recall, o anche True Positive Rate (TPR), o sensitivity**

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{istanze classificate correttamente come positive}}{\text{tutte le istanze realmente positive}} \tag{5.9}$$

  Considerando tutte le istanze realmente positive, quante di queste sono state individuate e classificate correttamente dal modello? O in altri termini, la proporzione di positivi effettivi (TN), correttamente identificati (il corrispettivo "positivo" di TNR, specificity[94]).

- **Accuracy**

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{\text{istanze correttamente classificate}}{\text{tutte le istanze}} \tag{5.10}$$

  La proporzione dei risultati correttamente identificati, sia che essi siano positivi ché negativi[94].

- F_1**-score**

$$F_1\text{-score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.11}$$

  è la media armonica fra la precisione e il recupero (recall).

- **Receiver Operating Characteristic Curve (ROC)[95]**
  Grafico 2D che mostra le performance di un modello classificatore a diverse soglie. Come ordinate si adopera il **TPR** e come ascisse il **FPR**. Nel caso di un singolo punto si può dire informalmente che: un classificatore, se nel diagramma ROC si pone più in alto a sinistra rispetto ad un altro, viene considerato migliore (caratteristica denotata dal gradiente colorato in Figure 5.6). La linea tratteggiata in Figura 5.6 denota un classificatore "casuale", ovvero

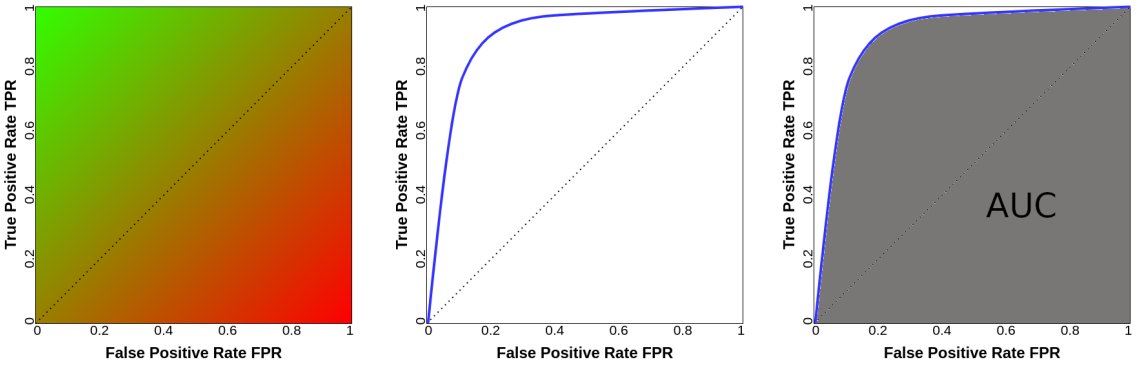


Figura 5.6: A sinistra viene presentato il diagramma ROC vuoto con il gradiente indicante la "bontà" del modello, in centro un esempio di curva nello spazio ROC ed infine a destra, la relativa AUC.

  che nel 50% delle volte, predice correttamente/scorrettamente la classe dell'istanza (TPR = FPR).
  Come si denota la curva? Ipotizziamo che il nostro modello fornisca in uscita una probabilità che denota l'appartenenza o meno di un pacchetto, ad un attacco. Questa probabilità, per i nostri scopi, dobbiamo trasformarla in una decisione di tipo binario (o blocchiamo il pacchetto, o lo lasciamo transitare), decisione che scaturisce dall'impostazione d'una soglia. E.g. se la probabilità di appartenenza ad un attacco del pacchetto supera il 50%, possiamo impostare il modello in modo che lo classifichi come maligno. Diverse soglie producono diverse matrici confusionali, ergo diversi punti nel grafico ROC, l'unione di questi crea la curva nello spazio ROC.[9][10]

- **Area Under the ROC Curve (AUC)[95]**
  Ai fini del confronto fra modelli, i grafici ROC potrebbero risultare scomodi, sarebbe infatti più conveniente avere a disposizione un singolo valore numerico. A questo scopo viene in soccorso la metrica AUC. Questa è l'area al di sotto della curva ROC, l'integrale per così dire, potrà variare dunque da 0 ad 1, dove il minimo indica un modello che sbaglia sempre la classificazione, mentre il massimo indica uno che classifica sempre correttamente.

A volte il trattamento di queste metriche può risultare confusionale, seguendo dunque l'adagio *"Un'immagine vale più di mille parole"*, proponiamo la Figura 5.7. In questa, le equazioni vengono rappresentate come divisioni fra gli elementi della confusion matrix TP,TN,FP,FN (Tabella 5.4).

[9] Un buon video che spiega l'argomento: ROC and AUC, Clearly Explained!
[10] ROC Curve: dimostrazione interattiva, disponibile qui.

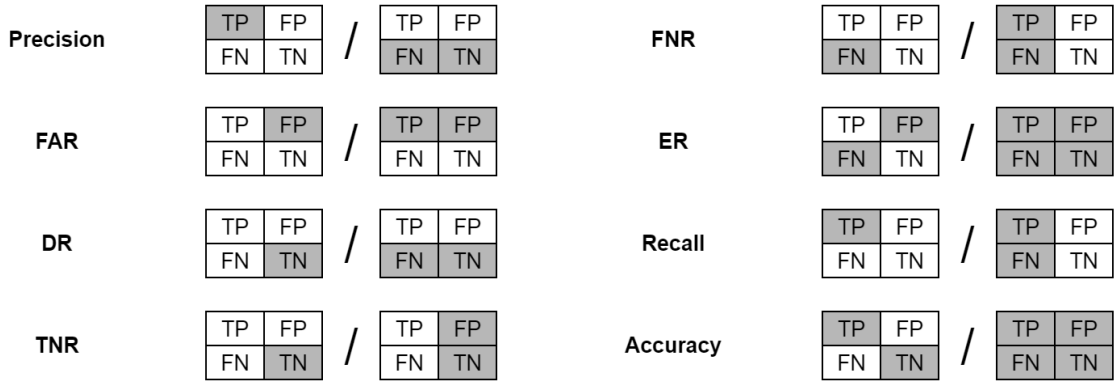


Figura 5.7: Confusion matrices delle metriche introdotte. Vengono evidenziati in grigio quali elementi contribuiscono al numeratore e al denominatore.

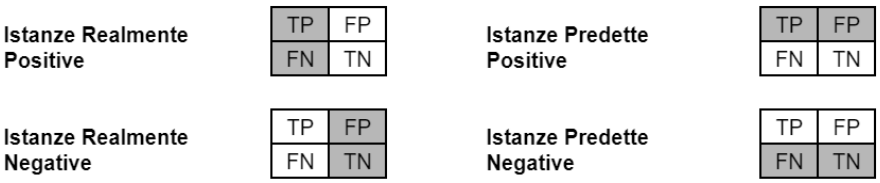


Figura 5.8: Significato di alcune combinazioni importanti

Un'ottima sorgente di informazioni, per il lettore interessato particolarmente al significato di ogni metrica, è disponibile al seguente sito.

5.3 Strumenti

Gli strumenti a disposizione dei ricercatori, adoperati per l'impiego di algoritmi di ML, sono molti. Principalmente per ragioni di semplicità d'uso e popolarità, la lista può essere ridotta a pochi elementi, rendendo più concisa e chiara l'esposizione di quest'ultimi. In questa sezione ci occuperemo solamente di citare gli strumenti riscontrati più frequentemente nella letteratura degli ultimi anni, con lo scopo finale di indirizzare il ricercatore alla scelta più consona in base al grado di esperienza e familiarità dell'ultimo. Le ricerche prese in considerazione sono 14, comprese fra la fine del 2016 e fine 2020, e nei primi risultati di *Google Schoolar*[11] dopo aver effettuato la seguente query di ricerca: *"DoS DDoS Machine Learning Deep Learning"*. La tabella a cui fare riferimento è la 5.5.

[11] Motore di ricerca che si concentra su articoli, libri, tesi, ecc. per maggiori informazioni rimandiamo il lettore al seguente sito.

Ricerca	Data	Datasets	ML o DL usati	Strumenti	Metriche									
					TNR	Acc	Prec	Recall	F1	FAR	FNR	DR	ER	Specificity
[27]	22 ottobre 2016	KDD CUP 1999	Heterogeneous Classification Ensemble Model	Matlab	✓	✓	✓							
[33]	22 novembre 2016	customized dataset	SAE	Matlab		✓	✓	✓	✓					
[34]	21 ottobre 2017	ISCX2012	combinazione di RNN LSTM CNN	Keras		✓								
[22]	11 aprile 2018	Custom IoT devices network	kNN, "KDTree" algorithm, LSVM , DT, RF, NN	Scikit-learn e Keras		✓	✓	✓	✓					
[133]	11 giugno 2018	Caida Ucsd DDoS Attack 2007	SimpleMKL method	Matlab e Code::Blocks						✓		✓	✓	
[134]	2018	customized dataset	RF	? e Apache Spark						✓	✓			
[135]	27 marzo 2019	Caida Ucsd DDoS Attack 2007, DARPA DDoS attack 1998 e customized dataset	BNN, LSTM RNN	Python, TensorFlow		✓	✓	✓	✓					
[28]	11 giugno 2019	CIC-DoS, CICIDS2017, CSE-CIC-IDS2018, e customized dataset	RFECV, RF, DT ,LR ,SGD ,Perceptron ,AdaBoost	Scikit-learn ?			✓	✓	✓	✓		✓		
[136]	2019	CICIDS2017	SAE- 1 SVM	?		✓	✓	✓	✓					
[137]	26 febbraio 2020	KDD CUP 1999	LS-SVM con SNORT IPS	SNORT IPS		✓	✓		✓					✓
[138]	29 febbraio 2020	CICDDoS2019	LSTM	Keras		✓								
[139]	31 ottobre 2020	customized dataset	SAE	TensorFlow			✓	✓	✓					
[140]	2 dicembre 2020	CICDDoS2019	CNN, in particolare ResNet18	?		✓	✓	✓	✓					
[29]	2020	CICIDS2017	NN	?					✓					
[51]	2020	CICDDoS2019	BayesNet, Bagging, kNN , SMO, Simple Logistic	Weka			✓	✓	✓	✓				

Tabella 5.5: Strumenti, datasets, e algoritmi adoperati in varie ricerche. I punti di domanda indicano la mancanza di informazioni da parte degli autori su determinati strumenti. Per gli acronimi usati, consultare il Glossario 9.

5.3.1 Python

Quando si parla di Machine Learning, uno dei linguaggi di programmazione più utilizzati e famosi nel campo, è sicuramente Python. Questa fama è stata costruita (in parte) grazie alla disponibilità e facilità di utilizzo di librerie per l'analisi dati e, appunto, di algoritmi di ML/DL. Libreria frequentemente riscontrabile nel campo di ricerca attuale, è sicuramente Scikit-learn.

Scikit-learn

Libreria progetto nata nel 2007, resa poi pubblicamente disponibile nei primi mesi del 2010. Offre ad un pubblico non esperto sul ML, l'implementazione di vari algoritmi di ML, in un contesto di programmazione di alto livello[12]. L'uso di questa libreria con strumenti come Matplotlib ed IPython, offrono un ambiente di lavoro simile a quello di Matlab[41]. Fra i lavori citati nella sezione 3.1, ne fanno uso: [28], e [22].

Keras [45]

Deep Learning API di tipo open-source, scritta in Python e che utilizza la piattaforma TensorFlow. A titolo informativo, i ricercatori in [34], [22], [138] ne fanno utilizzo.

5.3.2 TensorFlow [141]

Piattaforma end-to-end ed open-source, per la creazione e distribuzione di modelli di ML, sviluppata da Google. Come linguaggio di programmazione, è possibile utilizzare oltre a Python (quella raccomandata) 5.3.1, C++, Java, Go, Swift(Early Release), e molte altre[13]

5.3.3 MATLAB [142]

Software closed-source che offre un ampia gamma di funzionalità, incentrate per lo più sull'analisi dati e calcoli numerici. Descrivere sinteticamente questo software è al quanto complicato, data l'enorme quantità di funzionalità offerte, fra di queste possiamo elencare: sviluppo di algoritmi, modellazione fisica, analisi dei dati, modellazione matematica, progettazione e simulazione di sistemi, ecc. [14]
Fra i vari strumenti messi a disposizione, è presente anche *MATLAB per il machine learning*, che secondo il sito (citando letteralmente) offre: *"Migliore performance di esecuzione rispetto a piattaforme open source sulla maggior parte dei calcoli statistici e di machine learning"*. La scelta fra software di tipo open o closed, è lasciata totalmente alle preferenze personali del lettore, varie sono le argomentazioni a favore di una rispetto all'altra, la quale trattazione di esse, va ben oltre allo scopo di questo testo.

5.3.4 Weka [143]

Software open source scritto in Java, che mette a disposizione vari strumenti correlati al ML e DL, disponibile sia come applicazione corredata da GUI, oppure usufruibile tramite terminale o Java API. Grazie all'API è inoltre integrabile con Python, Spark e R. Fra i punti forti di questa applicazione, possiamo elencare:

- Programmazzione non richiesta: Citando letteralmente il sito [143]: *"Weka can be used to build machine learning pipelines, train classifiers, and run evaluations without having to write a single line of code"*, ovvero, non sono richieste nozioni di programmazione da parte dell'utente.

- Modularità: L'utente può creare processi custom, partendo dagli algoritmi di apprendimento di base, e dagli strumenti offerti dalla piattaforma.[144]

[12] per maggiori informazioni, visitare il sito Scikit-learn-User-Guide
[13] Documentazione dell' API
[14] per ulteriori informazioni sulle soluzioni offerte da MATLAB.

- GUI: La piattaforma/applicazione offre varie interfacce grafiche, rendendo la costruzione dei modelli più intuibile per l'utente finale.

Consultando la Tabella 5.5, si nota che ne fa utilizzo la ricerca [51].

5.4 Algoritmi analizzati

5.4.1 Introduzione

Consultando le varie ricerche presenti in letteratura, è possibile individuare vari gruppi di algoritmi, che vengono utilizzati frequentemente. Il gruppo da noi identificato, e che andremo a discutere, è composto dai seguenti elementi: Naïve Bayes, Decison Trees, Random Forests, Support Vector Machines, Stacked Autoencoders, Recurrent Neural Networks, Long Short-Term Memory (alcune volte raggruppata insieme alle RNN), Bayesian Networks, ed infine K-Nearest Neighbourhood. La scelta è stata effettuata dopo aver consultato i lavori più popolari risultanti nuovamente dalla query *"Dos DDoS Machine Deep Learning"* eseguita tramite il motore di ricerca *Google Schoolar*, la lista che si va a definire risulta simile a quella proposta nella Tabella 5.5, con l'aggiunta qui di altri lavori, ed il numero di citazioni associati ad essi: Tabella 5.6.

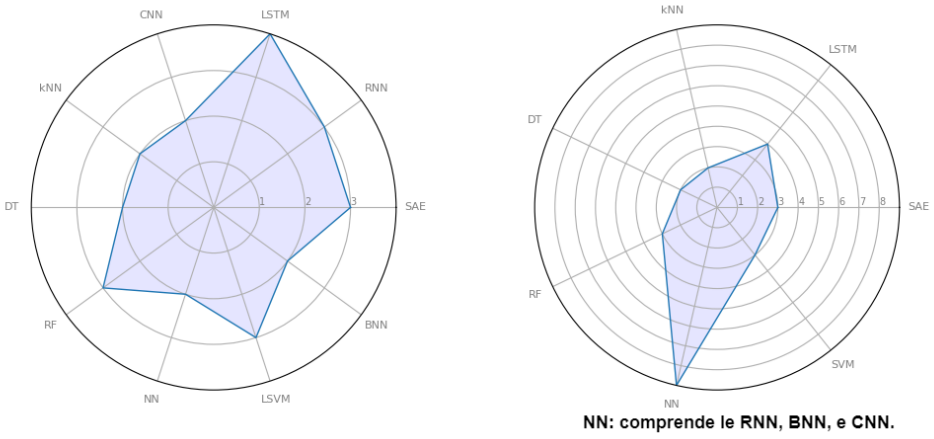


Figura 5.9: Frequenza delle metodologie adoperate nei lavori analizzati in Tabella 5.6. A sinistra le metodologie hanno un grado di suddivisione in più, mentre a destra all'interno della famiglia delle reti neurali NN, sono state unite le RNN,BNN e CNN.

Un gruppo simile a questo, a titolo informativo, è stato inoltre individuato anche dagli autori di [35].
Premettendo che uno studio approfondito di questi temi richieda un ampio spazio, che coinvolge pesantemente l'uso di strumenti matematici e statistici, decidiamo di eseguire un'analisi dai tratti generici, rendendo la trattazione una sorta di infarinatura sui concetti e tecniche utilizzati. Lo scopo finale infatti, è la preparazione ed indirizzamento per eventuali ricercatori, sulla scelta degli algoritmi più pertinenti. Oltretutto, riteniamo che ad accompagnamento ai temi citati, sia necessaria l'esperienza del ricercatore sull'utilizzo di queste tecniche[15]. Concludendo, scegliamo di esporre i vari algoritmi, seguendola struttura rappresentata in Figura 5.10.

5.4.2 Naïve Bayes (NB)

Algoritmi supervisionati che si basano sull'applicazione del teorema di Bayes, la traduzione letterale di "Naïve Bayes" è "Bayes ingenuo", dove il termine ingenuo va ad indicare l'adozione di un'assunzione.

[15] Seguendo la filosofia: "Experience is the best teacher", ovvero "L'esperienza è il miglior insegnante".

ID	Nome	Anno	Numero citazioni	Metodologia	CODICI	DISPONIBILE
[22]	Machine learning ddos detection for consumer internet of things devices.	2018	269	kNN,LSVM,DT,RF,NN	5,9,6,7,8	✓
[33]	A deep learning based DDoS detection system in software-defined networking (SDN).	2016	158	SAE	1	✓
[145]	DeepDefense: identifying DDoS attack via deep learning.	2017	140	RNN	2	
[34]	Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN.	2017	97	RNN LSTM CNN	2,3,4	✓
[28]	Smart detection: an online approach for DoS/DDoS attack detection using machine learning.	2019	41	RFECV, RF, DT, LR, SGD, Perceptron, AdaBoost	7,6	✓
[35]	A deep learning based intelligent framework to mitigate DDoS attack in fog environment.	2018	34	LSTM	3	✓
[29]	DeepDetect: detection of distributed denial of service attacks using deep learning.	2020	30	NN	8	✓
[139]	Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN.	2020	29	SAE	1	✓
[134]	Detection of DNS DDOS attacks with random forest algorithm on spark.	2018	15	RF	7	✓
[135]	Supervised learning-based DDoS attacks detection: Tuning hyperparameters.	2019	13	BNN, LSTM RNN	10,3,2	✓
[136]	A deep learning approach combining auto-encoder with one-class SVM for DDoS attack detection in SDNs.	2019	4	SAE- 1 SVM	1,9	✓
[137]	An effective mechanism to mitigate real-time DDoS attack.	2020	4	LS-SVM	9	✓
[51]	Network Intrusion Detection for Distributed Denial-of-Service (DDoS) Attacks using Machine Learning Classification Techniques.	2020	3	BayesNet, Bagging, kNN, SMO, Simple Logistic	10,5	✓
[140]	IoT DoS and DDoS Attack Detection using ResNet.	2020	2	CNN, in particolare ResNet18	4	✓
[138]	DoS and DDoS attack detection using deep learning and IDS.	2020	0	LSTM	3	✓

Tabella 5.6: Tabella utilizzata per individuare i modelli maggiormente utilizzati. La colonna codici serve per codificare la metodologia adoperata, e.g.: il numero "1" identifica gli SAE, il numero "2" le RNN, e così via. Infine la colonna disponibile serve a identificare se il lavoro è disponibile alla consultazione gratuitamente e pubblicamente. Il numero di citazioni è aggiornato al giorno 19/06/2021.

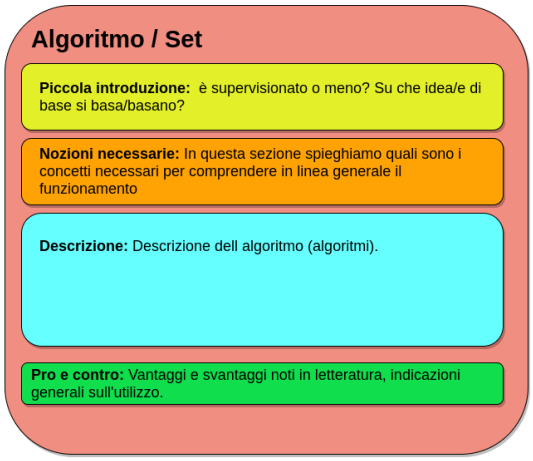


Figura 5.10: Architettura seguita nei paragrafi per l'esposizione degli algoritmi.

- **teorema di Bayes:**[16] Teorema che calcola la probabilità di un evento, tramite la conoscenza a priori, delle condizioni che potrebbero influire su di esso (in termini statistici, questa viene chiamata con "probabilità condizionata" o anche "probabilità a posteriori"). La notazione usata classicamente per indicare una probabilità condizionata è:

$$p(x|y) \tag{5.12}$$

$$\underbrace{p}_{\text{probabilità dell'}} \quad (\quad \underbrace{x}_{\text{evento x si verifichi}} \quad | \quad \underbrace{y}_{\text{sapendo che la condizione y si è verificata}} \quad) \tag{5.13}$$

Il Teorema è descrivibile matematicamente dalla seguente equazione:

$$\underbrace{p(x|y)}_{\text{prob. a posteriori}} = \frac{p(y|x)p(x)}{p(y)} \tag{5.14}$$

Facciamo un semplice ma intuitivo esempio sulla possibile applicabilità del teorema:
Abbiamo una scatola contenente 2 palline di colore rosso e 4 palline verdi, per un totale

[16] Utile anche la spiegazione offerta da 3Blue1Brown.

di 6. Noi siamo interessati al termine $p(x|y)$, che in questo caso possiamo scriverlo come $p(rossa|pescatoVerde)$ ed esprimibile con il quesito: Qual è la probabilità di pescare una pallina rossa sapendo che ho già pescato in precedenza una verde?
Per poter ora applicare il teorema di Bayes ci serve conoscere tre termini:

- $p(y|x)$ in questo caso è $p(verde|pescatoRossa)$: La probabilità di pescare una pallina verde sapendo che ne ho pescato in precedenza una rossa, ovvero $\frac{4}{5}$
- $p(x)$ in questo caso è $p(rossa)$: La probabilità di pescare una pallina rossa, ovvero $\frac{2}{6} = \frac{1}{3}$
- $p(y)$ in questo caso è $p(verde)$: La probabilità di pescare una pallina verde, ovvero $\frac{4}{6} = \frac{2}{3}$

Tramite il teorema:

$$p(rossa|pescatoVerde) = \frac{p(verde|pescatoRossa)p(rossa)}{p(verde)} = \frac{\frac{4}{5}\frac{1}{3}}{\frac{2}{3}} = \frac{2}{5} \tag{5.15}$$

Che è facilmente verificabile anche graficamente (Fig 5.11). Il posteriore $p(x|y)$ esprime dun-

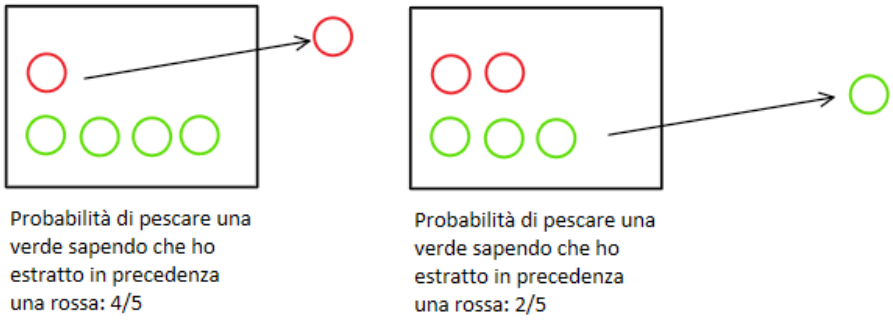


Figura 5.11: Un esempio sull'applicabilità del teorema di Bayes.

que quanto sappiamo su x dopo aver osservato y.

- **naïve:** L'ipotesi su cui si basano gli algoritmi NB, da cui deriva il termine naïve, consiste nel considerare le caratteristiche indipendenti fra di loro, ovvero, dato il vettore delle caratteristiche $X = (x_1, x_2, \ldots, x_n)$, e la classe C, la probabilità condizionata risulta essere:

$$p(X|C) = \prod_{i=1}^{n} p(x_i|C) \tag{5.16}$$

Proponendolo nel nostro contesto, l'indipendenza condizionata implica che le caratteristiche dei pacchetti del traffico (come lunghezza, dimensione, protocollo usato, ecc.) contribuiscono, alla stessa maniera sul giudizio del classificatore. Inoltre la presenza o assenza di una caratteristica non va ad influire sulla presenza od assenza delle altre. D'aiuto nella comprensione dell'ipotesi naïve può essere il caso proposto nella Figura 5.12.
Nonostante questa condizione nella realtà è il più delle volte violata, questi tipi di algoritmi, sono sorprendentemente efficienti per il ML.[36]

Collegando i concetti finora discussi, riprendiamo un vettore generico $X = (x_1, x_2, \ldots, x_n)$ e un set di classi binarie $C = [c+, c-]$; un algoritmo naïve Bayes parte dal omonimo teorema:

$$p(c|X) = \frac{p(X|c)p(c)}{p(X)} \tag{5.17}$$

Che indica la probabilità di appartenenza di un vettore X ad una classe $c \in C$. Il vettore X sarà classificato come appartenente alla classe $c+$ se e solo se:

$$\underbrace{f_b(X)}_{\text{classificatore bayesiano}} = \frac{p(c+|X)}{p(c-|X)} \geq 1 \tag{5.18}$$

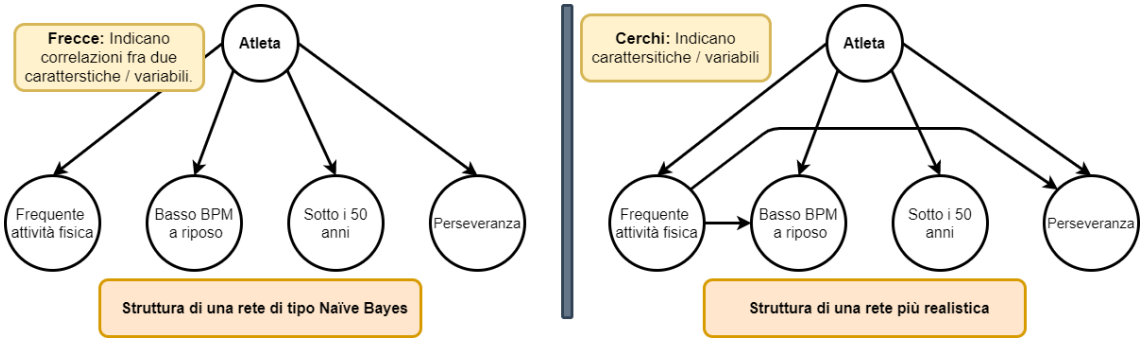


Figura 5.12: Grafici aciclici che rappresentano a sinistra una rete di tipo Naïve Bayes, mentre a destra una rete più simile alla realtà. Concentrandoci su una singola coppia di caratteristiche, il secondo caso risulta essere più realistico del primo, in quanto si presuppone che ad una frequente attività fisica, va ad accompagnarsi un battito cardiaco basso, nella condizione di riposo (ovvero sono condizioni correlate fra di loro).

In parole povere, se la probabilità di X ad appartenere a $c+$ è maggiore rispetto a quella di appartenere a $c-$, allora $X \in c+$. Ora applichiamo l'assunzione naïve, ovvero, assumiamo che gli attributi del vettore $X = (x_1, x_2, \cdots, x_n)$ siano indipendenti dalla generica classe C.

$$p(X|C) = p(x_1, x_2, \cdots, x_n|C) = \prod_{i=1}^{n} p(x_i|C) \tag{5.19}$$

Il classificatore di Bayes naïve è di conseguenza:

$$\underbrace{f_{nb}(X)}_{\text{naïve Bayes}} = f_b(X)\frac{p(X|c+)}{p(X|c-)} = \frac{p(c+|X)}{p(c-|X)}\prod_{i=1}^{n}\frac{p(x_i|c+)}{p(x_i|c-)} \tag{5.20}$$

o anche

$$f_{nb}(X) = \arg\max_{C} p(C)\prod_{i=1}^{n} p(x_i|C) \tag{5.21}$$

In base al tipo di distribuzione che si ipotizza avere $p(x_i|C)$, si hanno vari classificatori, e.g. Gaussiani, multinomiali, di Bernoulli, ecc.
Vantaggi:[37]

- **Flessibile:** Anche nel caso in cui l'assunzione naïve non è applicabile, questi classificatori presentano ottime prestazioni ed alti livelli di competizione con altri algoritmi di classificazione.
- **Probabilità in uscita:** Dato che l'algoritmo "ragiona" in termini di probabilità, la funzione in uscita sarà anch'essa espressa come probabilità. Viene data dunque al ricercatore una misura sul grado di confidenza con qui l'algoritmo classifica un determinato vettore x.
- **Compensazione delle classi sbilanciate:** Nello scenario di un dataset sbilanciato (vedi sezione 5.2.2) questi algoritmi possono applicare il cosiddetto "trucco della probabilità ridimensionata"[17]. Per la fase di training utilizziamo un dataset bilanciato, dal quale il modello ricava la $p_{\text{bil}}(C|x)$, mentre per la fase di testing, utilizziamo il dataset originale ottenendo la $p_{\text{vera}}(C|x)$ attraverso i seguenti passaggi:

$$\underbrace{p_{\text{bil}}(C|x)}_{\text{Fase di training}} = \frac{p(x|C)p_{\text{bil}}(C)}{\underbrace{p(x)}_{\text{data x, questa è costante}}} \propto p(x|C)p_{\text{bil}}(C) \Longrightarrow p(x|C) \propto \frac{p_{\text{bil}}(C|x)}{p_{\text{bil}}(C)} \tag{5.22}$$

[17] scaled likelihood trick

ora è possibile calcolare in fase di testing la $p_{\text{vera}}(C|x)$

$$\underbrace{p_{\text{vera}}(C|x)}_{\text{Fase di testing, siamo interessati a questa}} = \frac{p(x|C)p_{\text{vera}}(C)}{\underbrace{p(x)}_{\text{data x, questa è costante}}} \propto \overbrace{p(x|C)}^{\text{che per la 5.22}} \; p_{\text{vera}}(C) \propto \frac{p_{\text{bil}}(C|x)}{p_{\text{bil}}(C)} p_{\text{vera}}(C) \tag{5.23}$$

Svantaggi:

- **Dimensionalità del dataset:** Per ottenere dei buoni risultati richiede un numero elevato di istanze/vettori all'interno del dataset usato.[38]

5.4.3 Decision Tree (DT) [39][40]

Metodi supervisionati e non parametrici, usati per problemi di classificazione e regressione. Basano l'azione di classificazione mediante l'uso di alberi di decisione, creati dall'analisi dei vettori presenti nel dataset. Fra gli algoritmi DT più utilizzati si riscontrano: ID3, C4.5, C5.0 e CART.

- **Alberi di decisione:** Grafo strutturato verticalmente, dove vengono indicate le possibili scelte e le relative conseguenze in una struttura ad "albero rovesciato". Graficamente questi grafi sono simili alla visualizzazione dello sviluppo sotterraneo delle radici di un albero (Figura 5.13), ma si preferisce usare il termine "albero rovesciato" perché è più comodo adottare la terminologia botanica: i nodi del grafo terminali prendono il nome di foglie (leafs), il nodo padre ,posto in alto, prende il nome di radice (root), mentre i nodi interni vengono chiamati nodi di decisione o a volte semplicemente, nodi.

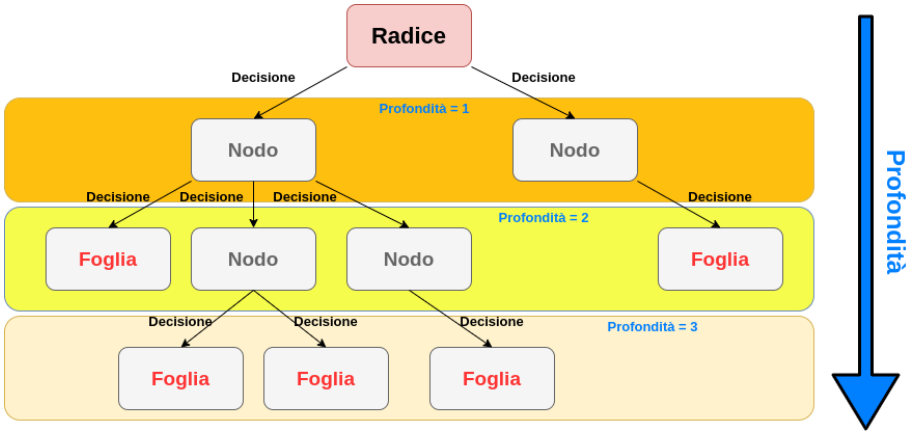


Figura 5.13: Terminologia usata spesso con gli alberi di decisione.

- **Impurità:** è una metrica atta a descrivere la purezza di un insieme, dove con purezza viene intesa come misura d'una corretta classificazione. Fra le più utilizzate troviamo "Impurità di Gini" (Gini impurity, o anche Gini index) utilizzata dall'algoritmo CART e "Guadagno d'informazione" (Information gain) usata da ID3, C4.5 e C5.0.

Matematicamente questi algoritmi seguono i seguenti passi:
Identifichiamo ora il generico vettore i del dataset con $x_i \in X$ dove $i = 1, 2, \cdots, n$, e con $c \in C$ le classi in esso definito. I dati che definiscono il generico nodo m lo indichiamo con Q_m che è funzione di due parametri j, t_m, ovvero dalla j-esima componente del vettore x, e da una soglia t.

$$Q_m^{\text{sinistra}}(j, t_m) = \{(x, c)|x_j \leq t_m\} \tag{5.24}$$

$$Q_m^{\text{destra}}(j, t_m) = \{(x, c)|x_j > t_m\} \tag{5.25}$$

La decisione effettuata al nodo Q_m è determinata in base al j-esimo vettore e ad una generica soglia t. La qualità della decisione dei nodi creati (Q_m^{sinistra} e Q_m^{destra}), è determinata dopodiché da una funzione di impurità $H(Q_m((j, t_m)))$.

$$\underbrace{G(Q_m, (j, t_m))}_{\text{Qualità della divisione al nodo } Q_m} = \frac{\#\text{ vett. a sinistra}}{\#\text{ totali}} \overbrace{H(Q_m^{sinistra}(j, t_m))}^{\text{"purezza" della divisione verso sinsitra}} + \frac{\#\text{ vett. a destra}}{\#\text{ totali}} \underbrace{H(Q_m^{destra}(j, t_m))}_{\text{"purezza" della divisione verso destra}}$$

Più chiaramente, senza le annotazioni all'interno della formula:

$$G(Q_m, (j, t_m)) = \frac{\#\text{ vett. a sinistra}}{\#\text{ totali}} H(Q_m^{sinistra}(j, t_m)) + \frac{\#\text{ vett. a destra}}{\#\text{ totali}} H(Q_m^{destra}(j, t_m)) \tag{5.26}$$

I parametri ottimali $(j, t_m)^*$ sono quelli che minimizzano la funzione $G(Q_m, (j, t_m))$:

$$(j, t_m)^* = \arg\min_{(j, t_m)} G(Q_m, (j, t_m)) \tag{5.27}$$

Per aiutare il lettore a comprendere meglio i passi seguiti dall'algoritmo, cerchiamo di spiegarlo in parole povere: il nodo radice riceve l'intero dataset X, pone una domanda di tipo "vero o falso" (equazione 5.26) riguardante uno degli attributi del generico vettore x e divide i vettori $x \in X$ in base alla risposta. In base alla qualità della domanda posta ($Q_m(j, t_m)$), vi saranno divisioni più o meno efficaci (calcolate dalla funzione $G(Q_m, (j, t_m))$). Il processo viene poi ripetuto iterativamente per ogni caratteristica del vettore x.[18]

Vantaggi:

- **White box e Visualizzazione:** Opposto al modello della scatola nera (Vedi la nota citata nel capitolo 5.1.2), questi algoritmi sono di facile interpretazione; la classificazione è infatti riducibile a delle semplici scelte binarie, rappresentabili tramite grafi.

- **Scelta automatica delle caratteristiche determinanti:** Le caratteristiche non fondamentali o affette da rumore vengono scartate automaticamente dai DT. Il processo di scelta delle caratteristiche utili per la corretta classificazione è infatti una qualità intrinseca delle DT.[41]

Svantaggi:

- **Tipicamente non competitivi:** Rispetto ad altri algoritmi supervisionati, presi singolarmente, ovvero senza applicare delle modifiche alla metodologia usata[19], i DT sono meno preformanti.

- **Instabilità:** Se il dataset cambia anche di poco, il DT novello che si va a generare può differire di molto rispetto a quello precedente.

5.4.4 Random Forest (RF) [42]

"Foresta di alberi casuali di decisione", questi algoritmi si basano sull'introduzione di componenti casuali nella costruzione degli alberi di decisione, effettuandone una media sulle relative predizioni per poi eseguire la classificazione finale. Metodi che utilizzano molteplici classificatori per poi unire in qualche maniera il loro risultato, prendono il nome di "ensemble methods" (metodi d'insieme).

[18] Potrebbe essere di aiuto al lettore il video offerto da Google Developers.

[19] con modifiche si intende generalmente l'aggregazione di molteplici DT attraverso metodologie come le random forest, bagging, boosting.

- **Variabili indipendenti e identicamente distribuite (IID o iid)**: Una variabile α si dice iid rispetto ad una diversa variabile β, se ha la stessa distribuzione di probabilità ed è mutuamente indipendente. Prendiamo in esame il lancio di un dado a 6 facce, in questo caso, gli eventi [esce la faccia x_i] con $i = 1, \ldots, 6$ sono delle iid. Ogni faccia ha infatti la stessa distribuzione di probabilità delle altre, e l'evento [esce la faccia x_3] non influisce sull'evento [esce la faccia x_6] o [esce la faccia x_1], ecc.

- **Introduzione di componenti casuali:** Sono le perturbazioni applicate agli alberi facenti parte della foresta. Queste azioni puntano a rendere più robusti al rumore gli alberi della foresta, diminuire la loro correlazione, ed aumentare così le performance del classificatore finale [43] . Fra le perturbazioni più comuni ci sono:
  -Alterare il set di training scegliendo a caso i vettori che ve ne fanno parte (bagging).
  -Rendere casuale la scelta su come si divide l'albero (random split selection).
  -Rendere casuale la scelta delle caratteristiche utilizzate per la divisione.

Essendo la trattazione matematica delle RF abbastanza estesa per essere discussa qui, e dato che le Random Forest si basano su classificatori precedentemente introdotti, abbiamo valutato di ometterla. Leo Breiman[42] definisce le Random Forest come: "Un classificatore RF consiste da un insieme di alberi di decisione $\{h(x, \Theta_k), k = 1, \cdots\}$ dove Θ_k sono vettori casuali indipendentemente ed identicamente distribuiti, in cui ogni albero vota la classe più popolare per il vettore di ingresso x".
La procedura comune su cui si basano può essere descritta dalle seguenti fasi:
- Vengono costruiti k alberi di decisione, ognuno accompagnato da un vettore casuale Θ_k (dove ogni Θ_k con $k = 1, \ldots$ è iid rispetto gli altri).
- Si creano così k classificatori identificati dalla funzione $h(x, \Theta_k)$ dove x è un vettore di ingresso.
- Una volta che si è soddisfatti dal numero di alberi creati, inizia il processo di votazione, dove ogni albero vota per la classe più popolare (probabile) per il vettore x.
- La procedura viene ripetuta per tutti gli altri vettori di ingresso $x \in X$.
Per ulteriori informazioni, consigliamo anche il lavoro svolto da Gilles Louppe, specificamente nella sezione "4 random forests" [41].

Vantaggi:

▲ **Ereditarietà:** La base di questi metodi sono le DT, da queste ereditano la qualità "Scelta automatica delle caratteristiche determinanti".

▲ **Importanza dei vettori:** Le caratteristiche salienti dei vettori, che contribuiscono alla corretta classificazione, vengono individuati da questi algoritmi. Di conseguenza è possibile ridurre dimensionalmente i vettori facenti parte del dataset, una volta identificate le caratteristiche (se presenti) non determinati alla classificazione. O ancora,infierire nuove informazioni utili per la costruzione o raffinamento di modelli.

▲ **Vicinanza:** Questi algoritmi possono dare una misura sulla "vicinanza" fra diversi ingressi. Il termine "vicinanza" indica quanto sono simili due istanze agli occhi della foresta di alberi. Di conseguenza, tramite questa misura, è possibile fare predizioni su altri ingressi.

Svantaggi:

▼ **Interpretabilità:** Seguire le decisioni prese da un singolo albero è un compito semplice, mentre seguire la stesse decisioni di alberi plurimi e diversi fra di loro, è molto più difficile. Di conseguenza, la caratteristica d'interpretabilità che hanno gli alberi di decisioni, in questi algoritmi, tende ad essere persa.

5.4.5 Support Vector Machines (SVM)

Set di metodi per l'apprendimento supervisionato, utilizzati per problemi di classificazione, regressione e identificazione di outliers. Si basano sull'utilizzo di uno o molteplici iperpiani per classificare vettori rappresentati in spazi vettoriali di n dimensioni.

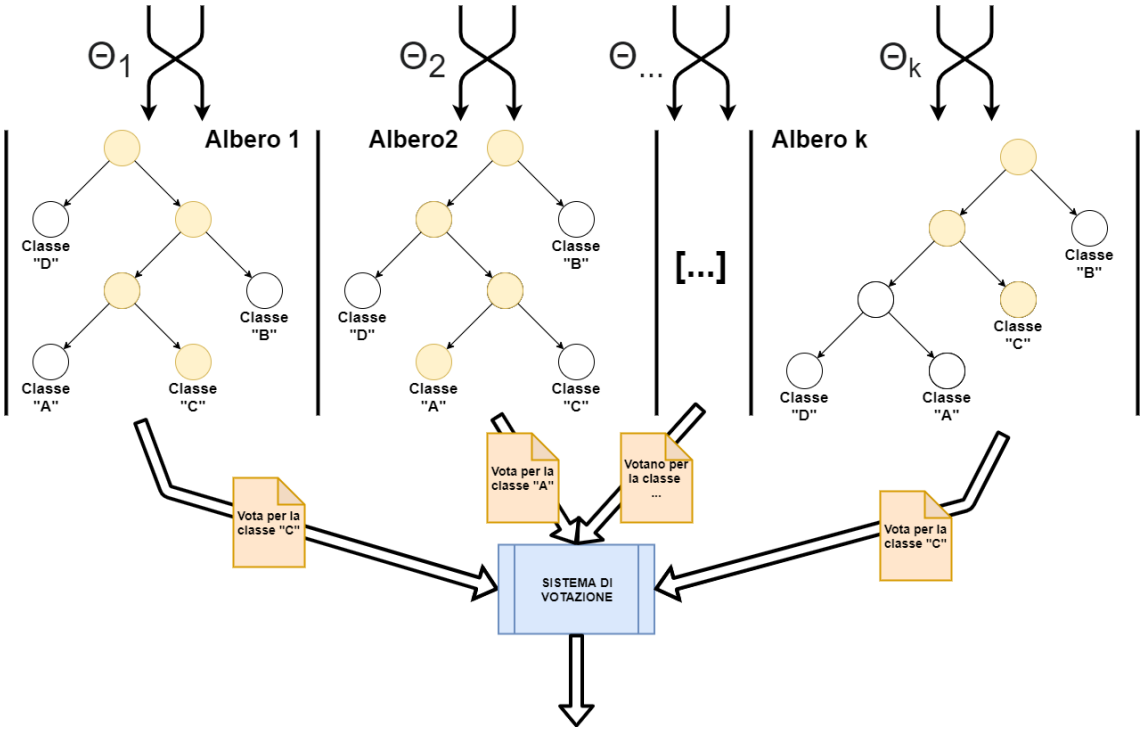


Figura 5.14: Schematizzazione di un algoritmo di tipo Random Forest.

- **iperpiano:** Prendiamo uno spazio di n dimensioni, un iperpiano è un sottoinsieme lineare di $n-1$ dimensioni. Per esempio, ponendoci un uno spazio bidimensionale, l'iperpiano è una retta, in uno spazio tridimensionale, l'iperpiano è un piano, e così via.
- **vettori:** Sono gli elementi di un determinato spazio di n dimensioni. Nel caso bidimensionale il punto identificato dalla coppia di numeri x, y è un vettore, nel caso tridimensionale un vettore è il punto identificato dalla generica triade x, y, z, etc.
- **margine:** è la distanza fra l'iperpiano di decisione e il vettore più vicino ad esso.
- **vettori di supporto:** Sono i vettori che descrivono, delineano la posizione del margine.
- **spazio degli input:** Spazio vettoriale dove vengono rappresentati i vettori di ingresso.
- **spazio delle caratteristiche, (feature space):** Spazio vettoriale dove vengono rappresentati i vettori di ingresso dopo aver applicato ad essi una generica trasformazione $\phi(\bullet)$
- **kernel:** Una funzione che ha come ingresso dei vettori appartenenti allo spazio degli input, e come uscita il loro prodotto interno (hermitiano) rappresentato nello spazio delle caratteristiche. Se abbiamo i vettori $\mathbf{a}, \mathbf{b} \in X$ dove X è lo spazio vettoriale "originario" dei dati di dimensione N, ed una funzione $\phi : X \to R^{N+c}$, allora $k(\mathbf{a}, \mathbf{b}) = \langle \phi(\mathbf{a}), \phi(\mathbf{b}) \rangle$ è una funzione kernel. Nell'eventualità di nuclei lineari, non è raro riscontrare modifiche alla nomenclatura delle SVM, con lo scopo di indicare tale proprietà. Osservando infatti la Tabella 5.6, troviamo la dicitura: LSVM, acronimo di "Support vector machine with linear kernel".

Dato che siamo interessati all'applicazione di questi algoritmi per risolvere un problema di classificazione, concentriamoci solo su questo aspetto. L'idea di base delle SVM è l'utilizzo di uno o più iperplani per effettuare una divisione fra dei punti (vettori) presenti in uno spazio di n dimensioni. La decisione se un dato vettore appartiene ad una determinata classe o meno, è dettata dalla sua posizione rispetto all'iperpiano. L'iperpiano è dunque il responsabile della classificazione, e la

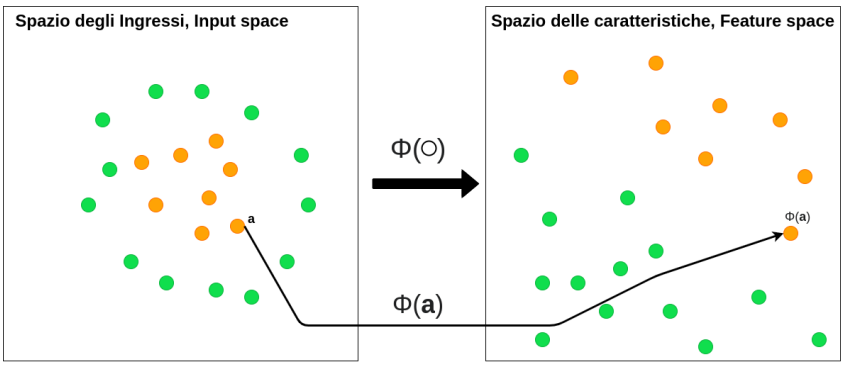


Figura 5.15: Rappresentazione degli spazi vettoriale di ingresso e delle caratteristiche

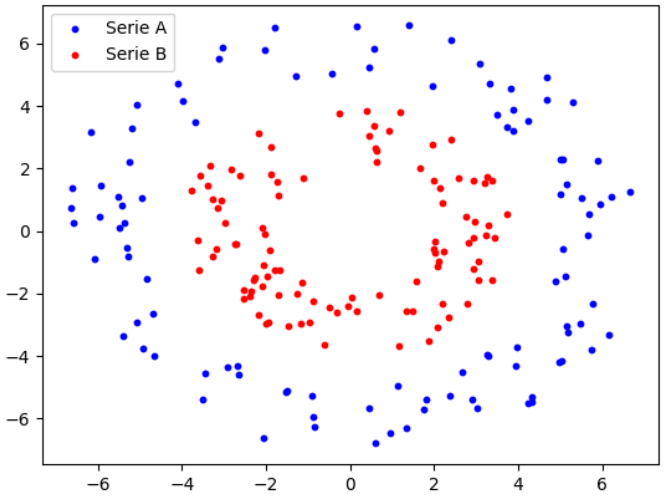


Figura 5.16: Esempio di dati non linearmente separabili.

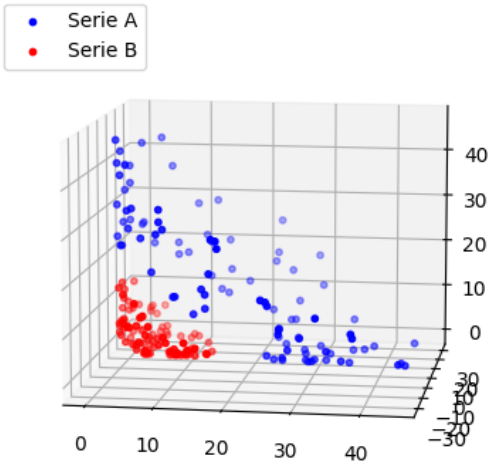


Figura 5.17: Gli stessi dati della Figura 5.16 dopo aver applicato la trasformazione $\phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$.

determinazione dello stesso è dettato dalla posizione dei suoi vettori di supporto.
Introduciamo un semplice esempio, esprimibile in un piano bidimensionale, per rendere facilmente rappresentabile e comprensibile il problema (Figura 5.18). Ipotizziamo di avere a disposizione una lista di persone, corredata da una triade di caratteristiche, ovvero, siamo in possesso di un dataset D contenente n persone p, in cui ogni persona p è individuata dalle seguenti tre caratteristiche: x_1, x_2, y_1. Per concretizzare il problema, ipotizziamo che le caratteristiche in questione siano i battiti cardiaci a riposo (x_1), l'età (x_2), e il rischio di infarto della persona(y_1), ed il nostro scopo sia di capire in base alle prime due caratteristiche, lo stato della terza. Innanzitutto, rappresentiamo le persone p in uno spazio bidimensionale, infatti, considerando solamente le prime due caratteristiche, risulta ovvia la loro rappresentazione in tale spazio: x_1 rappresenterà l'ascissa del punto, mentre x_2 l'ordinata (o viceversa). Per la terza caratteristica, ovvero l'obiettivo della classificazione y_1, utilizziamo la colorazione del punto nel grafico, se arancione il rischio è detto basso, mentre blu è detto alto. Avendo ora definito il problema, e la relativa possibile rappresentazione in Figura 5.18, dovrebbe risultare di facile comprensione il compito e decisione che l'algoritmo SVM attuerebbe per dividere i dati, ricordando l'introduzione di questo argomento: *"[...]Si basano sull'utilizzo di uno o molteplici iperpiani per classificare vettori rappresentati in spazi vettoriali di n dimensioni.[...]"*, l'SVM come iperpiano sceglierebbe la retta individuata sempre in Figura 5.18. In questo caso, la separazione che deve effettuare la SVM è banale, poichè i vettori presenti risultano essere linearmente separabili.
Ponendoci invece in una situazione come quella rappresentata dalla Figura 5.16, dove una separazione lineare risulta impossibile da ottenere, la ricerca di trasformazioni più "complesse"[20] risulta essere necessario (prima la trasformazione implicita era l'identità, non abbiamo dovuto infatti eseguire alcuna modifica ai dati d'ingresso), per cercare di rendere linearmente separabili, i vettori nel nuovo spazio vettoriale delle caratteristiche. Nel caso proposto in Figura (Figura 5.16), una

[20] Complesse rispetto a delle semplici trasformazioni lineari si intende.

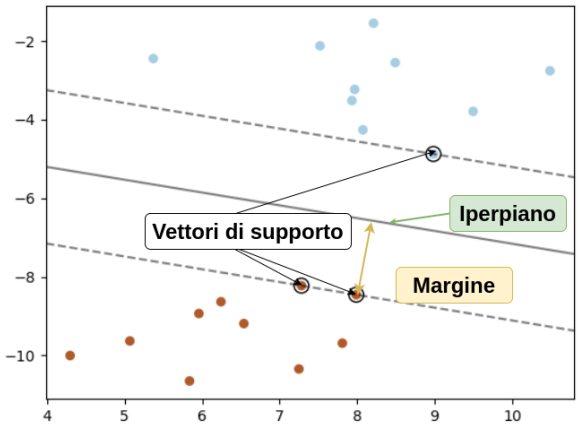


Figura 5.18: Esempio di uno spazio bidimensionale, linearmente separabile tramite l'uso di una SVM, codice disponibile su "SVM: Maximum margin separating hyperplane"

trasformazione che adempie allo scopo citato, è la seguente:

$$\phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix} \tag{5.28}$$

Che eleva l'originale spazio $\mathbb{R}^2$ ad uno delle caratteristiche $\mathbb{R}^3$, come notabile dal grafico in Figura 5.17.
Ora una domanda che potrebbe porsi il lettore è: "Ma se è possibile dividere i vettori tramite delle trasformazioni $\phi()$, perché utilizzare delle funzioni nucleo k?" Perché tramite l'utilizzo delle funzioni nucleo, riduciamo il calcolo dei nuovi vettori a dei semplici prodotti interni, senza nemmeno dover calcolare la generica trasformazione $\phi()$. L'utilizzo delle funzioni nucleo è equivalente alla separazione di vettori linearmente separabili, in un nuovo spazio vettoriale (Figura 5.15).
Riponendoci nel caso esposto in Figura 5.16 e 5.17, la funzione nucleo "associata" alla trasformazione $\phi()$ è :

$$\underbrace{k(\mathbf{a},\mathbf{b})}_{\text{funzione nucleo}} = \langle\phi(\mathbf{a}),\phi(\mathbf{b})\rangle = \phi(\mathbf{a})^\top \cdot \phi(\mathbf{b}) = \begin{pmatrix} a_1^2 & \sqrt{2}a_1a_2 & a_2^2 \end{pmatrix} \cdot \begin{pmatrix} b_1^2 \\ \sqrt{2}b_1b_2 \\ b_2^2 \end{pmatrix}$$

$$= a_1^2b_1^2 + 2a_1b_1a_2b_2 + a_2^2b_2^2 = (a_1b_1 + a_2b_2)^2$$

$$= \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix}^\top \cdot \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}\right)^2 = \underbrace{(\mathbf{a}^\top \cdot \mathbf{b})^2}_{\text{non serve calcolare la funzione } \phi(\mathbf{a}) \text{ o } \phi(\mathbf{b})}$$

Vantaggi:

- **Efficiente in memoria:** La funzione che determina come avviene la classificazione, è definita solamente da un sottoinsieme del set di training (ovvero, l'insieme dei vettori di supporto), rendendo questi tipi di algoritmi efficienti nell'uso della memoria. In altre parole, individuando con T il training set, e con S il support vector set, si ha $S \subseteq T$.
- **Unica soluzione:** Le SVM offrono un'unica soluzione, a differenza ad esempio delle reti neurali, dove sono disponibili molteplici soluzioni associati al minimo locale.
- **Non necessita l'esperienza dell'operatore per la scelta della trasformazione:** La funzione nucleo implicitamente contiene la trasformazione $\phi()$, quindi non è necessario fare

assunzioni su che genere di trasformazione possa rendere i vettori di ingresso, linearmente separabili.[102]

Svantaggi:

▼ **Stime di probabilità:** Direttamente le SVM non offrono le stime di probabilità di appartenenza ad una classe dei vettori. Queste devono essere calcolate attraverso altre metodologie.

5.4.6 Stacked Autoencoder (SAE)

Algoritmi non supervisionati facenti parte della famiglia del Deep Learning (DL) e dei metodi ensamble (uniscono più classificatori, in questo caso "impilandoli"[21]), il loro scopo principale è la riduzione dimensionale offrendo una "rappresentazione" del dataset obiettivo. Può essere vista come una rete neurale (NN) che cerca di copiare i sui ingressi alle sue uscite attraverso una compressione.

- **Autoencoder (AE):** Algoritmo non supervisionato che effettua un'astrazione sugli ingressi per ridurre la loro dimensionalità. Classicamente è composto da tre parti suddivise con questa nomenclatura: codificatore, codifica e decodificatore. La codifica viene effettuata tramite delle funzioni non lineari, le quali, solitamente, sono maggiormente preformanti rispetto ad altri metodi che si basano su funzioni lineari (come ad esempio la principal component analysis PCA) per estrarre delle caratteristiche salienti dei vettori[103].
  La struttura generale che seguono questa famiglia di algoritmi è simile a quella proposta in Figura 5.2; l'unica differenza da sottolineare risiede sulla dimensionalità del livello delle uscite. Gli AE cercano di riprodurre l'input, di conseguenza la dimensionalità fra il livello degli ingressi e quello delle uscite, deve essere identica; in parole povere, dato che ho x_1, x_2, x_3 devo necessariamente avere anche y_1, y_2, y_3). Matematicamente il livello di codifica (Hidden layer) è descritto dall'espressione:

$$\mathbf{h} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{5.29}$$

  Dove $\mathbf{W}$ è una funzione non lineare d'attivazione e $\mathbf{b}$ è un offset (bias), in poche parole, si può affermare che la codifica è determinata da una trasformazione non lineare sui vettori d'ingresso con in aggiunta un vettore di offset.

- **Spazio latente:** Nella trattazione delle SVM abbiamo introdotto il concetto di "spazio delle caratteristiche", il quale risulta utile alla macchina per individuare (separare) i vettori più facilmente. Uno spazio latente può essere visto semplicemente come uno spazio delle caratteristiche con la qualità aggiunta di essere di dimensioni ridotte rispetto allo spazio di input. L'altra particolarità di questo spazio è la relazione fra vicinanza fra i punti e la loro similitudine, per elaborare meglio, più due punti sono vicini nello spazio latente, e più questi si assomiglieranno nella realtà (Figura 5.19).

Vantaggi:

▲ **Non lineare:** I "blocchi" su qui si basano gli SAE (ovvero gli AE), hanno delle performance più elevate rispetto a quelli che usano metodi lineari per l'estrazione, come ad esempio, la principal component analysis (PCA)[104].

▲ **Non supervisionato:** Anche in presenza di dataset non etichettati, è possibile utilizzare questi algoritmi per estrarre informazioni importanti sul come procedere nel lavoro di classificazione.

Svantaggi:
Non abbiamo trovato informazioni rilevanti nei confronti di questo argomento, di conseguenza proseguiremo la discussione citando un lato negativo degli AE.

▼ **Specifici:** Una volta sottoposti al processo di trainig, gli AE non possiedono un grande grado di generalizzazione. Se i dati in ingresso differiscono di molto da quelli su cui il modello è stato allenato, quest'ultimo potrebbe non riuscire a soddisfare il livello di performance aspettato.

[21] "Stacked" in italiano significa impilato.

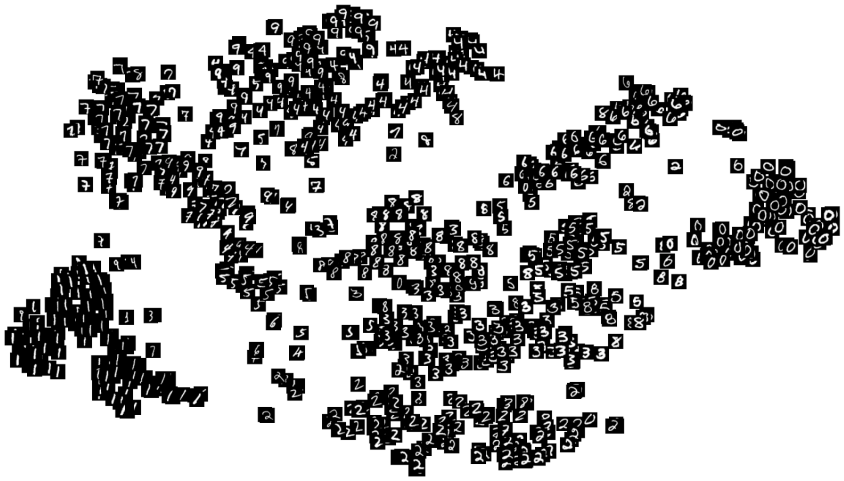


Figura 5.19: Esempio di uno spazio latente prodotto da t-SNE su un dataset contenente cifre numeriche scritte a mano. Immagine offerta da Julien Despois @juliendespois.

5.4.7 Recurrent Neural Network (RNN)[113]

Questi tipi di rete neurali (NN) sono dette ricorsive (R)[22] perché l'uscita non è solamente dipendente dallo stato degli ingressi attuali, ma anche da quelli precedenti. La ricorsione viene effettuata attraverso connessioni cicliche nel livello degli stati[23]. Per la loro natura, si prestano specialmente (ma non solo) per le serie storiche.

- **Serie storica:** Un'ottima definizione viene data da Dagum, E. Bee.[105]: "Una serie storica consiste di un insieme di osservazioni su un certo fenomeno ordinate nel tempo". Calandoci nuovamente nel nostro contesto, il traffico di rete costituisce una serie storica, una cattura di traffico è infatti rappresentabile come: $[x_1, x_2, \ldots, x_t]$ indicando con x_1 il primo pacchetto registrato dall'inizio della cattura, e con x_t quello registrato all'istante t.
- **Funzione di attivazione:** Sono funzioni utilizzate per lo più nelle reti neurali (NN), con il compito di trasformare il segnale d'ingresso in quello d'uscita; nella maggior parte dei casi la trasformazione applicata è non lineare[24]. Ad esempio, due funzioni molto comuni sono la sigmoide[106] e softmax (Figura 5.20). La funzione sigmoidea (sigmoid function), è caratterizzata dal fatto che il suo grafico ricorda la lettera "S" e la sua immagine è $[0, 1]$, mentre softmax è una combinazione di molteplici sigmoidee. Entrambe vengono consigliate per modelli che devono compiere una classificazione, di contro parte, se il modello è propenso al problema della scompara del gradiente, sono sconsigliate[25][107].

La presenza delle connessioni cicliche fra i livelli degli stati e l'uso di funzioni d'attivazione non lineari, rendono le RNN sistemi dinamici[26], e possessori di memoria. Modelli basati su queste reti, possono essere descritti dalle seguenti equazioni:[116]:

$$\mathbf{x}(t) = \mathbf{w}(t) + \mathbf{h}(t-1) \tag{5.30}$$

Il vettore $\mathbf{w}$ indica gli ingressi attuali nel sistema, non da confondere con il vettore $\mathbf{x}$ che indica invece, il livello degli ingressi. Il livello degli ingressi infatti, è la somma degli ingressi attuali, più il livello degli stati dell'istante precedente.

$$\mathbf{h}(t) = \underbrace{\sigma_1}_{\text{sigmoidee}} (\mathbf{x}(t)\mathbf{w}) \tag{5.31}$$

[22] Recurren Neural Network, in italiano si può tradurre con: reti neurali ricorsive
[23] chiamati con il nome hidden layers nell NN di tipo feed-foward (Figura 5.2).
[24] Spesso la lettera greca σ viene utilizzata per rappresentare queste funzioni.
[25] in particolare la funzione sigmoid e tanh
[26] In contrasto alle reti neurali feedfoward che sono funzioni.

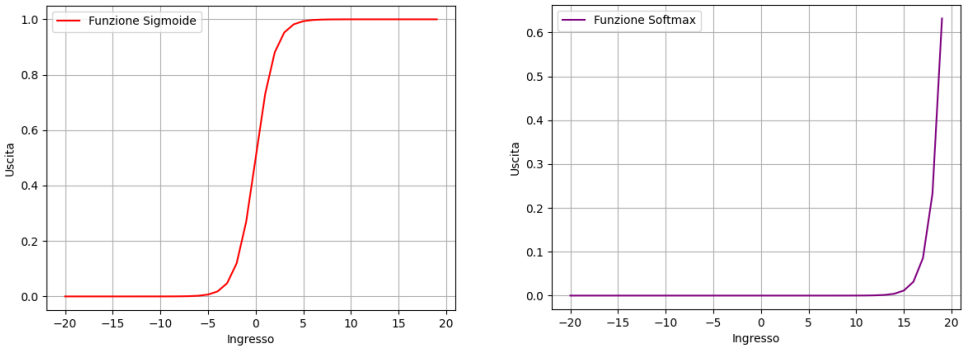


Figura 5.20: A sinistra la funzione sigmoidea, a destra la funzione softmax, con gli ingressi che variano da $[-20, 20]$. Entrambe vengono usate ampiamente come funzioni di attivazione.

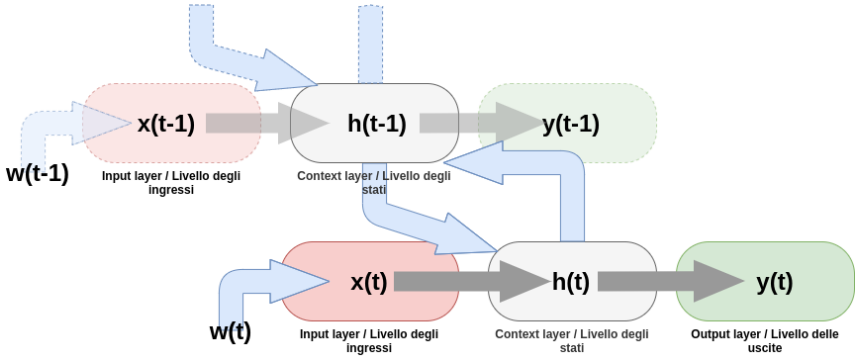


Figura 5.21: Schema di una RNN che segue la struttura di Elman [113]

$$\mathbf{y}(t) = \underbrace{\sigma_2}_{\text{softmax}} (\mathbf{h}(t)\mathbf{v}) \tag{5.32}$$

La Figura5.21 dovrebbe essere d'aiuto nel comprendere come queste reti adoperano.

Vantaggi:

▲ **Simulazione modelli dinamici** Nel 2005 è stato provato da [115] che i modelli dinamici possono essere approssimati, in un tempo finito e con un grado di accuratezza arbitrario, dalle RNN. Inutile sottolineare l'importanza di questa scoperta, data l'elevata presenza di modelli dinamici nel campo ingegneristico.

Svantaggi:

▼ **Scomparsa o esplosione del gradiente:** Dipendenze sul lungo periodo fra i dati non riescono essere prese in considerazione da questi tipi di algoritmi.[108][109].[27]

▼ **Difficoltà nel training:** Metodi come la discesa del gradiente sono di difficile applicazione su questi algoritmi. Durante il processo di apprendimento le RNN possono subire biforcazioni, che a loro volta, possono portare all'esplosione del gradiente[28]. Algoritmi diversi dalla discesa del gradiente possono essere utilizzati per la fase di training, tuttavia per essere applicate con buoni risultati, richiedono un livello di esperienza ed abilità da parte del ricercatore non indifferenti.[114]

[27] Problematica risolta poi con l'introduzione dell'LSTM.[113]
[28] Questo svantaggio è strettamente correlato con quello definito precedentemente.

5.4.8 Long Short-Term Memory (LSTM)[110][113]

In risposta alla problematica "Scomparsa o esplosione del gradiente" delle RNN, vengono in soccorso gli algoritmi LSTM. Categorizzabili come RNN, le LSTM aggiungono delle "unità di memoria" o "celle" che permettono di superare lo svantaggio "Difficoltà nel training" (Vedi Svantaggi RNN).

- **Porte:** In questo contesto, con porte intendiamo nodi dove le informazioni possono o meno passare, in base a determinate regole. Facendo un parallelo con i linguaggi di programmazione, sono simili espressioni condizionali *"if - else"*. Diciamo simili perché la decisione è correlata sempre ad una funzione d'attivazione classicamente non lineare, e.g. la funzione sigmoidea[29].

- **Pesi e biases:** Concetti fondamentali e che vanno oltre allo scopo di questo testo, sono i Pesi e biases. Per il momento basta al lettore sapere che verranno indicati con la lettera maiuscola W i pesi, e con la lettera minuscola b i bias. Sono dei valori numerici che vengono aggiornati iterativamente, durante il processo di apprendimento, per costruire il modello finale.

Facendo riferimento alla Figura 5.23 l'algoritmo può essere descritto dal seguente set di equazioni

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5.33}$$

Che descrive in che modo l'informazione viene mantenuta, questa è correlata alla "porta per dimenticare" e di conseguenza viene indicata con la lettera f (dall'inglese "**f**orget"). La porta degli ingressi produce invece due stati:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5.34}$$

$$L_t = \tanh(W_L \cdot [h_{t-1}, x_t] + b_L) \tag{5.35}$$

Che vanno tutti a contribuire all'aggiornamento dello stato interno C_t:

$$C_t = f_t * C_{t-1} + i_t * L_t \tag{5.36}$$

Ultimo passo, calcolare la nuova uscita h_t, attraverso l'omonima "porta delle uscite".

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5.37}$$

$$h_t = o_t * \tanh(C_t) \tag{5.38}$$

Nota: L'originale meglio dei suoi figli
Nonostante la concezione originale sia del 1995, secondo una ricerca condotta da [111] molte delle varianti proposte sulle LSTM, non migliorano significativamente le performance, rispetto all'architettura originaria.

Vantaggi:

▲ **Migliorie alle RNN "classiche":** Le LSTM risolvono le problematiche associate alle RNN, infatti, sono state create con quello stesso intento.

Svantaggi:

▼ **Non parallelizzabile:** La struttura intrinseca sequenziale delle RNN (vedi Figura 5.24), pone problemi alla loro parallelizzazione in caso di sequenze lunghe, rendendole computazionalmente intensive.[112]

[29] Mentre le espressioni *"if-else"* accettano qualunque condizione esprimibile tramite il linguaggio adoperato.

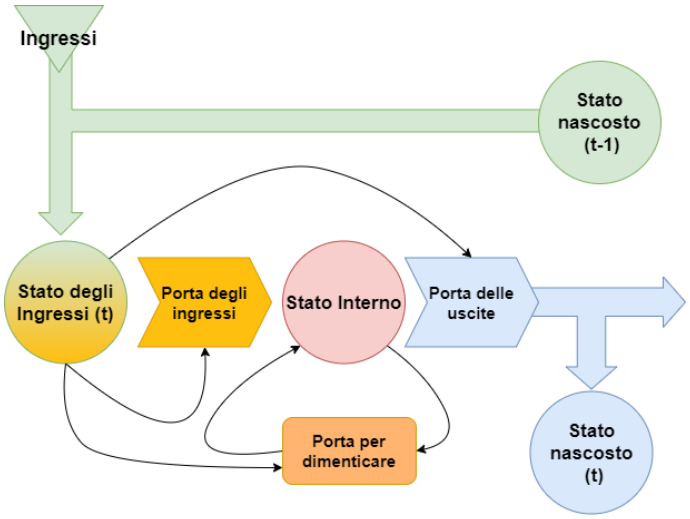


Figura 5.22: La struttura di una cella di memoria LSTM, così come proposta nel testo [113]

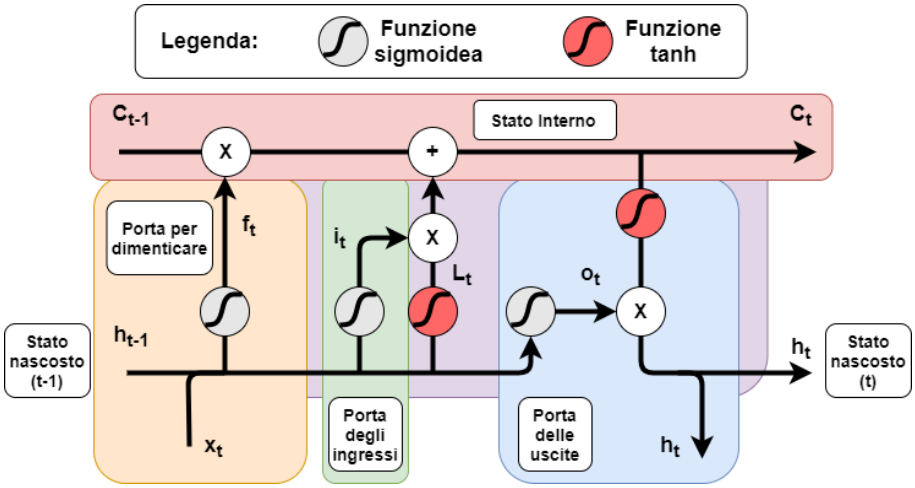


Figura 5.23: Un'altra rappresentazione di una cella LSTM, tratta spunto da Understanding LSTM Networks.

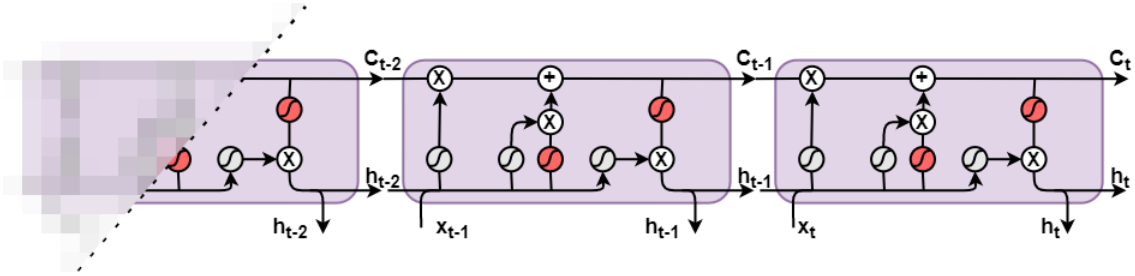


Figura 5.24: La struttura di una generica rete LSTM composta da più omonime celle.

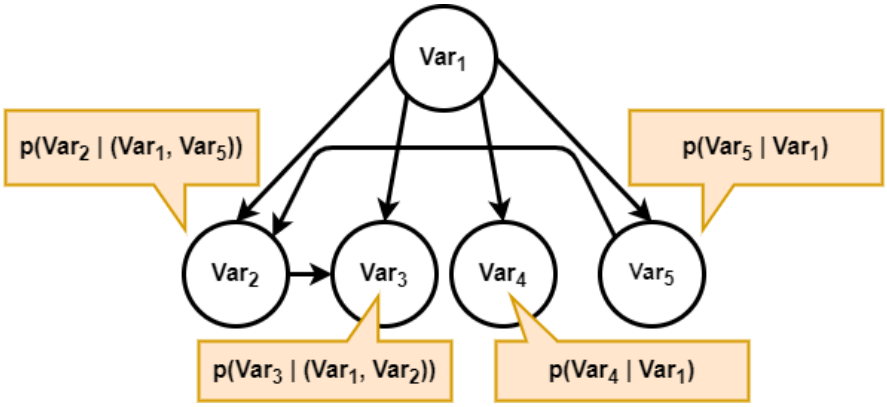


Figura 5.25: Rete bayesiana con le probabilità e le relazioni di tipo padre-figlio intercorrenti fra i nodi. Da notare che il rapporto di padre-figlio si ferma ad un "livello", la Var_3 non è infatti figlia della Var_1.

5.4.9 Bayesian Network (BN o anche BNN)

Riprendendo i concetti introdotti tramite la discussione sul NB, presentiamo altri algoritmi di tipo supervisionato, di simile natura. Questi risultano partizionabili in due macro fasi: una di ricerca della rete bayesiana [30] più adeguata per la descrizione del dataset utilizzato, ed una di apprendimento della tabella delle probabilità.

- **Reti Bayesiane:** Possono essere rappresentate, come già visto nel caso del NB, con dei grafici aciclici orientati. In questi grafici i nodi rappresentano le variabili aleatorie, mentre i vettori (le frecce) indicano una dipendenza dal punto di vista probabilistico (Figura 5.12). Questi reti ritornano utili anche in questo contesto, poiché soddisfano la proprietà di Markov locale.

  $$\text{Sia } A = [Var_1, \ldots, Var_n] \text{ un set di variabili} \tag{5.39}$$

  Una rete bayesiana, è un grafico aciclico orientato G con una tabella di probabilità:

  $$G_{tab} = \{p(a|\text{padri}(a))|a \in A\} \tag{5.40}$$

  dove padri(a) sono i nodi padre della variabile a.

- **Proprietà di Markov locale:** "Ogni variabile casuale è indipendente da quelle non discendenti da lei, dati le variabili padre"[118] in termini più semplici, ogni variabile aleatoria è legata probabilisticamente alle sue variabili aleatorie padre e figlia. Osservando il caso riportato nella Figura 5.25, si può dunque affermare dunque che Var_4 è indipendente, dalle Var_2, Var_3, Var_5. In altri termini:

  $$p(Var_4|Var_1) = p(Var_4|(Var_1, Var_2)) \tag{5.41}$$

Il problema di classificazione, usando questi concetti, diventa così, una ricerca dell'argomento che massimizza la probabilità $p(c|a)|a_1 \in A$. Ricordando come si legge la notazione (vedi teorema di bayes nella sezione naive bayes.), quale è la classe c che massimizza la probabilità d'appartenenza, sapendo che il vettore è composto dalle caratteristiche a_1 con $a_1 \subset [x_1, \ldots, x_n]$?

$$\arg\max_c p(c|a_1) \tag{5.42}$$

Avendo disponibile la rete bayesiana:

$$p(c|a_1) = p(A)/p(a_1) \tag{5.43}$$

[30] Anche chiamata, a volte, con: "Bayes nets", "Belief networks" e "Causal networks".

$$p(c|a_1) \propto p(A) \tag{5.44}$$

$$p(A) = \prod_{a_1 \in A} p(a_1|\text{padri}(a_1))) \tag{5.45}$$

Conoscendo tutte le variabili comprese in a_1, basta calcolare l'eq5.45, per ogni classe[31].
La ricerca della rete bayesiana più adeguata può avvenire seguendo diverse metodologie. Al momento, le più popolari sono l'utilizzo di punteggi locali come metrica, di test che verificano l'indipendenza probabilistica, oppure di punteggi globali.
La prima si basa sull'ottimizzazione di una misura di qualità, della rete bayesiana Q(Rete Bayesiana|Dataset). Misura poi decomponibile in una sommatoria o produttoria, a livello di singolo nodo, consentendo l'utilizzo di metodi di ricerca locale.
La seconda metodologia parte dal presupposto dell'esistenza di una rete bayesiana che descrive perfettamente le relazioni d'indipendenza generate dal dataset. Da questo presupposto, parte una ricerca per individuare le indipendenze condizionali fra i dati, con un assegnazione delle direzioni, affinché venga raggiunta una rappresentazione adeguata del dataset.
L'ultima metodologia, similmente alla prima citata, utilizza una misura di qualità della rete, con l'unica differenza di spostare la ricerca al livello globale. Le misure di qualità diventano dunque le metriche esposte nel Capitolo 5.2.3.

Vantaggi:

- **Performance sulle decisioni:** Nello studio condotto da [120] è stato dimostrato che queste reti hanno performance migliori rispetto alle DT, inoltre, sono più indicate per catturare decisioni complesse di difficile individuazione.

- **Stabilità:** Rispetto alle DT, un cambiamento leggero del dataset d'ingresso, non provoca forti cambiamenti della struttura del modello. Dato che le reti bayesiane si basano sulle relazioni di probabilità[32] intercorrenti fra le variabili, il problema del "variable masking" risulta essere attenuato rispetto alle DT. Con "variable masking" si intende: *"Se una variabile è altamente correlata con un altra, allora un piccolo cambiamento nei dati potrebbe spostare la scelta della divisione dell'albero, da una variabile, all'altra"* [120].

Svantaggi:[119]

- **Dimensioni e accuratezza:** In caso di dataset di dimensione elevata, questi algoritmi, non sono consigliati. Inoltre possono risultare non adatti se si necessita di predizioni accurate.

- **Retroazione:** Se il modello che stiamo andando a creare, ha delle retroazioni che sono importanti per la descrizione dello stesso, questi algoritmi vengono sconsigliati.

- **Complessità:** La ricerca della rete bayesiana più efficiente nella descrizione del modello in esame, è un compito difficile[122]. L'analisi dell'intero spazio delle soluzioni, sfruttando la "forza bruta" del calcolatore, può diventare presto insostenibile, infatti [121] ha dimostrato, che con n nodi, il numero di strutture da valutare diventano:

$$r(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} r(n-i) = n^{2^{O(n)}} \tag{5.46}$$

  Quindi, nel caso in cui, la rete necessiti di un numero maggiore di otto di nodi, una ricerca alla cieca senza euristiche, o altre metodologie, diventa computazionalmente intensa (riprendendo l'equazione 5.46, ed inserendo 8 nodi: $r(8) \simeq 78,37e^{10}$).

[31] L'implementazione proposta è quella adottata dallo strumento "Weka", abbiamo adottato questa scelta poiché, fra i lavori di ricerca consultati, risulta essere utilizzato quest'ultimo(documentazione su Weka).
[32] Tenendo sempre conto della proprietà di Markov locale.

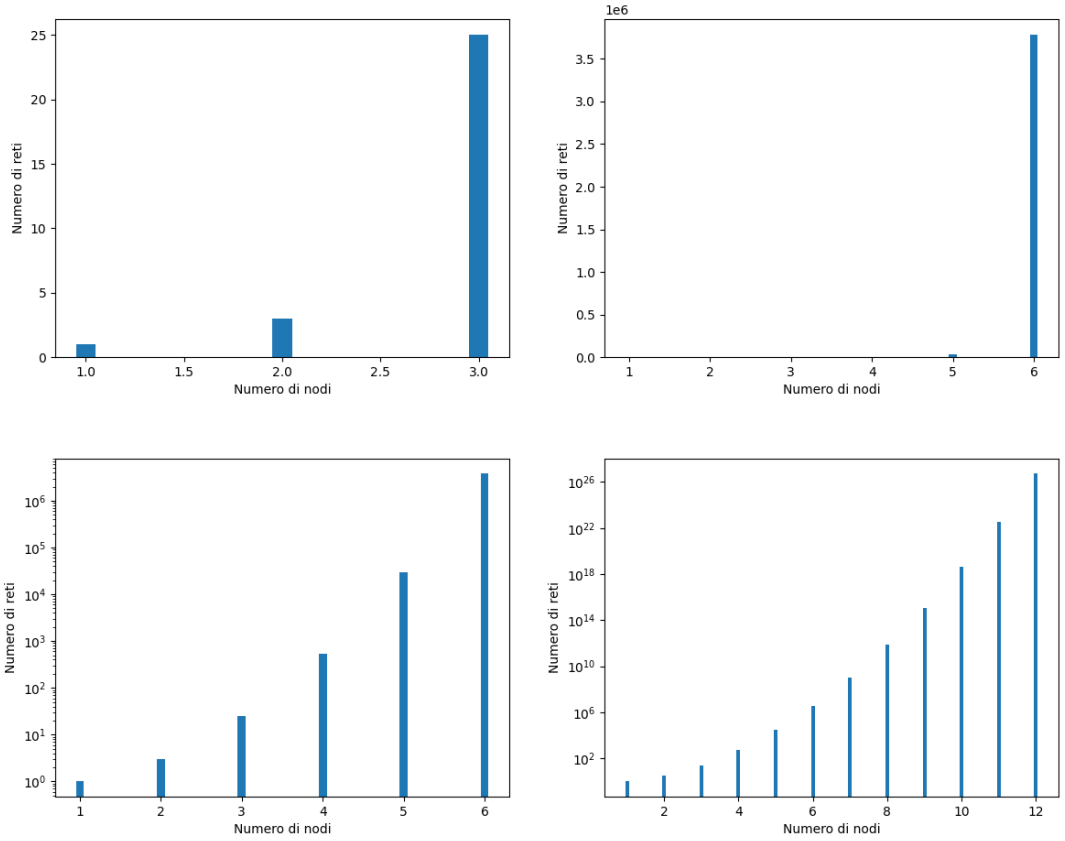


Figura 5.26: Progressione della dimensione dello spazio delle soluzioni. In alto la coppia di grafici adotta scale lineari in basso, per rendere l'idea della progressione della formula, la scala delle ordinate è logaritmica.

Nota: L'equazione 5.46
La sequenza dei numeri che si vanno a creare tramite questa equazione, partendo da 0 nodi, sono:

1, 1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343, 1213442454842881, 4175098976430598143, 31603459396418917607425, 521939651343829405020504063, 18676600744432035186664816926721, 1439428141044398334941790719839535103.

Se si è interessati a calcolare questa sequenza indipendentemente, il codice python è disponibile al sottocapitolo 7.2. (utillizzato per la creazione dei grafici in Figura 5.26) Altre proprietà su questa sequenza sono elencate su "The On-line Encyclopedia Of Integer Sequences (OEIS)".

▼ **Interpretabilità:** I risultati di questi algoritmi, sono delle reti bayesiane con tabelle di probabilità associate. Conseguentemente, il "ragionamento" alla base della scelta di una determinata classificazione può risultare complessa [122].[33]

5.4.10 k-Nearest Neighbourhood (kNN)

[33] D'altra parte, [119], cita l'interpretabilità come una caratteristica posseduta da questi algoritmi.

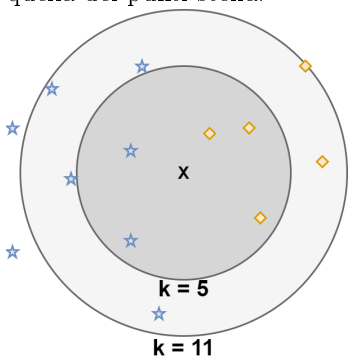
Uno dei più semplici algoritmi, dal punto di vista concettuale, di ML per la classificazione, è sicuramente kNN [123]. Ponendoci nuovamente in un generico spazio vettoriale per rappresentare i punti soggetti a classificazione: questo algoritmo, decide la classe d'appartenenza del generico punto x, in base alle classi dei k punti suoi più vicini. Da qui, il nome "k-esimo maggiormente vicino" (Figura 5.27).
Basandosi su spazi di rappresentazione dei dati, come quello latente, delle caratteristiche od euclidiano, ed essendo di semplice natura, non è necessario introdurre nuovi concetti per poterne spiegare il funzionamento. Gli unici due punti che richiedono una maggiore attenzione sono, la scelta della metrica utilizzata per definire la distanza fra i punti, ed la scelta del parametro k.
Ad esempio, la metrica di distanza più utilizzata risulta essere quella di tipo eucladiana [124], semplicemente esprimibile tramite la formula[125]:

$$D_{euclid}(\mathbf{a}, \mathbf{b}) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2} \tag{5.47}$$

o anche[124]:

$$D_{euclid}(\mathbf{a}, \mathbf{b}) = \sqrt{\frac{\sum_{i=1}^{n} (a_i - b_1)^2}{n}} \tag{5.48}$$

Figura 5.27: Esempio di una classificazione kNN, se il parametro k è uguale a 5, il punto x risulterebbe appartenente alla medesima classe dei punti contrassegnati dal rombo. Mentre con k uguale ad 11, la classe d'appartenenza di x, diventerebbe uguale a quella dei punti stella.



Con i punti $\mathbf{a}$ e $\mathbf{b}$, rappresentanti dei vettori ad n componenti (caratteristiche) soggetti al calcolo della distanza: $\mathbf{a} = [a_1, a_2, \ldots, a_n]$, $\mathbf{b} = [b_1, b_2, \ldots, b_n]$.
Data la natura della procedura di classificazione, ovvio che la scelta di una misura di distanza adeguata ricopre una grande ruolo. Conseguentemente, vari studi hanno ricercato la possibilità di automatizzare la scelta della misura di distanza più adeguata per il determinato dataset[34]. Fra questi, uno dei più famosi, risulta essere l'algoritmo proposto da Weinberger, Kilian Q., e Lawrence K. Saul[126]: "large margin nearest neighbor (LMNN)". Partendo dal presupposto che l'algoritmo kNN classifica correttamente un determinato punto, se i suoi k punti vicino condividono la stessa classe, vengono ricercate trasformazioni lineari applicabili allo spazio d'ingresso (similmente alla procedura seguita dalle SVM), perseguendo la massimizzazione di tale proprietà.
L'altro parametro rilevante di kNN è il numero di vettori da prendere in considerazione per la determinazione della classe, ovvero il parametro k. Un numero troppo basso di k porta ad un'eccessiva sensibilità al rumore, mentre un numero elevato, porta a confini poco definiti e perdita di precisione. Tipicamente, il numero ottimale viene derivato empiricamente, tramite l'esecuzione di diverse prove. Una metodologia comunemente adottata è il calcolo dell'errore commesso dal modello, per diversi k, partendo da 1, ad arrivare alla radice quadrata della cardinalità del dataset: (1-NN,2-NN,...,$\sqrt{n}$-NN , con n =numero istanze presenti nel dataset.)[127].

Nota: Limite superiore di k
Se si vuole evitare l'esplorazione di modelli con vari parametri k, una scelta molto comune, è utilizzare direttamente $\sqrt{n}$-NN, con n =numero istanze presenti nel dataset.).

Vantaggi:

- **Semplicità:** Nonostante la semplicità dell'algoritmo, molti problemi di classificazione risultano risolvibili egregiamente. Inoltre, è facilmente modificabile per assecondare eventuali esigenze particolari, anche da utenti non esperti.
- **Niente Training:** La procedura di classificazione, non ha bisogno di alcun periodo di training. L'assegnazione della classe al generico punto, si basa infatti sul calcolo di una distanza, conseguentemente, il modello è sempre pronto a classificare nuovi punti.

[34] Questo problema in inglese vine chiamato con il nome di "distance metric learning".

Svantaggi:

- **Scelta dei parametri:** Nel caso dell'algoritmo originale di kNN, la scelta sia della metrica di distanza, sia del parametro k, sono affidate all'operatore umano. Come citato precedentemente, questi parametri influenzano direttamente sulle performance del modello.
- **Importanza delle caratteristiche:** Nel processo di decisione, tutte le caratteristiche vengono prese in considerazione in egual modo. Conseguentemente, se vi sono molte caratteristiche non determinati alla classificazione, il modello potrebbe perdere accuratezza.[128]
- **Classi sbilanciate:** Derivante dalla semplice modalità di classificazione, questo algoritmo è molto suscettibile a dataset in cui vi sono classi sbilanciate.

5.5 Problematiche principali nella letteratura corrente

Premettendo che i lavori citati nella sezione 3.1 non costituiscono l'intero e vastissimo campo della individuazione degli attacchi DoS e DDoS, riteniamo che comunque l'insieme derivante, ne rappresenti un buon campione. Nella consultazione di questo campione, si sono notate varie problematiche, alcune correlate a singoli lavori, ed alcune comuni[35]. Queste ultime rivestono, come andremo ad esporre, un ruolo importante per l'effettiva applicabilità sul campo dei modelli[36]. La stesura di un insieme di regole, con lo scopo finale di normalizzare e fornire indicazioni certe nei riguardi del processo di validazione e training, per futuri lavori di ricerca, è di fondamentale importanza. Tutte queste problematiche comuni possono essere raggruppate in tre famiglie: una rilegata alla questione dei dataset, una alle metodologie utilizzate per la valutazione dei modelli, ed una all'utilizzo in ambienti reali dei modelli.

Dataset

Varie osservazioni sono già state fatte nella sezione 3.1.1, e molte altre in [129], evitando dunque una ripetizione, cerchiamo di riassumere i concetti fondamentali in questo sotto-paragrafo.

- **Vicinanza temporale:** Caratteristica fondamentale citata più e più volte in questo testo, è la data natia del traffico presente nel dataset usato per la propria ricerca. In un contesto altamente dinamico come la sicurezza informatica, dove vi è una continua corsa agli armamenti, è facile incorrere nel problema dell'obsolescenza. Vi sono vari autori che nonostante la disponibilità di dataset più recenti, scelgono comunque di costruire il proprio modello tramite dataset poco recenti (Figura 5.28). Ragione predominante alla base di questa scelta è collegata alla popolarità, e affermazione che possiede un determinato dataset rispetto ad un altro[37], ovvero, più il dataset è usato nella letteratura e datato, e più alto sarà il numero di ricerche e metriche disponibili agli autori per il confronto con il proprio modello.
- **Pluralità:** Numerose sono anche le ricerche che fanno uso di un singolo dataset per il training e per la validazione del modello, in linea generale, questa metodologia è sconsigliata. Argomentiamo, anche se possono essere prese contromisure da parte dei ricercatori per evitare problemi di overfitting sul determinato dataset,[38] è buona pratica eseguire una verifica aggiunta sulla capacità di generalizzazione del modello, tramite l'utilizzo di dataset prodotti/distribuiti da diversi autori. Per cercare di raggiungere una maggiore capacità di intercettamento dal proprio modello, è incentivato dunque, l'utilizzo plurimo di datasets.
- **Disponibilità:** Per ragioni di valutazioni e verifiche sulle performance dei modelli, da parti di attori terzi, l'utilizzo di dataset privati, o di difficile reperibilità, è altamente sconsigliato. Nel caso i ricercatori ritengano appropriato e necessario l'utilizzo di un dataset personale[39], o comunque modificato, per il proprio lavoro, la pubblicazione dello stesso, corredata con

[35] o comunque molto diffuse.
[36] secondo il nostro parere e di altri ricercatori.
[37] ad esempio, un dataset popolare ma datato è *KDD CUP 1999*.
[38] ad esempio con la tecnica della cross validation (convalida incrociata).
[39] in inglese detto *custom*

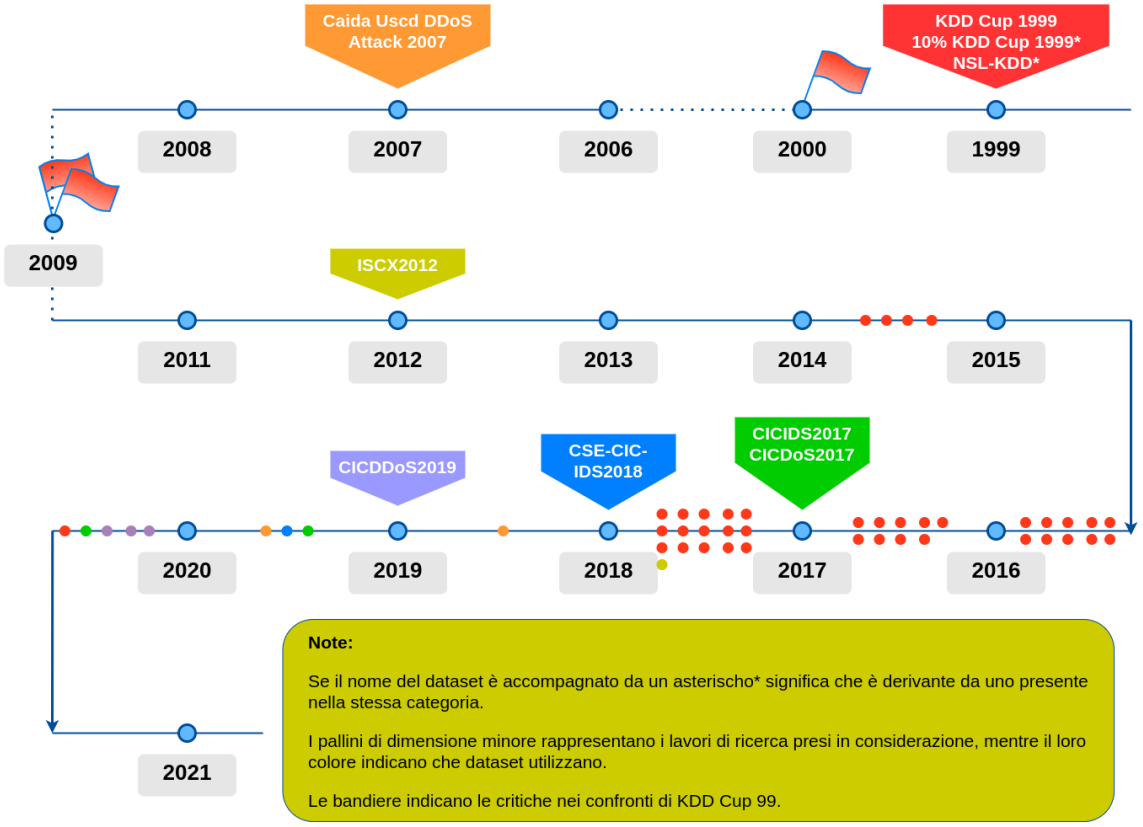


Figura 5.28: Locazione temporale dei lavori analizzati in [88] e in Tabella 5.5, corredate con le critiche [96][101][98].

un'adeguata documentazione, è da ritenersi (per il principio di trasparenza) d'obbligo. Anche in questo caso vi sono vari lavori che ve ne fanno utilizzo omettendo una descrizione soddisfacente.

Nota: Dataset "privati" e realistici
In base alla metodologia seguita nella compilazione dei dataset, si possono identificare tre distinte tipologie: dataset reali, sintetici e simulati. Datset di origine reale, ovvero derivanti dalle registrazioni di attacchi DoS/DDoS subiti, rappresentano un grande strumento per la ricerca; se si dovesse fare una classifica fra le tre categorie, ovviamente l'origine reale si piazzerebbe in testa. Come il lettore avrà potuto notare tuttavia, la maggioranza dei dataset da noi citati, appartengono alle ultime due categorie. Sfortunatamente, un grande svantaggio di questi dataset, prima di essere resi pubblicamente disponibili, vengono sottoposti a varie procedure di anonimizzazione, per garantire la privacy degli utenti della determinata rete/sottorete vittima dell'attacco. In un grande numero di casi, queste procedure, hanno come effetto collaterale, la diminuzione dell'utilità per l'applicazione di algoritmi ML.
Diverso è il discorso dell'utilizzo di datasets "privati", rispetto a quelli disponibili pubblicamente, questi possono essere generati/costruiti/acquisiti in modo da soddisfare appieno la domanda di ricerca. I dataset pubblici infatti, vengono considerati da alcuni ricercatori [46] come troppo generici e non abbastanza flessibili per poter soddisfare le necessità dei moderni IDS[a]. Inutile dire, test e validazioni da parte di altri ricercatori, in questo caso saranno obbligatoriamente assenti. La magnitudo di questa problematica è abbastanza soggettiva, in base alla corrente di pensiero, può essere considerata sia come uno svantaggio che come vantaggio.

[a] Ovviamente se il dataset contiene esclusivamente attacchi DoS/DDoS, il discorso cade.

- **Ambiente di generazione/cattura:** Oltre alla definizione e descrizione degli attacchi presenti nei datasets utilizzati, è importante dedicare una parte della ricerca, all'esposizione della topologia e tipologia dell'ambiente sorgente degli ultimi. Con tipologia e topologia si intende, a titolo esemplificativo, informazioni come il numero delle macchine presenti, la dimensione della rete, le tipologie di apparecchiature (switch, routers, firewalls, controllori, ecc.), i software utilizzati (nel caso di datasets sintetici/simulati) per la generazione, e mappe di rete. Queste informazioni diventano utili nella fase preliminare, in base alla domanda di ricerca, si può indirizzare la scelta dei dataset adeguati. Ad esempio, se si sviluppa un IDS commerciale, dataset che si incentrano su reti aziendali di media/grande dimensione sono da prediligere rispetto a dataset incentrati su reti domestiche. Risulta comunque chiaro che offrire questo genere di informazioni è onere dei produttori dei dataset in utilizzo, ai ricercatori basta indicare ai lettori la modalità per reperire tale documentazione.

Valutazioni

La discriminazione delle performance fra i modelli è eseguita nel processo di valutazione. Lavoro di elevata importanza, che ha come risultato l'assegnazione numerica rappresentativa della bontà generale del modello proposto, dovrebbe sempre seguire due principi base[40]:

- **Trasparenza:** I singoli valori numerici che indicano le performance del modello presi da soli, come in ogni aspetto ingegneristico, non hanno alcun peso ai sensi della valutazione finale. Ovvero, le metodologie, le metriche, i criteri, utilizzati dagli autori, devono essere sempre accompagnati da un ampia, dettagliata e sopratutto chiara, spiegazione. Il fine è fornire gli strumenti necessari a terzi di riprodurre i risultati ottenuti.
  Ad esempio, se vengono utilizzati subsets personalizzati, la semplice dichiarazione del loro impiego non basta:
  "[...] il dataset è stato diviso in x subset [...]" [41]
  "[...] il dataset è stato diviso in x subset, così definiti nella tabella y [...]" [42]

[40] riconducibili a quello padre di riproducibilità
[41] da considerarsi non soddisfacente
[42] buona pratica

Caratteristica	Descrizione
Configurazione della rete	Descrizione sulla topologia della rete, comprendente di quanti e quali tipi di dispositvi ne fanno parte. Con la spiegazione di come sono interfacciati con l'ambiente di test
Traffico di rete	I pacchetti di rete devono essere catturati dalla sorgente fino alla destinazione, comprendendo tutti i dispositivi.
Classificazione	Le istanze del dataset, devono essere opportunamente classificate per comprendere appieno le interazioni all'interno/esterno di esso.
Interazioni	Tutte le interazioni che avvengono all'interno ed esterno della rete devono essere catturate.
Cattura	Le cattture devono poter permettere la misurazione delle metriche per gli IDS (come ad es. DR, FPR, etc.)
Protocolli	Idealmente, tutte le iterazioni dovrebbero usare diversi protocolli.
Attacchi	Numerosi e vari attacchi devono essere presenti, che devono comprendere ovviamente anche gli attacchi più recenti
Anominità	Il paylod e l'header non devono essere omesse per questioni di anominità
Eterogeneità	Per coprire tutti i dettagli sulle procedure usate per rilevare gli attacchi, la cattura deve essere effettuataa da molteplici attori.
Caratteristiche	Le caratteristiche che identificano le istanze del dataset devono essere complete e ben definite.
Metadata	Documnetazione appropriata sulla descrizione dell'ambiente di test, dell'infrastruttura e scenari d'attacco, infrastuttura della vititma, deve essere presente.

Tabella 5.7: Caratteristiche necessarie per costruire un dataset ideale secondo [11]. La stessa tabella, ma in inglese, è proposta inoltre da [12]

- **Equità:** Per il confronto del proprio modello con quello di altri autori/ricercatori, dovrebbero essere adottate le stesse metodologie e metriche. Dato che nella maggior parte dei casi, l'autore non ha disponibilità del modello esatto della controparte[43], questo principio si riduce all'utilizzo degli stessi dataset per la valutazione, e delle medesime metriche. In linea generale, è compito dell'autore cercare di raggiungere il maggior grado di similitudine sui parametri di test.

La capacità di individuazione del modello non è da valutare singolarmente, la tempistica con la quale viene effettuata tale azione è da tenere in considerazione. Facendo un esperimento mentale, un ipotetico ed ideale modello capace di individuare correttamente il 100% degli attacchi, risulterebbe degradato se per raggiungere tali prestazioni, impiegasse un tempo eccessivo; l'attaccante ad esempio, potrebbe essere riuscito ugualmente a rendere irraggiungibili i servizi target, oppure, la velocità di connessione ed il tempo di risposta fra i servizi e gli utenti, a seguito del processo di inoltro traffico, potrebbe risultare calata drasticamente, impattando ugualmente la qualità di servizio. Conseguentemente nelle valutazioni, anche questo aspetto deve essere tenuto conto, riportare misure relative sia al tempo necessario per l'addestramento del modello, sia al tempo per l'individuazione dovrebbe essere buona pratica da parte del ricercatore.

Test in ambienti reali

Ultimo punto, ma non per ordine di importanza, è l'effettiva applicabilità del metodo/modello testato in un contesto reale. Tecnicamente riconducibile ad una metrica di valutazione[44], la consideriamo essere un problema distinto, poiché nella letteratura viene effettuata la medesima distinzione. Che sia un lavoro di ricerca (research) o compilativo (review), il punto finale di convergenza deve essere (o comunque preso in considerazione) l'effettiva applicabilità sul campo reale dei modelli analizzati/proposti. La problematica nasce proprio su questo frangente: vi sono diversi

[43] a meno che non lo rendano disponibile pubblicamente nella sua interezza

[44] la bontà del modello è determinata anche dall'effettiva applicabilità sul campo dello stesso, di conseguenza dovrebbe rientrare come metrica di valutazione.

casi in cui questa valutazione viene mal posta o argomentata in modo insoddisfacente [138][27][133]. Eseguire attacchi massicci Dos/DDoS realistici potrebbe risultare insostenibile per dei ricercatori; costituire ad esempio delle botnets e lanciare un attacco sulla propria rete, non è certo un'attività semplice/conseguibile[45]. Non riteniamo dunque corretto escludere dalla categoria "test in ambienti reali", tutte le simulazioni di attacchi tramite software in reti/sottoreti sia reali che simulate.[46]. Sempre in quest'ottica, osservazioni su eventuali degradazioni della qualità del servizio per gli utenti finali, devono essere effettuate.

[45] ricordiamo che le botnet possono raggiungere un numero elevato di macchine infette.

[46] generalmente gli attacchi vengono creati tramite softwares reperibili online, e.g. *Low Orbit Ion Cannon (LOIC), High Orbit Ion Cannon (HOIC), Slowloris, slow header, slow header slow post, ddosim, etc.*

Capitolo 6

Possibili soluzioni

A questo punto del discorso, il lettore avrà capito che il problema dell'individuazione degli attacchi DoS/DDoS sul campo reale, sia complesso e pieno di sfide. Nel processo d'esposizione abbiamo osservato molti lavori e sollevato diverse critiche presenti nella letteratura.
Coscienti di ciò, e senza alcuna pretesa (sapendo che *"Facile è criticare, difficile innovare"*), proponiamo alcune soluzioni, da noi ritenute valide direzioni per le future ricerche.

6.1 Testbed

Al momento della stesura di questo documento, sembra essere necessaria l'introduzione ed adozione di un nuovo dataset di riferimento che vada a sostituire il predecessore "KDD CUP 1999". Una discussione più ampia e seria, nei confronti di questo argomento deve essere sostenuta nell'ambiente di ricerca, che punti all'individuazione del candidato maggiormente promettente per i futuri lavori. Concentrandoci, tuttavia, esclusivamente sul pericolo rappresentato dagli attacchi DoS/DDoS, questo ruolo può essere, a nostra opinione, ricoperto dal "CIC-DDoS2019 Dataset". Essendo di recente data (2019), distribuito e curato dal CIC, ovvero da una fonte autorevole del campo, si pone come opzione appetitosa per i prossimi anni.[1] Ampliando invece il discorso per comprendere l'intera area degli IDS, questo ruolo può essere assunto da "CIC-IDS-2017" e "CSE-CIC-IDS-2018" (entrambe soddisfano i criteri riportati nella Tabella 5.7), in questo caso tuttavia, essendo più ampio il discorso, altre considerazioni che non sono state prese in considerazione in questo testo, devono essere fatte, le quali potrebbero portare a differenti conclusioni.
Poniamo nuovamente l'attenzione del lettore sulla dinamicità della sicurezza informatica: basare il proprio sistema difensivo su uno o più dataset di riferimento statici (una volta distribuiti non vengono aggiornati), non sembra essere una buona pratica. Una scelta, a nostro parere, più consona, è il continuo aggiornamento del datset (datsets) di riferimento, ogni qual volta una nuova tipologia/-metodologia d'attacco è scoperta. La soluzione proposta, presenta però nuovamente il problema correlato alla privacy: la scoperta di un nuovo attacco è solitamente collegata al successo dello stesso sul campo. Questo comporterebbe individuare manualmente, o tramite uno strumento su misura (ad hoc), i singoli pacchetti maligni che l'attore target ha ricevuto, scartando al contempo (per motivi di privacy) quelli appartenenti ad utenti benigni. Altra soluzione, una volta esaminato e capito il meccanismo su come agisce l'attacco novello, intraprendere il processo di simulazione e cattura per l'aggiunta al dataset di riferimento.
Questa proposta si basa però su diverse ipotesi:

- **Collaborazione:** Sia l'ambiente di ricerca, che gli attori vittime degli eventuali attacchi, devono essere disposti ad aggiornare un unico[2] dataset. I primi devono collaborare nell'individuazione del dataset di riferimento che sarà utilizzato, mentre i secondi, devono essere disposti a condividere l'attacco subito, dopo averlo filtrato dalle informazioni soggette a privacy.

[1] Rimane comunque il consiglio dell'utilizzo plurimo di dataset, sia per la fase di training, ché di testing, per il proprio modello (come discusso nelle sezioni precedenti).
[2] O plurimi.

- **Moderatori:** L'aggiornamento del dataset, deve essere affidato ad un gruppo di moderazione composto da soggetti del campo e privi di conflitti di interessi. L'istituto che ha creato il dataset potrebbe risultare, se collaborativo, come un buon candidato, un'altra opzione è la composizione di soggetti di diversi team di ricerca per assicurare l'assenza di conflitti d'interesse. Ancora meglio, sarebbe la partecipazione e collaborazione di istituti nazionali (o europei) per la sicurezza cibernetica come il CSIRT (Il quale già pubblica allerts pertinenti).[3]

- **Incentivazioni:** La comunicazione fra le "Vittime di attacchi novelli" e i "Moderatori" deve essere opportunamente incentivata. Oneroso è infatti il lavoro che i primi attori, devono eseguire per poter consegnare al team di moderazione il traffico maligno. Oltretutto non è raro che l'attore colpito voglia nascondere, per motivi di reputazione, l'attacco subito[4]. Conseguentemente, che sia di tipo monetario e/o di pubblicizzazione, questo tipo di incentivazione deve essere messa in atto.

La struttura che si andrebbe a definire sarebbe simile a quella schematizza nella Figura 6.1.

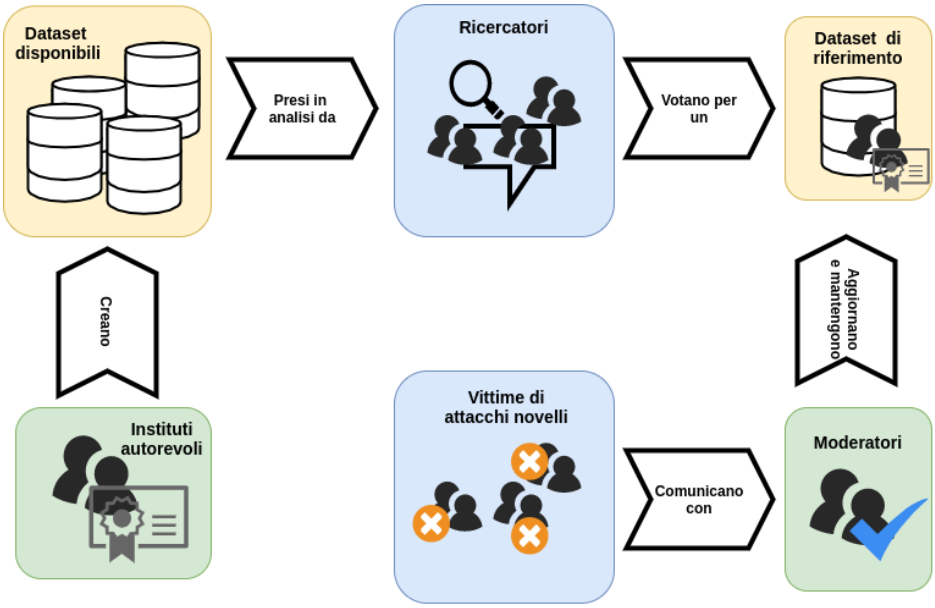


Figura 6.1: Struttura proposta per il processo dell'individuazione ed utilizzo di un nuovo dataset di testbed (riferimento).

6.2 Metodologie

In base alla mole del traffico di rete da analizzare, è consigliabile adottare modelli che sfruttano tecniche di campionamento. Un esempio di modello che segue questa direzione, è costituito dal lavoro svolto in [139]. Questa proposta è da valutare in caso la rete da proteggere assume dimensioni rilevanti, ovviamente, dove il carico di lavoro, non impatta negativamente sulle prestazioni dell'analizzatore, è possibile omettere il campionatore e relativa complessità aggiunta. Inoltre, questa metodologia, sembra essere facilmente implementabile con le SDNs.
L'altra valutazione degna di nota, sono appunto le SDNs (Software Defined Networks), ovvero, il relativo nuovo paradigma sull'architettura delle reti. Per evitare di dilungarci troppo a lungo su questo tema, basti sapere al lettore, che queste reti permettono il disaccoppiamento del piano di controllo, dal piano dei dati, tramite l'introduzione di un controllore centralizzato (Figura 6.2),

[3] Ulteriori informazioni oltre al sito in sé del CSIRT, sono offerte dalla pagina presente sul sito della Camera dei deputati.

[4] Si pensi ad un Istituto bancario, o di un azienda di alto rilievo, una notizia che può impattare negativamente sulla reputazione della stessa, comporterebbe una perdita finanziaria.

Nome della sfida	Descrizione
Affidabilità	La natura centralizzata del controllore, può rappresentare un single point of failure (SPOF) .
Scalabilità	La comunicazione fra molteplici controllori (se si opta ad una infrastruttura distribuita di tipo peer-to-peer) risulta essere difficoltosa a causa di carenti APIs di tipo westbound ed eastbound.
Sicurezza	Il controllore ha una visione e controllo, completa della rete, conseguentemente, basta la compromissione di quest'ultimo per avere il completo controllo dell'intera rete (SPOF).

Tabella 6.1: Alcune delle nuove sfide introdotte con l'applicazione delle SDNs.

consentendo al livello di controllo di configurare e gestire dinamicamente le infrastrutture presenti nella rete.
Dati i numerosi vantaggi rispetto alle reti tradizionali, la direzione qui consigliata per la ricerca, è lo studio e sviluppo in parallelo, di applicazioni[5] per la sicurezza delle reti SDNs. Ricerca che deve analizzare le nuove vulnerabilità introdotte nel sistema[48] a causa della struttura e funzionalità di questo paradigma. Ad esempio, a causa della loro natura centralizzata del piano di controllo, questi tipi di rete, possono introdurre criticità sul fronte degli attacchi DoS/DDoS[47] (Tabella 6.1).

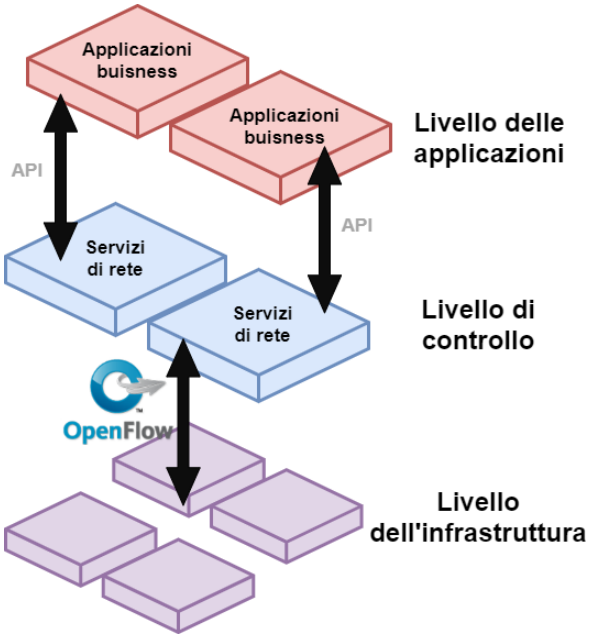


Figura 6.2: Architettura di una SDN, immagine basata da quella proposta da opennetworking.org (ONF).

Non è da nascondere che varie ricerche sono già state effettuate anche su questa nuova architettura, tuttavia, come sottolineato anche da [49], necessita di ulteriori approfondimenti, per valutare il nuovo potenziale[6].

[5] Nelle SDNs vengono chiamate "Applicazioni business" 6.2.

[6] In questo documento ci siamo dunque concentrati sulle reti tradizionali per varie ragioni, innanzitutto, essendo di tipo "tradizionale", sono quelle studiate e ricercate da un più vasto numero di ricercatori . In seconda battuta, sia per questioni di popolarità che di legacy, risultano tutt'oggi le reti più utilizzate.

Capitolo 7

Snippets

Perseguendo lo spirito di trasparenza, presentiamo i vari codici Python scritti ed utilizzati all'interno di questo testo. Occasionalmente, frammenti di codice sorgente atti ad esempi ed/o esposizioni, vengono chiamati in gergo con il nome di "Snippets" (in italiano letteralmente significa frammenti).

7.1 Calcolo dell'effettivo volume del dataset: CSE-CIC-IDS2018

In questa sezione, dopo aver fatto una piccola introduzione su **CSE-CIC-IDS2018**, esponiamo in che modo è stato verificato il numero di attacchi DoS, DDoS per il dataset.

7.1.1 Obbiettivi

Due fatti da prendere in considerazione sono il motivo e gli obbiettivi che i ricercatori si sono posti per la costruzione del dataset. Nella sezione 3.1.6 parliamo del lavoro svolto da [50], il problema in questo caso risiede nel fatto che l'obiettivo del ricercatore era incentrato solo sugli attacchi DoS e DDoS, mentre quello perseguito dai ricercatori CIC e CSE era incentrato sugli IDS in generale. Spieghiamoci meglio, gli IDS devono ricoprire un grande ombrello di tipi di attacco, per citarne alcuni: Bruteforce attack, Web attack, Infiltration attack,Botnet attack. Gli attacchi DoS e DDoS ne sono solo una parte, un sottoinsieme, questo, più al fatto che vi sono anche qui numerosi tipi di attacchi DoS e DDoS, hanno portato ai ricercatori CIC e CSE di selezionarne solo alcuni. Questo perché, come detto prima, non era il loro obiettivo primario.

7.1.2 Struttura

Dato che siamo interessati solo agli attacchi DoS e DDoS, scarteremo dalla descrizione della struttura tutti gli altri tipi di attacco, se il lettore è interessato, può sempre consultare il sito CIC per avere ulteriori informazioni. Tutti gli attacchi hanno luogo in quattro giorni, in particolare, il 15,16,20 e il 21 del febbraio del 2018. Nella Tabella 7.1 troviamo tutte le informazioni necessarie offerte dal CIC. Per quanto riguarda a i file csv, sono suddivisi per giorno, e tutti quanti possiedono

Tipo di attacco	Giorno di riferimento
DoS-GoldenEye	Thurs-15-02-2018
DoS-Slowloris	Thurs-15-02-2018
DoS-SlowHTTPTest	Fri-16-02-2018
DoS-Hulk	Fri-16-02-2018
DDoS attacks-LOIC-HTTP	Tues-20-02-2018
DDoS-LOIC-UDP	Tues-20-02-2018
DDoS-LOIC-UDP	Wed-21-02-2018
DDoS-HOIC	Wed-21-02-2018

Tabella 7.1: Struttura di CSE-CIC-IDS2018, con incluso solo gli attacchi DoS e DDoS

la colonna *Label*, dove viene specificato di che natura è il pacchetto. In caso di traffico normale, il pacchetto avrà nel campo *Label* la dicitura *Benign*, in tutti gli altri casi[1] la dicitura sarà formata dal suffisso *DoS attacks-* più il nome dell'attacco in questione; ad esempio *DoS attacks-SlowHTTPTest*.

7.1.3 Metodologia

Siamo ora a conoscenza dei giorni in cui sono stati eseguiti solo attacchi DoS e DDoS (la presenza di traffico normale vi è in tutti i giorni), e la struttura dei file csv, possiamo scrivere un semplice script per il conteggio. Lo script 7.2 è stato scritto in Python, è molto semplice, ma comunque spieghiamo brevemente il suo funzionamento. Apre il file csv di riferimento, legge solo la colonna *['Label']*, se riscontra la stringa di testo *['Benign']* aumenta un contatore chiamato di scarto, altrimenti in tutti gli altri casi[2], aumenta un contatore diverso, chiamato semplicemente conteggio. Alla fine della lettura del file, scrive i risultati in un file a parte.

Listing 7.1: Codice usato per la verifica del numero di attacchi e traffico normale

```
import os
import pandas

cartella_riferimento = """dove sono presenti i file csv"""

nome_file = (os.listdir(cartella_riferimento)["""il numero del file csv"""])
nome = os.path.join(cartella_riferimento, nome_file)
colonna = ['Label']
"""Apriamo il file csv indicato e leggiamo solo la colonna ['Label']"""
file = pandas.read_csv(nome, usecols=colonna, dtype={'Label':str})
lista = file.values.tolist()

documento = open("Riassunto.txt", "w")
conteggio = 0
conteggio_scarto = 0
for i in range(len(lista)):
        """Se nella colonna ['Label'] trovi la stringa ['Benign'] incrementa
           il conteggio dello scarto di uno"""
    if str(lista[i]) == "['Benign']":
        conteggio_scarto = conteggio_scarto + 1
    else:
        """Altrimenti, incrementa il conteggio"""
        documento.write(str(lista[i]))
        conteggio = conteggio + 1
"""Una volta letto tutto il file, scrivi i conteggi e un messaggio di completamento"""
doc2 = open("Conteggio.txt", "w")
doc2.write("Attacchi totali trovati in " + str(nome_file) + " = " + str(conteggio)
           + "\n" + "Totale pacchetti " + str(conteggio_scarto + conteggio))
print("COMPLETATO")
```

7.1.4 Risultati

Proponiamo qui i risultati del codice 7.2 mettendoli a confronto con i numeri dati da [50]. Dall'analisi della tabella, possiamo dedurre (anche se non possiamo esserne sicuri), che l'autore di [50] abbia preso solo in considerazione il giorno 21 (Le celle evidenziate).

7.1.5 Reperibilità

Per avere tutto il dataset è possibile seguire la procedura delineata dal sito CIC nel paragrafo *To download this dataset*. Dato che siamo interessati a controllare solo il numero di attacchi DoS, DDoS, e i pacchetti benigni, la nostra necessità si riduce ai file *.csv*. Grazie inoltre alla dimensione ridotta di questi, rispetto agli altri presenti nel dataset, possiamo reperirli più facilmente dal sito kaggle.com.

[1] nel caso di DDoS il suffisso sarà *DDoS attacks-*

[2] Non serve fare anche qua un matching, abbiamo già verificato che nei giorni che stiamo andando a verificare con lo script, sono stati eseguiti solo attacchi DoS e DDoS

Nome File	Num. Attacchi	Num. Traffico Benigno	Totale
02-15-2018.csv	52,498	996,077	1,048,575
02-16-2018.csv	601,803	446,772	1,048,575
02-20-2018.csv	576,191	7,372,557	7,948,748
02-21-2018.csv	687,742	360,833	1,048,575
TOTALE	1,918,234	9,176,239	11,094,473
	Numeri secondo [50]		
	686,012	360,833	1,046,845

Tabella 7.2: Risultati del codice 7.2 e confronto con [50]

7.2 Equazione 5.45

Per la costruzione dei grafici rappresentati in figura 5.26, si è adoperato il seguente codice:

Listing 7.2: Codice usato per il calcolo dell'equazione 5.45

```
import math

import matplotlib.pyplot
from matplotlib import pyplot as plt


'''Adibita per il controllo dei risultati della formula'''
def look_up(n):
    lista = [1, 1, 3, 25, 543, 29281, 3781503, 1138779265, 783702329343,
             1213442454842881, 417509897643059814 3, 316034593964189176074 25,
             521939651343829405020504063, 186766007444320351866648169267 21,
             143942814104439833494179071983953510 3]
    if n < len(lista):
        return lista[n]


'''Formula per il calcolo del numero di reti'''
def calcola_reti(num_reti):
    sommatoria = []
    risultato_finale = 0
    if num_reti == 1 or num_reti == 0:
        return 1
    else:
        i = 1
        while i <= num_reti:
            parte_A = math.pow(-1,(i+1))
            parte_B = coeff(num_reti,i)
            parte_C = math.pow(2, (i * (num_reti - i))) * calcola_reti((num_reti - i))
            sommatoria.append(parte_A * parte_B * parte_C)
            risultato_finale = risultato_finale + (parte_A * parte_B * parte_C)
            i = i+1
        #print(sommatoria)
        return risultato_finale


'''Per il grafico'''
x = []
y = []

for i in range(12):
    i = i + 1
    x.append(i)
    y.append(calcola_reti(i))

plt.bar(x,y, width=0.1)
matplotlib.pyplot.yscale("log")
plt.ylabel("Numero di reti")
plt.xlabel("Numero di nodi")
```

```
plt.show()
```

Il calcolo del coefficiente binario ("parte_B") è gestito dalla funzione "coeff(n,k)", mentre l'equazione 5.46 è stata divisa per comodità (come in visualizzato in Figura 7.1), in tre parti: "parte_A", "parte_B" e "parte_C".

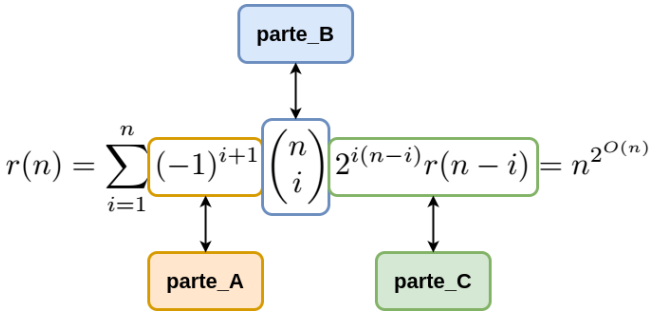


Figura 7.1: Divisione della formula 5.45.

Capitolo 8

Conclusioni

La mitigazione degli attacchi DoS e DDoS tramite tecnologie basate sul ML e DL è un campo corredato da un grande numero di ricerche e possibili soluzioni. Un campione rappresentativo di queste è stato scelto, circoscrivendo l'attività di analisi ad un numero più gestibile, permettendoci di intraprendere una discussione ampia ma completa.
Alla fine di questa, possiamo trarre alcune affermazioni: la scelta degli algoritmi atti per la classificazione tenderà, con il passare del tempo e con l'avanzamento tecnologico, a soluzioni basate sul DL (anziché solo sul ML); tendenza già individuabile oggigiorno, come suggerito dalla Figura 5.9. La mancanza di decisione, nel campo accademico, sul tipo di algoritmo di classificazione[1] più indicato per questo tipo d'attività, porta a dedurre, che il fattore determinante alla scelta di quest'ultimo, piuttosto che essere correlato principalmente alla natura su cui esso si basa, sia invece, la dimestichezza e conoscenza del ricercatore ad costituire il fattore determinante. Più chiaramente, le diversità in performance riscontrate dai vari lavori, sono (in un grande numero di casi) sormontabili, tramite una migliore scelta di parametri, conoscenza, preparazione dati, ovvero, tramite l'adozione di algoritmi di cui si conosce a fondo il significato dei parametri e metodologie di funzionamento.
Detto ciò, fra i numerosi algoritmi di ML e DL presenti nel campo dell'apprendimento automatico, ci è stato comunque possibile delineare, una lista formata da quelli più efficienti, ed utilizzati, dalla letteratura corrente, circoscrivendo la scelta a questi: Naïve Bayes, Decison Trees, Random Forests, Support Vector Machines, Stacked Autoencoders, Recurrent Neural Networks, Long Short-Term Memory (alcune volte raggruppata insieme alle RNN), Bayesian Networks, ed infine K-Nearest Neighbourhood (sezione 5.4).
Infine, dopo aver sottolineato i problemi di obsolescenza correlati al dataset di rifermento (KDDCup 99), è stato proposto un nuovo sostituto (CIC-DDoS2019 Dataset), in parallelo a possibili soluzioni nei confronti dell'ambiente di testbed (sezione 6.1) e direzioni future su implementazioni baste sulle SDNs.
Con l'auspicio di aver soddisfatto le aspettative del lettore, concludiamo il documento.

[1] Fra quelli discussi nella sezione 5.4.

Capitolo 9

Glossario

- Acc: Accuracy, accuratezza in italiano, per maggiori informazioni consultare il capitolo 5.2.3.
- ACCS: Australian Centre for Cyber Security.
- AFRL/SNHS: Air Force Research Laboratory.
- ANN: Artificial Neural Network.
- API: Application Programming Interface.
- BAF: Bandwidth Amplification Factor.
- Bagging: Bootstrap Aggregating.
- BayesNet: Bayesian Network.
- Benchmark: Test, metodologia, strumento, che rappresenta il punto di riferimento per un determinato campo. Utilizzato per valutare prestazioni, bontà ed altre caratteristiche di nuove tecnologie/metodologie.
- BNN: Binarized Neural Network.
- CART: Classification And Regression Tree.
- CIC: Canadian Institute for Cybersecurity.
- CISA: Cybersecurity & Infrastructure Security Agent.
- closed-source: Software a pagamento, la cui licenza ne vieta la ridistribuzione non autorizzata, la modifica, lo studio, ecc. Viene chiamato anche software proprietario.
- CNN: Convolutional Neural Networks.
- CSE: Communications Security Establishment.
- CSIRT: Computer Security Incident Response Team - Italia
- csv: comma separated values, essenzialmente una tabella excel.
- CharGen: Character Generator Protocol [75].
- DARPA: Defence Advanced Research Project Agency.
- DDoS: Distributed Denial of Service.
- DoS: Denial of Service.
- DNS: Domain Name System.

- DT: Decision Tree.
- DR: Detection Rate, per maggiori informazioni consultare il capitolo 5.2.3.
- ecc.: et cetera (formula latina), in italiano 'e le cose che rimangono'.
- ECHO: Servizio di debugging che semplicemente reinvia al mittente qualunque dato mandato[77].
- ER: Error Rate, per maggiori informazioni consultare il capitolo 5.2.3.
- e.g. exempli gratia, locuzione latina che in italiano diventa "per esempio".
- et al. Questa abbreviazione è riscontrata solitamente nelle citazioni, abbrevia l'espressione latina et alii, che significa "e altri".
- FAR: False Allarm Rate, per maggiori informazioni consultare il capitolo 5.2.3.
- FNR: False Negative Rate, per maggiori informazioni consultare il capitolo 5.2.3.
- F1: Metrica utilizzata spesso nel campo del machine learning, per maggiori informazioni consultare il capitolo 5.2.3.
- GUI: Graphical User Interface, interfaccia grafica.
- ICMP: Internet Control Message Protocol, descritto nella RFC 792.
- IDPS: Intrusion Detection Prevention System.
- IDS: Intrusion Detection System.
- ISCX: Information Security Center of Excellence.
- ISP: Internet Service Provider.
- kNN: k-Nearest Neighbors.
- LDAP: Lightweight Directory Access Protocol.
- LR: Logistic Regression.
- LSTM: Long Short-Term Memory.
- LSVM: Lagrangian Support Vector Machine.
- LS-SVM: Least Squares Support Vector Machines.
- LSVM: Support Vector Machine with Linear kernel.
- Middlebox: Qualunque dispositivo (anche non fisico) con ruolo di intermediario, che svolge operazioni differenti da quelle standard di un router IP, sul percorso dei datagrammi, fra l'host sorgente e l'host destinatario. RFC 3234
- MIT: Massachusetts Institute of Technology
- MSSQL: Microsoft Structured Query Language.
- NetBIOS: Network Basic Input/Output System.
- NN: Neural Network.
- NTP: Network Time Protocol [78].
- open-source: Software gratuito, la cui licenza permette a chiunque la modifica del codice sorgente. Fra le licenze più popolari vi sono: GNU (General Public License), la licenza BSD e la licenza Apache.
- PAF: Packet amplification factor.

- Pattern: Disposizione, modello, schema, può anche assumere caratteristiche ripetitive, ad esempio una facciata composta di mattoni.
- pcap: Tipo di estensione di file, indica l'abbreviazione di ***pac**ket **cap**ture*.
- Prec: Precision, precisione, per maggiori informazioni consultare il capitolo 5.2.3.
- Query: Interrogazione di un database, ovvero una domanda, una richiesta che viene effettuata.
- RF: Random Forest.
- RFECV: Recursive Feature Elimination with Cross-Validation.
- RNN: Recurrent Neural Network.
- RPC: Remote Procedure Call.
- SAE: Stack AutoEncoder
- SAE - 1 SVM: Stack AutoEncoder con One-Class Support Vector Machine.
- SGD: Stochastic Gradient Descent.
- SimpleMKL: Simple Keyhole Markup Language.
- SMO: Sequential Minimal Optimization.
- SNMP: Simple Network Management Protocol.
- SPOF: Single Point Of Failure.
- SKM-HFS: Semi-supervised K-Means Detection algorithm using Hybrid Feature Selection method.
- SYN: Flag del protocollo TCP, Sezione 3.1 di RFC 793.
- SSDP: Simple Service Discovery Protocol.
- SVM: Support Vector Machines.
- TCP: Transmission Control Protocol, RFC 793.
- TNR: True Negative Rate, per maggiori informazioni consultare il capitolo 5.2.3.
- TFTP: Trivial File Transfer Protocol [80]..
- UDP: User Datagram Protocol, RFC 768.
- UNB: University of New Brunswick.
- Weka: Waikato Environment for Knowledge Analysis.

Capitolo 10

Bibliografia

[1] Cost of Web Application and Denial of Service Attacks: EMEA Ponemon Institute, September 2018, pagina 14 akamai.com/report

[2] A survey on anomaly and signature based intrusion detection system (IDS), Mrs. Anshu Gangwar et al Int. Journal of Engineering Research and Applications ISSN : 2248-9622, Vol. 4, Issue 4(Version 1), April 2014, pp.67-72

[3] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", IEEE 53rd International Carnahan Conference on Security Technology, Chennai, India, 2019

[4] Andreea Bendovschi, Cyber-Attacks – Trends, Patterns and Security Countermeasures, Procedia Economics and Finance, Volume 28, 2015, Pages 24-31, ISSN 2212-5671, https://doi.org/10.1016/S2212-5671(15)01077-1. (http://www.sciencedirect.com/science/article/pii/S2212567115010771

[5] Zuech, R., Khoshgoftaar, T.M. & Wald, R. Intrusion detection and Big Heterogeneous Data: a Survey. Journal of Big Data 2, 3 (2015). Disponibile qui.

[6] Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2

[7] Dong, Bo, and Xue Wang. "Comparison deep learning method to traditional methods using for network intrusion detection." 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN). IEEE, 2016.

[8] Sahar, Nan Md, Suhaila Sari, and N. S. A. M. Taujuddin. "Intrusion-Detection System Based on Hybrid Models." IOP Conference Series: Materials Science and Engineering. Vol. 917. No. 1. IOP Publishing, 2020.

[9] Kok, S., et al. "A comparison of various machine learning algorithms in a distributed denial of service intrusion." Int. J. Eng. Res. Technol 12.1 (2019): 1-7.

[10] Amor, Nahla Ben, Salem Benferhat, and Zied Elouedi. "Naive bayes vs decision trees in intrusion detection systems." Proceedings of the 2004 ACM symposium on Applied computing. 2004.

[11] Gharib A, Sharafaldin I, Lashkari AH, Ghorbani AA. An evaluation framework for intrusion detection dataset. In: 2016 International Conference on Information Science and Security (ICISS). IEEE; 2016. p. 1–6.

[12] Thakkar, Ankit, and Ritika Lohiya. "A review of the advancement in intrusion detection datasets." Procedia Computer Science 167 (2020): 636-645.

[13] Shurman, Mohammad, Rami Khrais, and Abdulrahman Yateem. "DoS and DDoS attack detection using deep learning and IDS." Int. Arab J. Inf. Technol. 17.4A (2020): 655-661.

[14] Göcs, László & Johanyák, Zsolt & Szilveszter, Kovács. (2016). Review of Anomaly-Based IDS Algorithms. TEAM 2016, Proceedings of the 8th International Scientific and Expert Conference.

[15] Liberios VOKOROKOS, Anton BALÁŽ, Martin CHOVANEC, INTRUSION DETECTION SYSTEM USING SELF ORGANIZING MAP , Acta Electrotechnica et Informatica No. 1, Vol. 6, 2006

[16] Patel, K. & Buddhadev, Bharat. (2013). An Architecture of Hybrid Intrusion Detection System. International Journal of Information & Network Security. 2. 197-202. 10.11591/ijins.v2i2.1753.

[17] Kamarudin, Muhammad Hilmi, et al. "A logitboost-based algorithm for detecting known and unknown web attacks." IEEE Access 5 (2017): 26190-26200.

[18] Shurman, Mohammad M., Rami M. Khrais, and Abdulrahman A. Yateem. "IoT Denial-of-Service Attack Detection and Prevention Using Hybrid IDS." 2019 International Arab Conference on Information Technology (ACIT). IEEE, 2019.

[19] Abbes, Tarek, Adel Bouhoula, and Michaël Rusinowitch. "Protocol analysis in intrusion detection using decision tree." International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004.. Vol. 1. IEEE, 2004.

[20] NIST,Karen Scarfone, Peter Mell, Guide to Intrusion Detectionand Prevention Systems(IDPS), Special Publication 800-94, disponibile su NIST.gov

[21] Yihunie, Fekadu, Eman Abdelfattah, and Ammar Odeh. "Analysis of ping of death DoS and DDoS attacks." 2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT). IEEE, 2018.

[22] Doshi, Rohan, Noah Apthorpe, and Nick Feamster. "Machine learning ddos detection for consumer internet of things devices." 2018 IEEE Security and Privacy Workshops (SPW). IEEE, 2018.

[23] Jaramillo, L. E. S. (2018). Malware Detection and Mitigation Techniques: Lessons Learned from Mirai DDOS Attack. Journal of Information Systems Engineering & Management, 3(3), 19. https://doi.org/10.20897/jisem/2655

[24] De Donno, Michele, et al. "Analysis of DDoS-capable IoT malwares." 2017 Federated Conference on Computer Science and Information Systems (FedCSIS). IEEE, 2017.

[25] Dong, Bo, and Xue Wang. "Comparison deep learning method to traditional methods using for network intrusion detection." 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN). IEEE, 2016.

[26] Samek, Wojciech, Thomas Wiegand, and Klaus-Robert Müller. "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models." arXiv preprint arXiv:1708.08296 (2017).

[27] Jia, Bin, et al. "A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning." Journal of Electrical and Computer Engineering 2017 (2017).

[28] Lima Filho, Francisco Sales de, et al. "Smart detection: an online approach for DoS/DDoS attack detection using machine learning." Security and Communication Networks 2019 (2019).

[29] Asad, Muhammad, et al. "DeepDetect: detection of distributed denial of service attacks using deep learning." The Computer Journal 63.7 (2020): 983-994.

[30] Williamson, Carey. "Internet traffic measurement." IEEE internet computing 5.6 (2001): 70-74.

[31] Liu, Hongyu, and Bo Lang. "Machine learning and deep learning methods for intrusion detection systems: A survey." applied sciences 9.20 (2019): 4396.

[32] Adadi, Amina, and Mohammed Berrada. "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)." IEEE access 6 (2018): 52138-52160.

[33] Niyaz, Quamar, Weiqing Sun, and Ahmad Y. Javaid. "A deep learning based DDoS detection system in software-defined networking (SDN)." arXiv preprint arXiv:1611.07400 (2016).

[34] Li, Chuanhuang, et al. "Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN." International Journal of Communication Systems 31.5 (2018): e3497.

[35] Priyadarshini, Rojalina, and Rabindra Kumar Barik. "A deep learning based intelligent framework to mitigate DDoS attack in fog environment." Journal of King Saud University-Computer and Information Sciences (2019).

[36] Zhang, Harry. "Exploring conditions for the optimality of naive Bayes." International Journal of Pattern Recognition and Artificial Intelligence 19.02 (2005): 183-198.

[37] Murphy, Kevin P. "Naive bayes classifiers." University of British Columbia 18.60 (2006).

[38] Jadhav, Sayali D., and H. P. Channe. "Comparative study of K-NN, naive Bayes and decision tree classification techniques." International Journal of Science and Research (IJSR) 5.1 (2016): 1842-1845.

[39] James, Gareth, et al. An introduction to statistical learning. Vol. 112. New York: springer, 2013.

[40] "Decision Trees" disponibile sul sito scikit-learn.org.

[41] Louppe, Gilles. "Understanding random forests: From theory to practice." arXiv preprint arXiv:1407.7502 (2014).

[42] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[43] Leshem, Guy. "Improvement of adaboost algorithm by using random forests as weak learner and using this algorithm as statistics machine learning for traffic flow prediction. Research proposal for a Ph. D." Research proposal for a Ph. D. thesis, the Hebrew university of Jerusalem (2005).

[44] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[45] Disponibile online al seguente sito: https://keras.io/about/

[46] Drewek-Ossowicka, Anna, Mariusz Pietrołaj, and Jacek Rumiński. "A survey of neural networks usage for intrusion detection systems." Journal of Ambient Intelligence and Humanized Computing (2020): 1-18.

[47] Dridi, Lobna, and Mohamed Faten Zhani. "SDN-guard: DoS attacks mitigation in SDN networks." 2016 5th IEEE International Conference on Cloud Networking (Cloudnet). IEEE, 2016.

[48] Scott-Hayward, Sandra, Gemma O'Callaghan, and Sakir Sezer. "SDN security: A survey." 2013 IEEE SDN For Future Networks and Services (SDN4FNS). IEEE, 2013.

[49] Dayal, Neelam, et al. "Research trends in security and DDoS in SDN." Security and Communication Networks 9.18 (2016): 6386-6411.

[50] Kiourkoulis, Stefanos. "DDoS datasets: Use of machine learning to analyse intrusion detection performance." (2020).

[51] Hussain, Yasar Shahid. "Network Intrusion Detection for Distributed Denial-of-Service (DDoS) Attacks using Machine Learning Classification Techniques." (2020).

[52] Lucian Constantin, Attackers abuse exposed LDAP servers to amplify DDoS attacks

[53] Reflection Attacks, di Akamai

[54] Rossow, Christian. "Amplification Hell: Revisiting Network Protocols for DDoS Abuse." NDSS. 2014.

[55] Gasti, Paolo, et al. "DoS and DDoS in named data networking." 2013 22nd International Conference on Computer Communication and Networks (ICCCN). IEEE, 2013.

[56] ATTACKERS USING NEW MS SQL REFLECTION TECHNIQUES, By Akamai, https://blogs.akamai.com/2015ms-sql-reflection-attacks.html

[57] : SECURITY BULLETIN: MS SQL REFLECTION DDOS, akamai's [state of the internet] / security bulletin, disponibile qui: ms-sql-server-reflection-ddos-mc-sqlr-threat-advisory.pdf

[58] MC-SQLR Transport, Miscorsoft

[59] SQL Server Resolution Protocol https://docs.microsoft.com/MC-SQLR

[60] SSDP DDoS Attack, https://www.cloudflare.com/ssdp-ddos-attack/

[61] Radware Emergency Response Team, SSDP DDoS Attack Mitigation, Version 1.0 Rev. 1, November 10, 2014, disponibile qui: https://support.radware.com

[62] DNS Amplification and Reflection Attacks, da cisco.com

[63] Park, Jeman, et al. "Where are you taking me? behavioral analysis of open dns resolvers." 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, 2019.

[64] Bulusu, Pranahita, "Detection of Lightweight Directory Access Protocol Query Injection Attacks in Web Applications" (2015). Master of Science in Computer Science Theses. Paper 1

[65] Bobrov, Anton, and Gilles Bellaton. "Lightweight directory access protocol (LDAP) join search mechanism." U.S. Patent No. 9,608,958. 28 Mar. 2017.

[66] Liron Segal, Old Protocols, New Exploits: LDAP Unwittingly Serves DDoS Amplification Attacks

[67] Alert (TA14-017A) UDP-based amplification attacks. Accessed: 11-07-2017, [Online], disponibile qui: https://us-cert.cisa.gov/ncas/alerts/TA14-017A

[68] NetBIOS name server reflection DDoS attack di akamai

[69] PROTOCOL STANDARD FOR A NetBIOS SERVICE ON A TCP/UDP TRANSPORT: DETAILED SPECIFICATIONS, RFC 1002 (Internet Standard), RFC - Internet Standard (March 1987; Errata) Also known as STD 19, disponibile qui rfc1002

[70] Denial of Service Attack in NetBIOS Services, Software Engineering Institute CERT Coordination Center, Vulnerability Note VU#32650

[71] Douglas R. Mauro & Kevin J. Schmidt. (2001). Essential SNMP (1st ed.). Sebastopol, CA: O'Reilly & Associates. pagine: [2]

[72] Paxson, Vern. "An analysis of using reflectors for distributed denial-of-service attacks." ACM SIGCOMM Computer Communication Review 31.3 (2001): 38-47.

[73] OHMORI, Motoyuki. On a SNMP DoS attack against Vulnerable Architecture of Network Equipment. Vol. 2016. SIG Technical Reports, 2016.

[74] portmap Protocol , da Oracle

[75] Character Generator Protocol, RFC 864

[76] Xiaoming, Li, Valon Sejdini, and Hasan Chowdhury. "Denial of service (dos) attack with udp flood." School of Computer Science, University of Windsor, Canada (2010).

[77] Echo Protcol, J. Postel, May 1983, disponibile qui: RFC 862

[78] Network Time Protocol, RFC 958

[79] Czyz, Jakub, et al. "Taming the 800 pound gorilla: The rise and decline of NTP DDoS attacks." Proceedings of the 2014 Conference on Internet Measurement Conference. 2014.

[80] THE TFTP PROTOCOL (REVISION 2), RFC 1350

[81] Sieklik, Boris, Richard Macfarlane, and William J. Buchanan. "Evaluation of TFTP DDoS amplification attack." computers & security 57 (2016): 67-92.

[82] Ahmed K. Al-Shammari, et al., Distributed Denial of Service Attack Categories in Software-Defined Networks, Australian Journal of Basic and Applied Sciences. 12(11): 25-28. DOI: 10.22587/ajbas.2018.12.11.6, disponibile tramite Australian Journal of Basic and Applied Sciences

[83] Dharmadhikari, Chinmay, et al. "A study of DDoS attacks in software defined networks." IRJET 6.12 (2019).

[84] 1996 CERT Advisories, Software Engineering Institute, Carnegie Mellon University, 21 CA-1996-21: TCP SYN Flooding and IP Spoofing Attacks , pagina 5, pagina 123, WhitePaper

[85] Sonar, Krushang, and Hardik Upadhyay. "A survey: DDOS attack on Internet of Things." International Journal of Engineering Research and Development 10.11 (2014): 58-63.

[86] Kolahi, Samad S., Kiattikul Treseangrat, and Bahman Sarrafpour. "Analysis of UDP DDoS flood cyber attack and defense mechanisms on Web Server with Linux Ubuntu 13." 2015 International Conference on Communications, Signal Processing, and their Applications (ICCSPA'15). IEEE, 2015.

[87] Liu, Hongyu, and Bo Lang. "Machine learning and deep learning methods for intrusion detection systems: A survey." applied sciences 9.20 (2019): 4396.

[88] Xin, Yang, et al. "Machine learning and deep learning methods for cybersecurity." Ieee access 6 (2018): 35365-35381.

[89] Dayan, Peter, Maneesh Sahani, and Grégoire Deback. "Unsupervised learning." The MIT encyclopedia of the cognitive sciences (1999): 857-859.

[90] Zhu, Xiaojin Jerry. "Semi-supervised learning literature survey." (2005).

[91] Barnes, Lindsey R., et al. "False alarms and close calls: A conceptual model of warning accuracy." Weather and Forecasting 22.5 (2007): 1140-1147.

[92] Ruuska, Salla, et al. "Evaluation of the confusion matrix method in the validation of an automated system for measuring feeding behaviour of cattle." Behavioural processes 148 (2018): 56-62.

[93] Florkowski, Christopher M. "Sensitivity, specificity, receiver-operating characteristic (ROC) curves and likelihood ratios: communicating the performance of diagnostic tests." The Clinical Biochemist Reviews 29.Suppl 1 (2008): S83.

[94] Zhu, Wen, Nancy Zeng, and Ning Wang. "Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations." NESUG proceedings: health care and life sciences, Baltimore, Maryland 19 (2010): 67.

[95] Fawcett, Tom. "An introduction to ROC analysis." Pattern recognition letters 27.8 (2006): 861-874.

[96] Tavallaee, Mahbod, et al. "A detailed analysis of the KDD CUP 99 data set." 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, 2009.

[97] Hamid, Yasir, M. Sugumaran, and V. R. Balasaraswathi. "Ids using machine learning-current state of art and future directions." Current Journal of Applied Science and Technology (2016): 1-22.

[98] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Trans. Inf. Syst. Secur., 3(4):262–294, 2000.

[99] , Gniewkowski, Mateusz. "An overview of DoS and DDoS attack detection techniques." International Conference on Dependability and Complex Systems. Springer, Cham, 2020.

[100] Siddique, Kamran & Akhtar, Zahid & Khan, Farrukh & Kim, Yangwoo. (2019). KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research. Computer. 52. 41-51. 10.1109/MC.2018.2888764.

[101] Brown, Carson, et al. "Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict." 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. IEEE, 2009.

[102] Auria, Laura, and Rouslan A. Moro. "Support vector machines (SVM) as a technique for solvency analysis." (2008).

[103] Aouedi, Ons, Mohamed Anis Bach Tobji, and Ajith Abraham. "An ensemble of deep auto-encoders for healthcare monitoring." International Conference on Hybrid Intelligent Systems. Springer, Cham, 2018.

[104] Aouedi, Ons, Kandaraj Piamrat, and Dhruvjyoti Bagadthey. "A Semi-supervised Stacked Autoencoder Approach for Network Traffic Classification." 2020 IEEE 28th International Conference on Network Protocols (ICNP). IEEE, 2020.

[105] Dagum, E. Bee. Analisi delle serie storiche: modellistica, previsione e scomposizione. Springer Science & Business Media, 2001.

[106] Liu, Hui. Robot Systems for Rail Transit Applications. Elsevier, 2020.

[107] Sharma, Sagar, and Simone Sharma. "Activation functions in neural networks." Towards Data Science 6.12 (2017): 310-316.

[108] Kim, Jihyun, and Howon Kim. "An effective intrusion detection classifier using long short-term memory with gradient descent optimization." 2017 International Conference on Platform Technology and Service (PlatCon). IEEE, 2017.

[109] Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6.02 (1998): 107-116.

[110] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[111] Greff, Klaus, et al. "LSTM: A search space odyssey." IEEE transactions on neural networks and learning systems 28.10 (2016): 2222-2232.

[112] Vaswani, Ashish, et al. "Attention is all you need." arXiv preprint arXiv:1706.03762 (2017).

[113] Wang, Haohan, and Bhiksha Raj. "On the origin of deep learning." arXiv preprint arXiv:1702.07800 (2017).

[114] Lukoševičius, Mantas, and Herbert Jaeger. "Reservoir computing approaches to recurrent neural network training." Computer Science Review 3.3 (2009): 127-149.

[115] Li, Xiao-Dong, John KL Ho, and Tommy WS Chow. "Approximation of dynamical time-variant systems by continuous-time recurrent neural networks." IEEE Transactions on Circuits and Systems II: Express Briefs 52.10 (2005): 656-660.

[116] Mikolov, Tomáš, et al. "Recurrent neural network based language model." Eleventh annual conference of the international speech communication association. 2010.

[117] Chen, Serena H., and Carmel A. Pollino. "Good practice in Bayesian network modelling." Environmental Modelling & Software 37 (2012): 134-145.

[118] Friedman, Nir, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers." Machine learning 29.2 (1997): 131-163.

[119] Chen, Serena H., and Carmel A. Pollino. "Good practice in Bayesian network modelling." Environmental Modelling & Software 37 (2012): 134-145.

[120] Janssens, Davy, et al. "Improving performance of multiagent rule-based model for activity pattern decisions with Bayesian networks." Transportation research record 1894.1 (2004): 75-83.

[121] Robinson, R. W. (1973), "Counting labeled acyclic digraphs", in Harary, F. (ed.), New Directions in the Theory of Graphs, Academic Press, pp. 239–273. See also Harary, Frank; Palmer, Edgar M. (1973), Graphical Enumeration, Academic Press, p. 19, ISBN 978-0-12-324245-7.

[122] Leray, Philippe, and Olivier Francois. "BNT structure learning package: Documentation and experiments." Laboratoire PSI, Universitè et INSA de Rouen, Tech. Rep (2004).

[123] Zhang, Zhongheng. "Introduction to machine learning: k-nearest neighbors." Annals of translational medicine 4.11 (2016).

[124] Hu, Li-Yu, et al. "The distance function effect on k-nearest neighbor classification for medical datasets." SpringerPlus 5.1 (2016): 1-9.

[125] Short, R., and Keinosuke Fukunaga. "The optimal distance measure for nearest neighbor classification." IEEE transactions on Information Theory 27.5 (1981): 622-627.

[126] Weinberger, Kilian Q., and Lawrence K. Saul. "Distance metric learning for large margin nearest neighbor classification." Journal of machine learning research 10.2 (2009).

[127] Jirina, Marcel, M. J. Jirina, and K. Funatsu. "Classifiers based on inverted distances." New fundamental technologies in data mining. Vol. 1. InTech, 2011. 369-387.

[128] Kim[1], J. I. N. H. O., B. S. Kim, and Silvio Savarese. "Comparing image classification methods: K-nearest-neighbor and support-vector-machines." Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics. Vol. 1001. 2012.

[129] Ring, Markus, et al. "A survey of network-based intrusion detection data sets." Computers & Security 86 (2019): 147-167.

[130] Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani. "Toward generating a new intrusion detection dataset and intrusion traffic characterization." ICISSp. 2018.

[131] Tajbakhsh, Arman, Mohammad Rahmati, and Abdolreza Mirzaei. "Intrusion detection using fuzzy association rules." Applied Soft Computing 9.2 (2009): 462-469.

[132] Sheikhan, Mansour, and Maryam Sharifi Rad. "Using particle swarm optimization in fuzzy association rules-based feature selection and fuzzy ARTMAP-based attack recognition." Security and Communication Networks 6.7 (2013): 797-811.

[133] Cheng, Jieren, et al. "Adaptive DDoS attack detection method based on multiple-kernel learning." Security and Communication Networks 2018 (2018).

[134] Chen, Liguo, et al. "Detection of DNS DDOS attacks with random forest algorithm on spark." Procedia computer science 134 (2018): 310-315.

[135] Kim, Meejoung. "Supervised learning-based DDoS attacks detection: Tuning hyperparameters." ETRI Journal 41.5 (2019): 560-573.

[136] Mhamdi, L., et al. "A deep learning approach combining auto-encoder with one-class SVM for DDoS attack detection in SDNs." 2020 IEEE Eighth International Conference on Communications and Networking (ComNet). IEEE, 2021.

[137] Abubakar, Rana, et al. "An effective mechanism to mitigate real-time DDoS attack." IEEE Access 8 (2020): 126215-126227.

[138] Shurman, Mohammad, Rami Khrais, and Abdulrahman Yateem. "DoS and DDoS Attack Detection Using Deep Learning and IDS." INTERNATIONAL ARAB JOURNAL OF INFORMATION TECHNOLOGY 17.4 A (2020): 655-661.

[139] Ujjan, Raja Majid Ali, et al. "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN." Future Generation Computer Systems 111 (2020): 763-779.

[140] Hussain, Faisal, et al. "IoT DoS and DDoS Attack Detection using ResNet." arXiv preprint arXiv:2012.01971 (2020).

[141] Abadi, Martín, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467 (2016). Mentre il per il sito, consultare il seguente link.

[142] Sezione dedicata al ML, visitare il seguente link.

[143] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016. Mentre il per il sito, consultare il seguente link.

[144] Hall, Mark, et al. "The WEKA data mining software: an update." ACM SIGKDD explorations newsletter 11.1 (2009): 10-18.

[145] Yuan, Xiaoyong, Chuanhuang Li, and Xiaolin Li. "DeepDefense: identifying DDoS attack via deep learning." 2017 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, 2017.